Empirical Software Engineering (2026) 31:97
https://doi.org/10.1007/s10664-026-10815-5

Exploring actions, interactions and challenges in software modelling tasks: an empirical investigation with students

Shalini Chakraborty[1] · **Javier Troya**[2] · **Lola Burgueño**[2] · **Grischa Liebel**[1]

Received: 5 August 2024 / Accepted: 28 January 2026 / Published online: 21 March 2026

Abstract

Background Software modelling is a creative yet challenging task. Modellers often find themselves lost in the process, from understanding the modelling problem to solving it with proper modelling strategies and modelling tools. Students learning modelling often get overwhelmed with the notations and tools. To teach students systematic modelling, we must investigate students' practical modelling knowledge and the challenges they face while modelling.
Aim We aim to explore students' modelling knowledge and modelling actions. Further, we want to investigate students' challenges while solving a modelling task on specific modelling tools.
Method We conducted an empirical study by observing 16 pairs of students from two universities and countries solving modelling tasks for one hour.
Results We find distinct patterns of modelling of class and sequence diagrams based on individual modelling styles, the tools' interface and modelling knowledge. We observed how modelling tools influence students' modelling styles and how they can be used to foster students' confidence and creativity. Based on these observations, we developed a set of guidelines aimed at enhancing modelling education and helping students acquire practical modelling skills.
Conclusions The guidance for modelling in education needs to be structured and systematic. Our findings reveal that different modelling styles exist, which should be properly studied. It is essential to nurture the creative aspect of a modeller, particularly while they are still students. Therefore, selecting the right tool is important, and students should understand how a tool can influence their modelling style.

Keywords Software Modelling · Observation Study · UML · Education

1 Introduction

Software modelling holds significant promise for enhancing various aspects of software and systems engineering, such as productivity (Agner and Soares 2013) and cost efficiency (Kirstan and Zimmermann 2010). Despite these advantages, the widespread

Communicated by: Jeffrey C. Carver.

Extended author information available on the last page of the article

adoption of software modelling across the entire field remains limited (Gorschek et al. 2014). Extensive research has explored the reasons behind the limited adoption, revealing issues like subpar code generation, inadequate tool support, and a lack of guidance or training (Forward et al. 2010; Whittle et al. 2013, 2014; Mohagheghi et al. 2013; Liebel et al. 2018b, a). Specifically, it has been suggested that engineers are reluctant to embrace modelling because it requires excessive effort and offers little usefulness, a view shaped by their educational background (Gorschek et al. 2014; Torchiano et al. 2013). Within the educational domain, substantial efforts have been dedicated to understanding how to teach modelling within university curricula, exemplified by works such as (Westphal 2019; Burgueño et al. 2018b; Kolovos and Cabot 2016; Paige et al. 2014; Whittle and Hutchinson 2011; Stikkolorum et al. 2015; Liebel et al. 2016). This body of literature often takes the form of proposed course designs (Schmidt et al. 2014; Westphal 2019), experiential reports detailing challenges or best practices, particularly concerning tools (Burgueño et al. 2018b; Kolovos and Cabot 2016; Paige et al. 2014; Whittle and Hutchinson 2011), or papers presenting quantitative studies on student opinions (Stikkolorum et al. 2015; Liebel et al. 2016). Collectively, this wealth of knowledge serves as a valuable source of experience and inspiration for the effective teaching of modelling practices. However, research on students' perspectives on modelling needs to provide more detailed explanations, such as specific challenges and benefits regarding modelling assignments, modelling tools, syntax, and semantics. Furthermore, experience reports from researchers with a specific focus on software and systems modelling may run the risk of being unrepresentative for instructors lacking such a focus. Given that most students will transition into professional roles post-graduation, it becomes crucial to comprehend their perceptions. It is essential to understand the challenges students currently face and explore the process of modelling that students practice in universities.

In our prior research (Chakraborty and Liebel 2023), we engaged in 16 interviews with students and 5 interviews with instructors to explore students' perceived understanding of modelling and the challenges they encountered. The findings of our study indicate that students recognise specific advantages of modelling, particularly in areas where informal models suffice, such as gaining a system overview or conceptual comprehension of the problem domain. However, the majority of the students remains skeptical about the value of modelling in detailed system design. Many students expressed challenges related to issues such as selecting the appropriate notation, determining what aspects to express in the models, and understanding how to apply modelling techniques to unfamiliar domains.

In this paper, we dig deeper into students' modelling practices by undertaking an observational study. We conducted two modelling sessions where students participated in pairs and solved modelling tasks. Our goal here is to observe students' modelling actions and interactions during the modelling task and explore the challenges experienced by an individual or pair.

The study follows two research questions (RQs):

- RQ1: What actions do students take on a modelling tool to solve a task?
- RQ2: What challenges do students face while solving modelling tasks?

RQ1 focuses on the actions students undertake within a modelling tool as they solve a given task. Understanding the specific steps students employ in a modelling tool provides valuable insights into their cognitive processes and problem-solving strategies. RQ2 investigates the challenges students experience while engaging in modelling tasks, including challenges based on students' modelling perspectives, tool knowledge and the communicative dynamics between pairs of students engaged in collaborative problem-solving.

This paper presents findings from an observational study conducted with 32 students from two different universities, Reykjavik University, Iceland, and University of Malaga, Spain. The students worked in pairs, engaged in solving tasks using two modelling tools, namely MagicDraw[1] and PlantUML[2].

Our results show that students need more knowledge of structuring and completeness in modelling. While students are confident drawing class diagrams, sequence diagrams still pose a challenge to them. In both cases, students need to improve how they read and process the problem before modelling its solution. We observed how modelling tools influence students' modelling styles and can be used to provide students with the necessary confidence and creativity. Finally, we formed a set of guidelines based on these observations, which will help in modelling education and help students adhere to the practical skills of modelling.

The rest of the paper is structured as follows. In Section 2, we discuss the existing literature related to modelling education, students' modelling experience and observational studies. In Section 3, we describe the pilot studies, observational study setup, data collection and data analysis strategies. In Section 4, we mention the summarised results of the observational study, followed by a detailed discussion of the results in Section 5. Finally, we conclude our paper in Section 6.

2 Related Work

Model-Driven Engineering (MDE) or Model-based Engineering (MBE) is the current state-of-the-art in software abstraction, where models are used as primary artifacts in the software engineering process. It has gained popularity in academia and industry for managing the complexity of modern software and is considered a significant advancement in software development (Hutchinson et al. 2011). Although the benefits of MBE are often considered obvious, higher abstraction levels do not guarantee better software and while code generation may boost productivity, the effort to develop models and make manual modifications can negate this benefit (Hutchinson et al. 2011). Regardless of the success of MBE, abstract models are crucial to the future of software development. When it comes to software modelling, UML (Unified Modelling Language) and UML-based development methods have become de facto standards in SE. Text books on UML and object-oriented design commonly refer to heuristics for creating UML diagrams, e.g., Rumbaugh et al. (1991); Bruegge and Dutoit (2009); Abbott (1983).

In the following, we discuss literature related to UML modelling in education, especially modelling done by students, modelling challenges and observational studies as a method of investigation.

[1] https://www.magicdraw.com/

[2] https://plantuml.com/

2.1 Modelling in Education and Modelling by Students

Numerous studies have explored modelling in education, with a predominant focus on teaching modelling methodologies (Westphal 2019; Gilson 2018; Gonnord and Mosser 2018) or presenting experience reports from modelling courses (Akayama et al. 2013; Paige et al. 2014; Kolovos and Cabot 2016; Burgueño et al. 2018a). In (Westphal 2019), Westphal presented the design of a software engineering course heavily influenced by and focused on UML modelling. In (Gilson 2018), the authors proposed a course where students acted both as language designers and users to evaluate the usability of software language engineering. Gonnord and Mosser (2018) took a reverse approach, guiding students from low-level C code to designing a modelling language workbench.

Akayama et al. (2013) shared their experiences and opinions on tool use in software modelling education. The paper describes various approaches taken by the individual authors and discusses factors such as modelling tools versus pen and paper, the conflict between design and programming concepts, and methods to measure the quality of models. Paige et al. (2014) discussed what they consider to be bad practices in teaching modelling. They identified issues such as covering too broad a range of modelling-related topics and focusing on syntax instead of semantics. Similarly, Kolovos and Cabot (2016) presented a corpus of use cases for courses teaching model-based engineering (MBE). In (Burgueño et al. 2018a), the authors discuss previous challenges with modelling encountered in various software engineering courses. They then present a case study in which students define a system and its views, simulate them, examine their relationships, and conduct several types of analyses on the complete system specifications. The authors also include a survey to know students' feedback.

While some surveys assess students' overall experiences with modelling (Ciccozzi et al. 2018; Agner et al. 2019), there remains a scarcity of studies providing a detailed account of students' experiences in modelling or reporting results derived from students actively creating models or solving modelling problems. In the domain of business process modelling, studies like (Pinggera et al. 2013) have documented the modelling experiences of 115 students, highlighting three distinct modelling styles, "*We could distinguish (1) an "efficient modelling style" characterized by a limited time needed to think about the modelling task, and a fast rate of adding elements to the model; (2) a "layout-driven modelling style" which involves much time in creating a comprehensible layout while being less efficient in creating the model; and (3) an "intermediate modelling style" that is neither particularly efficient nor invests particularly into model layout"*.

2.2 Modelling Challenges by Students

Multiple studies reported students' challenges regarding software modelling. Stikkolorum et al. (2015) explore student difficulties and modelling strategies within UML Class diagrams by providing students with a specialized UML editor featuring feedback mechanisms. The findings identify four distinct strategies (depth-less, depth-first, breadth-first, and ad hoc), representing various approaches to constructing class diagrams based on task requirements, encompassing associations, attributes, and problem-solving elements. Reuter et al.'s

research (Reuter et al. 2020) delves into students' challenges with UML diagrams across two modelling courses, resulting in the compilation of a comprehensive catalogue of issues associated with UML diagrams. Agner et al. (2019) broaden the scope with a survey on modelling tool use among 117 students, uncovering issues such as a lack of feedback, diagram-drawing difficulties, and tool complexity. In a different study, Hammouda et al. (2014) compare the utilization of modelling tools to traditional pen-and-paper methods. Through a survey, they evaluate students' perceptions of the differences, ultimately finding no clear advantage for either approach.

2.3 Observational Studies in Literature

Observational studies are beneficial in investigating participants' thinking processes and actions. In their paper, Aniche et al. (2021) observe 13 developers thinking aloud different test cases. Their findings show three different strategies used by developers while testing and explain the reason behind these strategies. Dekel and Herbsleb (2007) perform an observational study of several collaborative design exercises. Their results show the popularity of the ad-hoc nature of collaborative design and the behaviours of different design teams in different contexts and used communication mediums. In their study, Mangano et al. (2014) observe eight professional software designers doing 14 hours of design activities on whiteboards to analyse the communication between designers and whiteboard sketching. Carver et al. report in their study (Carver et al. 2003) an observational approach where an experimental subject executing a procedure was observed by another individual. The study was conducted in pairs, with the participant implementing the technology referred to as the **executor**, and the individual observing the technology's application termed as the **observer**. Importantly, the role of the observer did not involve collaborative work with the executor in applying the technology; rather, their responsibility was to ensure that the executor adhered faithfully to the technology's procedural steps. The observer also took notes on the executor's utilization of the technology, highlighting instances where challenges were encountered. After a certain point, the roles of executor and observer were swapped. The authors claimed that this observational study technique provided a level of detail regarding individual process steps and their efficacy that is challenging to obtain through traditional post-experiment questionnaires. A similar strategy has been used in pair programming for ages. Pair programming (Williams and Kessler 2003) is becoming a popular practice in software development (Williams et al. 2003). In their survey, Begel et al. investigate the advantages of pair programming (Begel and Nagappan 2008), identifying benefits such as enhanced code understanding, increased creativity and brainstorming, and improved design.

Motivated by these insights, we embraced a pair programming-style observational study approach to achieve similar benefits in the context of modelling. Acknowledging modelling as an inherently creative task, especially for our predominantly first-year undergraduate participants, collaborative problem-solving emerges as a valuable asset. In contrast to the approach outlined in Carver et al. (2003), in our study the **observer** assumes the role of a **navigator** who directed the **executor**. This shift in dynamics differs from the traditional **executor** role, to a **driver**, emphasizing the collaborative and interactive nature of our pair programming-inspired methodology.

3 Method

Observational studies constitute a cornerstone in scientific research, providing a valuable method for investigating and understanding natural phenomena in their real-world context. Unlike experimental studies, which involve deliberate manipulation of variables, observational studies involve the passive observation of subjects in their natural settings. For our study we followed the method proposed by Williams and Kessler (2003); Williams et al. (2003) inspired by pair programming, which is a popular practice in software development. Begel et al. explore the benefits of pair programming in a survey (Begel and Nagappan 2008). They identified code understanding, creativity and brainstorming, and better design among the benefits of this method. We decided to follow a pair programming approach (in our case *pair modelling*) for similar advantages. We are also following constructive interaction (Andriessen et al. 2003; Baker 1999; Miyake 1986) and collaborative learning (Veerman et al. 1999) for data analysis. Furthermore, we are using the Empirical Standards for Software Engineering Research (Ralph et al. 2020)[3]. In particular we are using author checklist provided in the *General Standard* category, which contains the specific criteria that can be used by authors to conduct and report research. Understanding and solving a problem with software modelling is a creative task. Modellers read the problem from their perspective and conclude the solution based on individual thinking and knowledge of the domain. The modellers' style, apart from modelling tools and notations, influences the solution. There are no significant studies in Software Engineering (SE) regarding software modelling style. A few studies in business process modelling (e.g., Pinggera et al. 2012) lead the way towards learning about style.

Our study mainly focuses on modelling actions, like *Add, Delete, Edit*, and challenges to understanding students' modelling process. We detail our study steps in the following subsections, from the pilot to the final study setup.

3.1 Pilot Study 1

For pilot 1, we utilised the tool Papyrus and a plug-in extension called ModRec (Ragnarsson et al. 2021) to create the models and gather data, respectively. With ModRec, modelling actions can be automatically captured and can be viewed as a separated file. The study was promoted to undergraduate software engineering students at Reykjavik University (RU), resulting in three volunteers. Pilot 1 was conducted in September, 2022. All three participants were briefed on the pilot's objectives and had the option to withdraw at any time. They did not receive any compensation for their participation. It was also clarified that their performance would not have any impact on their grades, neither for good nor bad. All three were first-year students familiar with UML modelling.

The students were tasked with drawing a class diagram to depict the domain of a restaurant food court system. They were instructed to use Eclipse Papyrus as their modelling tool and were given a demonstration of the tool's interface and the ModRec plug-in. Several requirements were provided, and they were allotted 30 minutes to complete the diagram. The details of the problem description used in Pilot 1 can be found in the Appendix A. We observed students facing difficulties with Papyrus, realising they invested most of their time

[3] https://www2.sigsoft.org/EmpiricalStandards/tools/

understanding the tool instead of focusing on the task. Additionally, we realised that the example domain, a restaurant food court, was not interesting enough to students.

Based on these observations, we made three changes: (i) using a more relevant and exciting problem domain, (ii) adding not just structural, but behavioural aspects to the modelling problem and (iii) not prescribing any modelling tool, letting the participants select one themselves.

3.2 Pilot Study 2

After integrating in our study the changes from the pilot study 1, we selected a dating app as the problem domain (more details below) and created two tasks involving drawing a class diagram and two sequence diagrams. We introduced structural and behavioural modelling in the pilot 2 to get more details about the students' modelling process. The first task was to draw a class diagram to capture the app's domain. The second task was to draw two sequence diagrams to model two dating app functions.

We select the dating app as our domain to get students interested in solving the problem. Our study is based on student volunteers, and getting their attention towards the tasks is necessary. We looked through past courses and realised that it is not a traditional problem domain, and students might be curious to attempt it. Also, we decided to go with one static and one dynamic UML diagram type. Prior experience (Chakraborty and Liebel 2023) suggests that class diagrams are the easiest UML diagram type for students to understand, and sequence diagrams are the hardest. Lopes et al. (2019) find that students find sequence diagrams not easy to use alone, as they need other diagrams, such as use case and class diagrams. In (Reuter et al. 2020), Reuter et al. survey students' problems with UML diagrams. Regarding sequence diagrams, the authors find students need help with the order of elements, actions in a sequence diagram, needing help to identify interacting classes that are needed as objects for the sequence diagram, etc. Inspired by these studies, we decided to use class and sequence diagrams. Since we let students use their preferred modelling tool, we decided to record their screens to get the modelling actions following a method equivalent to pair programming as stated above (Williams and Kessler 2003; Williams et al. 2003).

For the first problem, student A plays the driver, and student B is the navigator. That means student A solves the problem by drawing diagrams, while student B navigates through the problem, giving insights and inspecting the drawing. For the second problem, students switched roles. Also, we thought by pairing up, future participants might feel more motivated to join the study. We are aware of the shortcomings of pair programming, such as personality clashes, and bad communication affecting the design. However, we wanted to see how communication between a pair influences the design.

The study was divided into four sections: Reading the full problem description (10 minutes), one of the participants solving task 1 with a class diagram (20 minutes), taking a small break (10 minutes), the other participant solving task 2 with sequence diagrams (20 minutes). The details of the problem description used in Pilot 2 can be found in the Appendix A. We then conducted the pilot study 2 in November, 2022 with two PhD students of the computer science department at RU in . Both PhD students had prior experience of UML. The participation was voluntary, they had the option to withdraw at any time and did not receive any compensation for their participation. The two volunteer PhD students selected a tool of their own (draw.io), based on their previous experience with the tool.

After observing the second pilot study, we decided to change the time frame and a few details in the problem description used for the study. We reduced the time for reading the problem and break to 5 minutes each and added 5 minutes each to the problem-solving. We realised that giving participants an open problem description was not a good idea, as the two volunteers in the second pilot study took a safe path and created generic diagrams without going into details. For the main study, we recruited volunteers from two universities (11 pairs from University of Malaga, Spain, and 5 pairs from RU, Iceland).

In the next three sections, we discuss the study design and setup (Section 3.3), data collection (Section 3.4) and data analysis (Section 3.5).

3.3 Study Design and Study Setup

We performed the final study at the University of Malaga (UMA) in Spain and RU in Iceland. In UMA, we considered Software modelling and design, a bachelor course in 3rd year with 110 students. The course is about structural modelling with class diagrams and behavioural modelling with sequence diagrams and state machines. It is taught how to model the transition of a system over time. Design patterns and the conversion of class diagrams into object-oriented code are also studied. We advertised the study in the course. One of the authors of this paper (a lecturer in the course) presented details of the study and research goals earlier in the course. Later, we asked for voluntary student participants, and 11 pairs signed up for the study.

In RU, we considered a bachelor course in the second semester of the Software Engineering program, with 51 students. The course centres on analysing, modelling, and designing software systems, employing an object-oriented approach and utilising UML. Throughout the course, students learn class, sequence, state machine, activity, and component diagrams. Additionally, the curriculum includes the practical implementation of the designed systems using the Python programming language. We made the study the last assignment in the course. Five pairs volunteered for the study. Thus, we asked them to follow the study setup. The rest of the students finish the problem as individual assignments for the course.

Table 1 shows the differences between pilot study 1, pilot study 2 and the main study, explaining the changes in our approaches. For the final study, we changed the problem description from pilot study 2 and added listed requirements to give students more specific details and restrictions about the problem domain. Then, we kept the tasks similar to pilot 2. The details of the final problem description can be found in the Appendix .

Students signed up in pairs, selecting their own partners. In UMA, participation was voluntary, and participants could withdraw from the study at any time. The students were orally informed about the purpose of the study: to carry out a research experiment. They knew and agreed that their data would be anonymised and the results could be part of research publications. The task was undertaken near the end of the semester, in mid-December 2022. Students in RU signed a consent form before participating in the study. Participation was voluntary, and participants may withdraw from the study at any time, without penalty. Furthermore, participants could deny the researchers conducting the study the right to use the collected data (from their participation). Since the task was part of the course assignment, contributing 3 per cent to the final grade, participants received grades according to course

Table 1 The changes between Pilot study 1,2 and the main study

Studies	Task	Tools	Time and participants	Output
Pilot study 1	Modelling the domain of a restaurant food court system with class diagram	Papyrus and ModRec	30 mins, 3 individual participants	Log file from ModRec and resulting diagram
Changes: modifying the task by changing the problem description, adding pair-modelling, increasing the time and no restriction on tools				
Pilot study 2	Modelling the domain of a dating app with class diagram and modelling two functionalities of the app with sequence diagram	Draw.io (participants' choice)	60 mins, 10 minutes to read the problem, 20 minutes for class diagram, 10 minutes break and 20 minutes for sequence diagram, one pair	Screen recording with audio and resulting diagrams
Changes: modifying the task by adding detailed requirements to the same problem description, modifying the time distribution				
Main study	Modelling the domain of a dating app with class diagram and modelling two functionalities of the app with sequence diagram	MagicDraw and PlantUML (participants' choice)	60 mins, 5 minutes to read the problem, 25 minutes for class diagram, 5 minutes break and 25 minutes for sequence diagram, 16 pairs	Screen recording with audio and resulting diagrams

rules and feedback on their modelling solutions. The study was conducted in April 2023. At both institutions, no ethics approval was needed for this study and, therefore, none was sought.

Each pair got one laptop and a recorder to record their conversations, and we also asked them to use screen recording software so we could record their modelling activities. At UMA, all students used Magicdraw as they were familiar with that tool. At RU, all students used PlantUML as it was the tool used in that course. Students received the problem description in English. To get their authentic reaction, make them feel comfortable and avoid language barriers, we gave them the option to converse in their mother tongue or in English. All 11 pairs from UMA conversed in Spanish, three out of five pairs in RU spoke Icelandic, and two spoke English.

3.4 Data Collection

11 pairs from Malaga participated in the study in Spain. Similarly, five pairs from Reykjavik participated in the study at the end of their course in Iceland. In Malaga, the study was conducted in December 2022, and four months later, in April 2023, the study was conducted in Iceland. After a brief analysis of the data collected from all 16 pairs and an extensive analysis with two pairs each from UMA and RU, we concluded the study. We observed a sufficient similarity in the data, indicating a saturation point (Baltes and Ralph 2022). The repetition of data suggested that further collection would be redundant, confirming that we had reached an adequate sample size. Also, we found sufficient diversity, depth and probable issues demonstrating validity of the collected data.

Students had 5 minutes to read the problem description, 25 minutes to solve the first problem (class diagram), a break of 5 minutes and then 25 minutes to solve the second problem (sequence diagram). We collected three forms of data: (i) *modelling actions through screen recordings*, (ii) *voice recordings of the conversations between each pair*, and (iii) *PDFs of each pairs' final diagrams*. The three forms of data address our RQs directly. The screen recordings summarise actions taken by students while modelling. All three data sets explain the challenges students face while modelling.

3.5 Data Analysis

Our data analysis was conducted in multiple stages, refining our approach iteratively before finalising a structured method applicable to the entire dataset. Initially, we classified the recordings (both screen and voice) into three main sections:

- **Problem Reading Phase (5 minutes)**: This captures the initial stage when a pair reads and discusses the problem statement.
- **Class Diagram Modelling Phase (25 minutes)**: This includes all actions related to constructing the class diagram.
- **Sequence Diagram Modelling Phase (25 minutes)**: This involves creating two sequence diagrams following the class diagram.

Each of these sections was further categorised based on our research questions (RQs) into three key dimensions:

- **Actions**: This includes observable modelling actions such as Adding, Deleting, and Editing elements in the diagram.
- **Communication**: This covers verbal interactions between the pair, including planning discussions, agreements, disagreements, and decision-making points that influenced their modelling actions.
- **Challenges**: This consists of verbal expressions or modelling actions that indicate difficulties encountered during the process.

For action analysis, we systematically recorded every modelling action performed by a pair in the modelling tool, such as adding a class, deleting an attribute, or adding an association. Each action was logged with its corresponding timestamp to capture the modelling progres-

sion. For communication analysis, we extracted dialogues between pairs, linking them with timestamps to understand the reasoning behind their actions. This included conversations reflecting planning, conflict resolution, or significant decision-making moments. For challenge analysis, we identified any verbal or modelling actions that indicated difficulties students faced during diagram construction. Figure 1, shows our initial approach. The arrows indicate the sequence from reading the problem to modelling the class diagram to modelling the sequence diagrams.

We analysed four pairs (2 from UMA and 2 from RU) with the above-mentioned approach. However, we realised distributing the data like this was tiresome, and threats of researcher bias were increasing. Hence, we refined our approach again and focused on the class and sequence diagrams this time, precisely how a pair approached and solved those modelling tasks.

Figure 2 describes our final analysis framework. We dedicated the first step towards data collection. Next we break down students modelling into three distinct actions, *Add, Delete* and *Edit*, based on the screen recordings. An *Add* action occurs when students incorporate new elements into their model, ranging from classes, attributes, and methods to associations between classes, participants, and messages–both direct and return. Conversely, a *Delete* action is logged when students remove any component from their models. Lastly, an *Edit* action encompasses modifications to existing *Add* actions. This involves alterations such as changing the name of a class or message, adjusting multiplicities, and redirecting associations between classes. This structured classification provided a clear framework for analysing how students' models evolved throughout the study.

Then, we focused on each action and incorporated the audio feature in the analysis. That is, we tried to understand the frequency of those actions and the reasons behind their occurrence. Finally, we then looked at students' whole modelling process and the final diagrams to understand what type of challenges students experienced while drawing the diagrams.

Similar to our previous approach, we selected two pairs from each university, performed the analysis, and discussed among the four authors. When we reached an agreement, the first

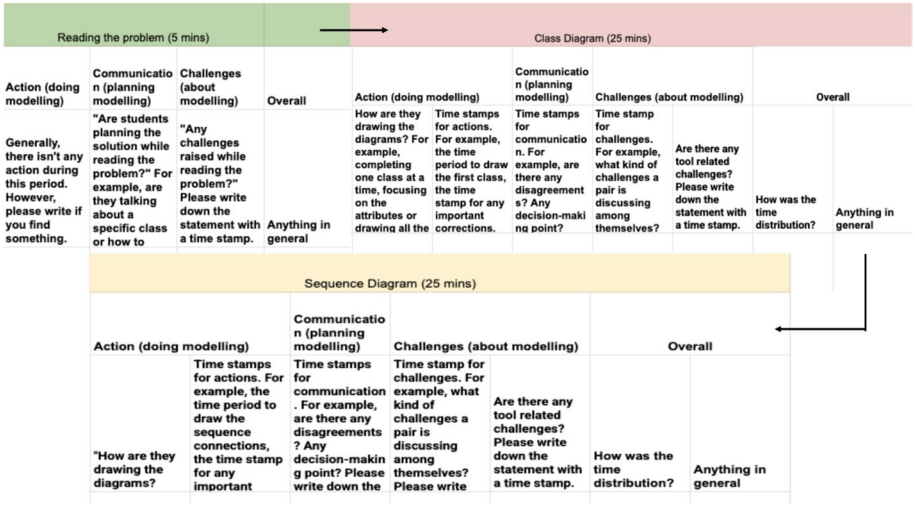


Fig. 1 Initial Data Analysis

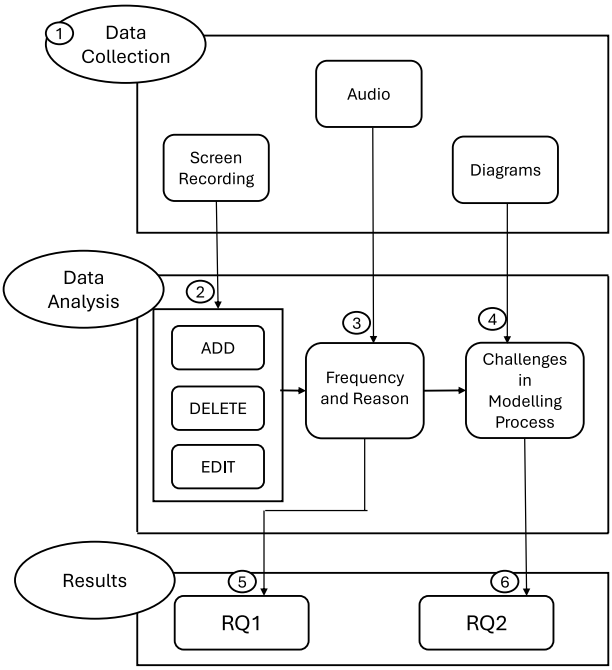


Fig. 2 Data Analysis

author conducted the rest of the data analysis. We provide a complete action log[4] containing all recorded modelling actions across students. This dataset allows for a more granular examination of our findings.

After this step, we qualitatively analysed the distribution of actions over the entire modelling process. That is, we grouped similar distributions into clusters and discussed these among the author team. Additionally, we consulted the communication logs to seek explanations for the time distribution. Based on this analysis, different clusters of modelling behaviour emerged. We hypothesise that these clusters constitute *modelling styles*, i.e., "consistent individual differences in preferred ways of creating and processing software models" (Chakraborty and Liebel 2024).

Based on our complete analysis, we answered the RQs (step 5 and 6). We report them in detail in the next section (Section 4).

3.6 Threats to Validity

The results in this paper are based on observing 16 pairs of students across two universities in two different countries. The collected data was in three different languages. We identified several threats to validity, which we describe below along with the measures we took to mitigate them, classifying the validity threats in three categories as described in Wohlin et al. (2012).

[4] https://doi.org/10.5281/zenodo.15914484

3.6.1 Internal Validity

Internal validity focuses on how confident we can be that the elements of the study actually caused the observed outcome. There may be other factors influencing the outcome, which are beyond our control or have not been measured.

One of the threats in our study is that the students are Icelandic and Spanish, while the problem description was in English. To mitigate this threat, one of the authors, who is Spanish, was present during the study at UMA so that students could clear their doubts. In RU, the original course instructor, who is Icelandic, helped students with questions.

Two additional internal threats were tool selection and problem description. We allowed students to choose their own tools, acknowledging that different tools might influence the results. However, it was important not to restrict students to unfamiliar tools. The text-based and visual approaches of these tools were not predefined by us; rather, we observed how certain tool properties influenced students' modelling. In future experiments, we will carefully consider tool selection. For the problem description, we kept it as simple as possible. While we recognise that problem complexity could pose a potential threat, we mitigated this by conducting two pilot studies to refine and finalise the problem description.

3.6.2 External Validity

External validity is concerned with whether we can generalise the results beyond the scope of our study.

Our study participants are all bachelor students, and while doing the study, they were active in a course which involved software modelling with UML. One threat is the modelling background of the participants and the use of a UML diagram in the study. To mitigate this, we ensured that all participants were enrolled in the same course within their respective universities. Additionally, we structured the study to include 16 pairs working on the same problem, allowing us to collect multiple diagrams. Furthermore, to explore potential differences, we included students at different academic levels: senior-year students from UMA and first-year students from RU. This intentional sampling provides insight into how students at different stages of their education approach the task. We also used two modelling diagrams, class and sequence, in the study settings to capture both students actions for both structural and behavioural modelling. Despite all participants are bachelor students, we believe this threat is minimised because students were active in a course involving software modelling with UML, and they had been doing similar exercises during the course, so all the concepts and knowledge were fresh in their minds. Therefore, we believe the same study involving software engineers might have had similar results.

Another threat is given by the fact that we opted for doing the experiment with volunteers, which introduces a selection bias. We observed that, even if some participants failed the course or did not take the final exam (which means that they were not necessarily high performers), in general, participants performed slightly better than the general student group[5]. Alternative approaches could have been followed. For instance, (i) Requiring participation from all students. However, this introduces different risks, like

[5] Participants had average grades of 6.9±3.36 points out of 10, while the general group had 5.3±3.27 points.

demotivation, which might negatively affect outcomes; (ii) Employing a stratified random sampling strategy, where students from various performance levels are randomly selected to participate. However, it would require to re-design our courses, since currently we cannot know the students' performance until they take the final exam, after which there are no more lectures.

This qualitative study, involving 32 individuals, provides valuable insights that can be highly relevant and applicable to similar contexts. While claiming generalisability may not be appropriate, especially, that we chose a pair programming setup–an approach not typically used for modelling in industry–our results provide a strong foundation for understanding the phenomenon studied and can potentially inform practices in similar contexts.

3.6.3 Conclusion Validity

Conclusion validity focuses on how confident we can be that the study elements and settings are genuinely related to the observed outcome and whether the study can be repeated.

We kept a flexible study setting for our student participants as we wanted to capture their interactions with each other and with modelling. The study occurred in their university rooms, where students used their preferred modelling tool and modelled with their language. Conclusion validity refers to the extent to which the conclusions drawn from a study are credible and justifiable. It ensures that the relationships identified between variables are genuine and not influenced by external factors or biases. We acknowledge that our study primarily relied on qualitative analysis, specifically in-vivo coding. While this approach allowed us to closely examine students' modelling actions and decision-making processes, we recognise that it introduces potential risks of misinterpretation. We cannot entirely eliminate the presence of researcher bias in our results, however we implemented multiple measures to mitigate this threat as much as possible. For data analysis, we began with a smaller sample size from our data (two pairs from each university) and had all four authors analyse the data. We then discussed our findings among ourselves, and once we reached an agreement, we analysed the rest of the data accordingly. Consistency was maintained throughout our study design, data collection, and analysis between the two universities. For analysing the screen recordings, we used the terms Add, Delete, and Edit, with all four authors defining and agreeing on these terms. Voice recordings presented a challenge due to the involvement of three languages. However, we transcribed the conversations into English and used in-vivo coding to capture the students' perspectives accurately. Additionally, we have now provided a complete action log[6], which enhances transparency and allows others to verify our findings. Regarding participant variability, we acknowledge that we did not explicitly characterise students' prior experience with modelling. While all participants were enrolled in software engineering courses where modelling is a part of the curriculum, we recognise that variations in their prior exposure to modelling could have influenced their actions. Finally, we hypothesise that the observed clusters of modelling behaviour are modelling styles, i.e., that they are *consistent* and related to *individual differences*. However, future studies will have to investigate whether these behaviours are indeed reproduceable and if other factors might lead to these observations.

[6] https://doi.org/10.5281/zenodo.15914484

4 Results

We analysed 16 student pairs from two universities as they solved a modelling task by drawing class and sequence diagrams using two modelling tools. We observed several individual actions taken and formulated different challenges faced by the pairs to solve the modelling problem. Below, we answer our two research questions using the collected data.

4.1 RQ1: What Actions do Students Take on a Modelling Tool to Solve a Task?

To answer RQ1, we analysed students' screen recordings and distinguished styles to draw different diagrams. These individual styles when creating diagrams are caused by students' learning, preferences when solving tasks, cognitive ability and communication between the pair. These individual ways of modeling are on a micro level, detailing how students modelled each element of these UML diagrams. Additionally, the styles observed are specific with respect to class and sequence diagrams and highlight the individuality of students while drawing each type of diagram. For example, we Tables 2 and 3 show a co-occurrency matrix of the actions taken by a pair of students from RU while modeling the class and sequence diagram. This co-occurrence table shows the number of actions (ADD; DEL; EDIT) that were taken consecutively by the students[7]. Combining all the styles observed from students while drawing both class and sequence diagrams reflects the diverse modelling styles among them. In a previous study (Chakraborty and Liebel 2024), we defined modelling style as "**consistent individual differences in preferred ways of creating and processing software models**". The individual preferences displayed while drawing both diagrams revealed hints of distinct modelling styles. In their study, Pinggera et al. (2013) conducted two large-scale modelling sessions involving 115 students. Their study is similar to ours in terms of recording modelling sessions and considering Add, Edit, and Delete as three distinct modelling actions. After cluster analysis, they reported three distinct model-

Table 2 Co-occurrency matrix - Class Diagram - RU_Pair1

Row	ADD	ADD ASSOC	ADD CLASS	DEL	EDIT
ADD	21	0	9	4	17
ADD ASSOC	0	1	0	0	1
DEL	0	0	0	0	2
EDIT	0	0	0	0	8

Table 3 Co-ocurrency matrix - Sequence Diagram - RU_Pair1

	ADD ACTOR	ADD LOOP	ADD MESSAGE	EDIT	DELETE	OTHER
ADD ACTOR	4	3	14	4	0	0
ADD LOOP	0	0	12	5	1	2
ADD MESSAGE	6	12	64	20	7	14
EDIT	0	2	5	6	1	1
DELETE	0	3	26	15	6	2
OTHER	8	0	6	5	0	1

[7] For reproducibility reasons, the corresponding R script to obtain this data (or data for any other pair) is provided here:https://doi.org/10.5281/zenodo.15914484.

ling styles for business process modelling. We recognise that it is premature to categorise the actions observed in our study as styles based on just two diagrams and 32 students. But unlike Pinggera et al. (2013), we recorded students conversation as well, which gave us more depth into their modelling actions. **Therefore, we decided to answer RQ1 by presenting these styles as *"modelling preferences"*, which can later be examined with more diagrams and a wider range of problems.**

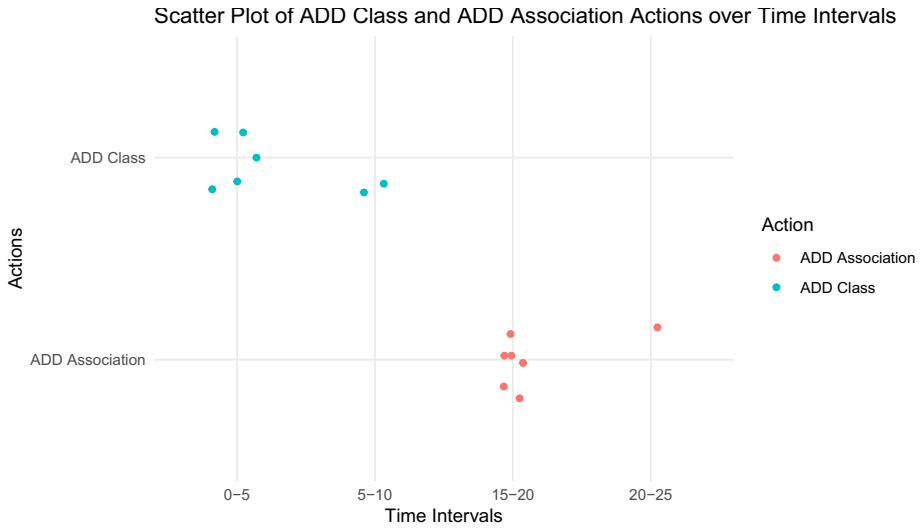
4.1.1 Modelling Preferences for Class Diagram

Students employ diverse strategies when it comes to completing a class with all its attributes and methods, and we observed *three* distinct preferences. *Firstly*, some opt for empty classes, primarily following the "noun identification" method. They identify all the necessary classes for the diagram, draw them, and subsequently add attributes and methods. The *second* category involves students starting with the identification of one class at a time, gradually adding basic attributes before progressing to the next class. Lastly, the *third* preference involves a mixed method, where students begin by creating most of the classes, then move back and forth between them, adding attributes and methods and more classes if needed.

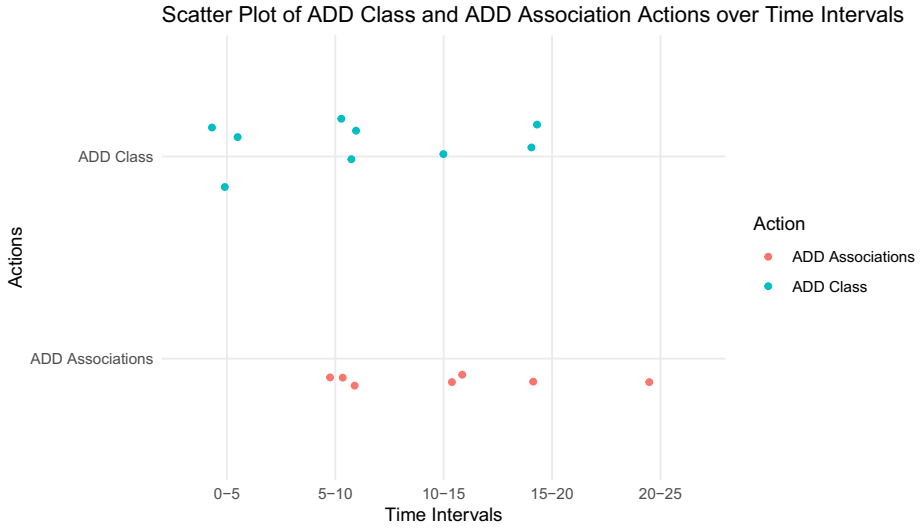
By combining the audio with these actions, we observed that students visualised the problem and mapped to its solution in a way that made constructing classes and their relationships appear disorganised. This might be influenced by their experience and familiarity with the diagram style. However, while these factors may play a role, the reason is also related to their personal style. A notable piece of evidence is the difference between the two datasets from the two universities. At UMA, the students were in their third year, while at RU, the students were in their first year. We observed the third approach to drawing classes in four out of the five pairs at RU, which might be due to their inexperience. At UMA, the distribution of the three approaches was almost equal. Hence, we can assume this variation is due to personal style, with individual preferences for moving back and forth between problem and solution, rather than completing a class fully. Instead, they try to complete different pieces of the whole problem and establish the solution in that way.

Students adopt *two* distinct preferences when connecting classes in their class diagrams. The *first* preference involves drawing all classes initially and establishing connections later, while the *second* preference entails sequentially drawing classes and their associations. Variations in the second preference include drawing two classes before connecting them and drawing one class, adding an association, drawing a second class, and then connecting them.

Notably, all five pairs using PlantUML, a text-based modelling tool, follow the first preference. In contrast, all 11 pairs using MagicDraw, a more visually-oriented tool, favour the second preference. Hence, it seems the tool significantly influences these choices. PlantUML's code-based nature encourages the construction of larger blocks (classes) first for efficiency, as students can view the generated model after running the code. In contrast, MagicDraw's graphical interface provides visual feedback from the start, with associations drawn by dragging arrows or lines, offering a better visual experience compared to PlantUML. Figure 3 shows two different graphs showing two behaviours seen in the two modelling tools. In the figures, the x-axis represents time, while the y-axis shows the actions taken during the modelling of the class diagram. To simplify and enhance understanding, we have selected two key actions: **Add Class** and **Add Association**. In both figures, the blue dots



(a) Connecting classes in PlantUML



(b) Connecting classes in MagicDraw

Fig. 3 Different ways to connect classes with in PlantUML and MagicDraw

represent the combined number of class additions from two pairs, while the red dots indicate the combined number of added associations for two pairs from each university. In Fig. 3a, we can see two actions, Add Class and Add Association, from two pairs of RU students. The figure clearly shows how the Add Class actions happened initially, followed by the Add Association action. In Fig. 3b, we see a different picture; with the same two actions and two pairs from UMA, we can see the actions are intertwined.

4.1.2 Modelling Preferences for Sequence Diagram

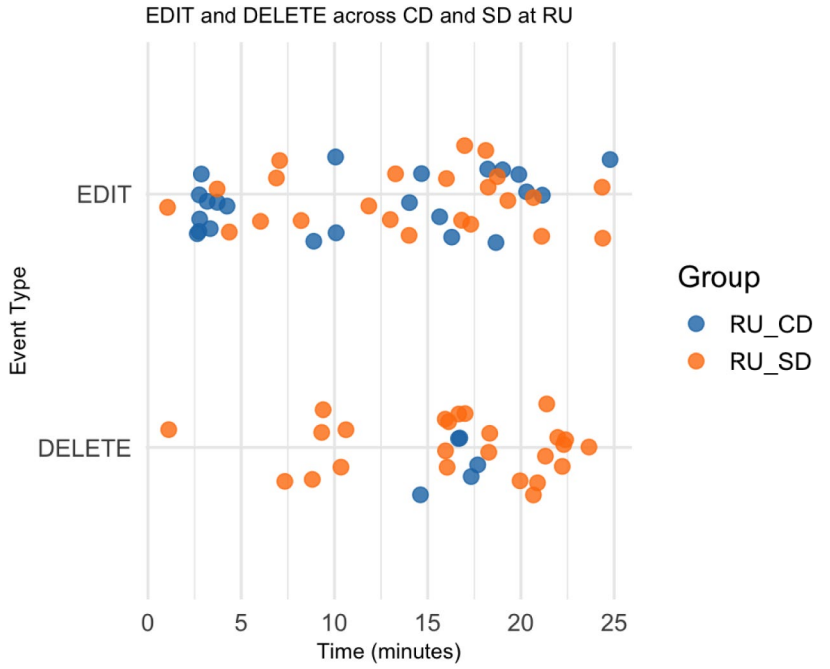
In contrast to class diagrams, students often modified their sequence diagrams. Hence, the emphasis was more on the Delete and Edit actions. Students exhibited a higher frequency of Delete and Edit actions when working on sequence diagrams, a trend consistently observed across both tools. Figure 4 illustrates two figures of students modifying both class and sequence diagrams across both tools. Figure 4a and bdemonstrate the frequency of Edit and Delete actions performed by RU students in PlantUML and UMA students in MagicDraw, respectively. The X-axis denotes time intervals and Y-axis represents the two different actions. In Fig. 4a, we could see that in PlantUML the difference between the two diagrams is more prominent, as in class diagram students rarely used Delete action. For students using MagicDraw, the Edit action was performed almost equally for both diagrams, but Deleete action was more frequent for sequence, same as PlantUML.

After analysing the videos, we concluded the following purposes for using a higher amount of Delete actions in sequence diagrams:

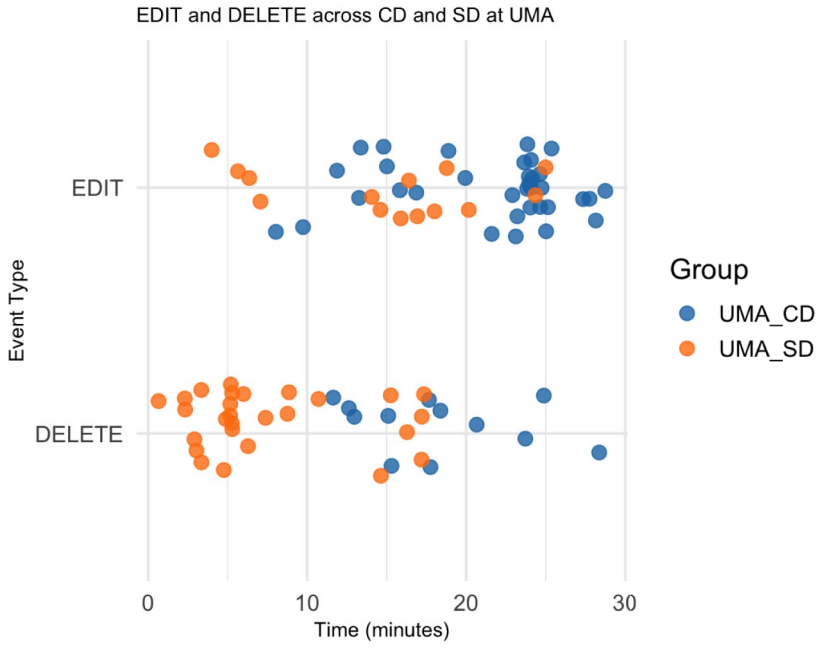
1. Communication Issues: students were modelling in pairs and following an approach similar to pair programming. We found the *driver* deleting some elements as there was a misunderstanding or miscommunication between the *driver* and the *navigator*.
2. Wrong Syntax/Tool issues: the *driver* was realising syntax mistakes and hence deleting an element, heavily influenced by the tool's interface. When utilising PlantUML, we noted a frequent reliance on the "copy-paste" function by students due to its text-based nature. This increased use of the "copy-paste" function consequently resulted in a higher frequency of Delete and Edit actions during the modelling process, as pasted content had to be adapted.
3. Brainstorming: Students deleted elements from the diagram while brainstorming the solution. Students would Add a tentative element, e.g., a class name or an association, to have a basis for discussions. These concepts would then frequently be edited or deleted again while the discussion was ongoing.

While reviewing the videos, we encountered situations where Delete and Edit actions were back-to-back, necessitating a clear distinction between these two actions. It is important to note that, in class diagrams, we did not observe this, indicating that students are more confident about modelling the domain with class diagrams.

We observed inconsistent preferences or strategies across the sequence diagrams, with students employing varied message types throughout their models. We believe this happened because they did not have a clear method, like the "noun identification" method, to follow. In Fig. 5, we can see two sequence diagrams for creating a profile in the dating app with appropriate information and two different diagrams with different strategies. In Fig. 5a, students used three objects, i.e., *Account*, *Profile* and *Verify*, to demonstrate profile creation with verification in the dating app. For the same problem, the second pair used two objects *DatingApp* and *BackEnd* in Fig. 5b. Note that both pairs used different strategies to fulfil the requirements, like feeding and verifying information into the profile. Additionally, while the first pair used one direct message line to feed all the personal information into the profile, the second used separate message lines. The problem description listed all the personal

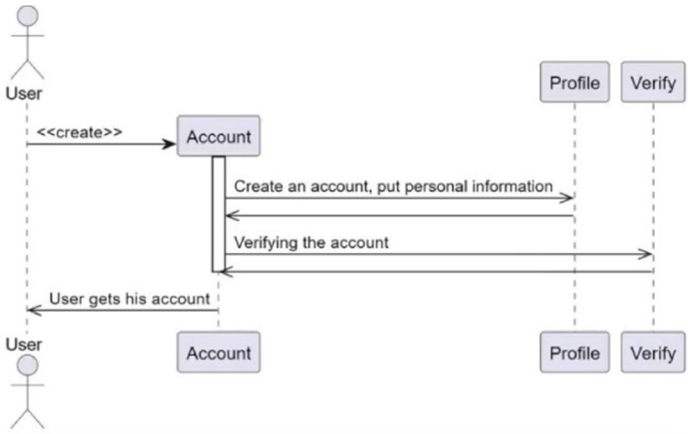


(a) Edits and Deletes in PlantUML (RU)

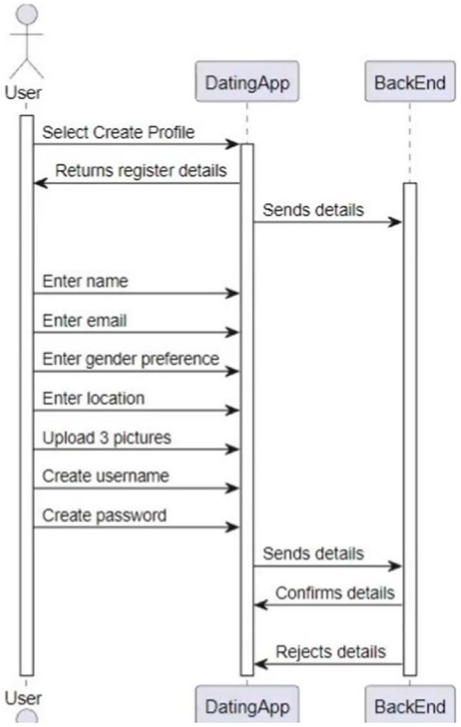


(b) Edits and Deletes in MagicDraw (UMA)

Fig. 4 Modifying the class vs sequence diagram across both tools



(a) Pair using one direct message line to feed information into the profile



(b) Pair using several direct message lines to feed information into the profile

Fig. 5 Sequence Diagrams from RU with different strategies for profile creation

information needed to create a profile. Notice the difference between the two pairs and how they read the problem.

Overall, we observed a key foundational difference between class and sequence diagrams. Students tended to be more compact and structured when working on class diagrams,

whereas their actions were more scattered and less systematic during sequence diagram modeling. In the next sections we discuss more on this as challenges.

Tables 2 and 3 present co-occurrence matrices of modeling actions during the creation of class and sequence diagrams. Both tables illustrate the actions of one representative pair (RU_Pair1) working with PlantUML. Each cell in the matrix reports how many times two different types of actions occurred within a short time window (60 seconds), suggesting that they were part of the same modeling context or cognitive activity. For instance, the high co-occurrence count between ADD and EDIT in Table 2 shows that participants frequently edited recently added elements, reflecting an iterative refinement process. The rows and columns correspond to categories of modeling actions: ADD CLASS, ADD ASSOC (association), EDIT, and DEL (delete). Diagonal cells represent repeated occurrences of the same action type in close succession (e.g., adding multiple classes in a row), while off-diagonal cells highlight how different types of actions were intertwined.

These tables shed light on participants' modeling behavior. For example, a strong co-occurrence between ADD ASSOC and EDIT in Table 2 suggests that students often added associations and then immediately refined their properties (such as cardinalities). Conversely, the low co-occurrence between ADD CLASS and DEL indicates that once classes were created, they were rarely removed, reflecting relatively stable decisions about the class structure. Similarly, in Table 3, the strong co-occurrence between ADD MESSAGE and EDIT highlights students' uncertainty and frequent adjustments of message terminology while working on sequence diagrams. Table 3 also shows a more diverse mix of co-occurrences compared to class diagrams. While the tables here only illustrate the behavior of a single student pair for readability, the analysis of all 16 pairs revealed consistent patterns. The full datasets are available, and additional tables can be provided upon request.

4.2 RQ2: What Challenges do Students Face While Solving Modelling Tasks?

To answer RQ2, we integrated the modelling actions with the interactions between students while solving the modelling problem. Through analysis of 16 student pairs and their recorded interactions, we identified recurring challenges in modelling tasks by examining statements that referenced specific difficulties. These were consolidated into three main categories, as shown in Table 4. The table organizes these findings into five columns: (1) the primary challenge categories, (2) unique codes identifying specific aspects of each challenge, (3) descriptions of these codes, (4) representative student quotes exemplifying the codes, and (5) the source identifiers in the format (Pair_ID, Diagram), where each pair is labeled by their university affiliation and number (e.g., UMA_Pair3), and the diagram type indicates the modeling context of the statement.

The main three challenges and its respective aspects are described below with two types of data: performed actions and recorded interactions.

4.2.1 Inconsistent Strategies Across Different Diagrams

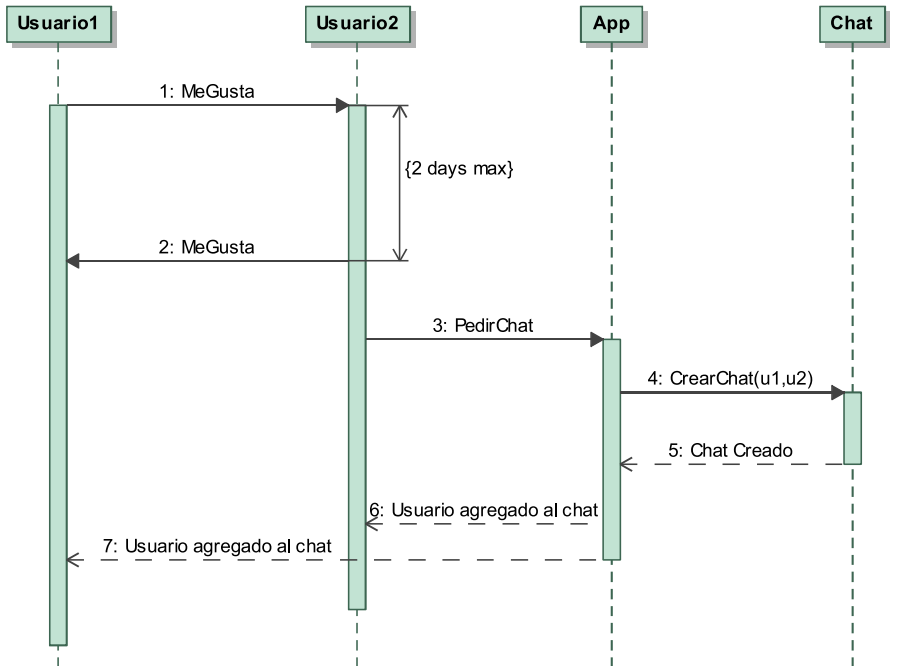
One of the challenges we observed is the use of inconsistent strategies while solving the modelling problem. In Table 4, we present the codes *strategies* and *error*, which capture two key aspects of students' interactions related to this challenge. The *strategies* code reflects instances where students discussed general difficulties or uncertainties without taking any

Table 4 Codes and representative examples of modeling challenges

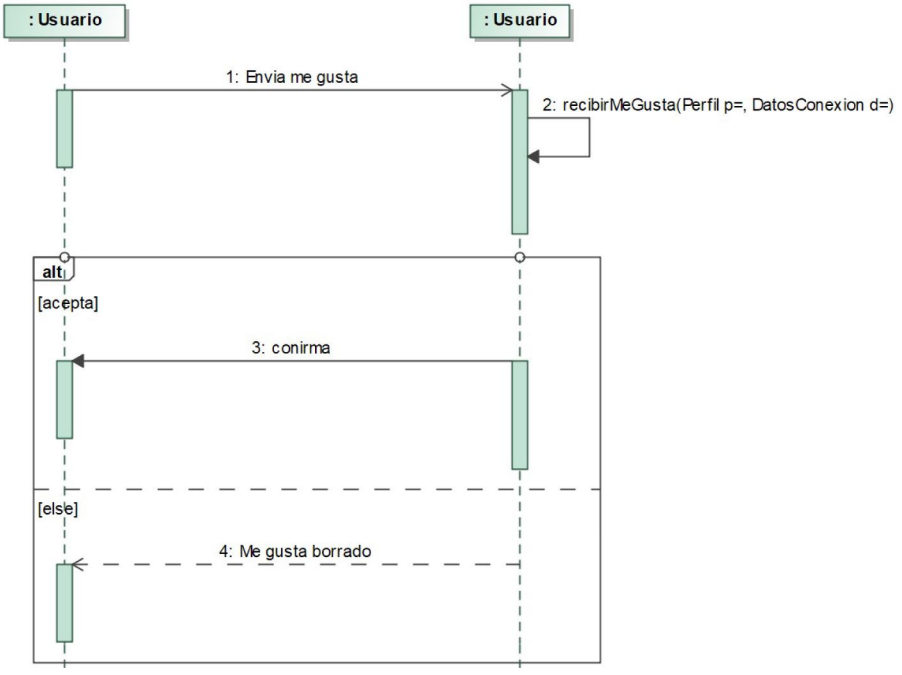
Challenge	Code	Description	Example Statement	Source (Pair_ID, Diagram)
Interpreting Problem	refer_problem	Students refer explicitly to the problem description or task prompt when unsure about modeling content	"But how can I move that though? Like maybe we have to deactivate. Should read first."	(RU_Pair3, SD)
	refer_diagram	Students refer to a particular diagram or diagram element when unsure of the modeling content	"Can we change the class diagram? I don't think so."	(UMA_Pair4, SD)
	refer_diagram	Students refer to missing or inconsistent diagram elements	'This attribute is not in the class diagram. Should we add it?"	(UMA_Pair3, SD)
Inconsistent Strategies	strategies	Students refer to general difficulties of drawing a particular diagram	"I think we need to draw the activation line before sending the message to dating app, but then how to stop the message thread?"	(RU_Pair4, SD)
	errors	Students perform wrong actions to draw a particular diagram	"No, this is not right. But like it makes sense to delete it because, like, if if they don't answer them, the message thread gets deleted. "	(UMA_Pair8, SD)
Completeness of Diagram	completion	Students express doubt about completing a diagram	"Whatever. We are done, I hope! We just have to fix this later, maybe."	(UMA_Pair3, SD)
	completion	Student expresses doubt about completeness or correctness	"I don't know if this will work...Better than expected."	(RU_Pair1, SD)

on-screen actions. In contrast, the *error* code refers to incorrect actions performed by students while working on a diagram. This issue was less evident in class diagrams. Although we observed different strategies for creating a class diagram, they were not particularly disorganised. However, in the sequence diagrams, students used varying terms to describe messages and objects. We could see an evidence of this challenge through students actions in Fig. 4b. Students using MagicDraw edited their class diagram more often towards the end of their session and deleted from their sequence diagram towards the beginning of their session. The higher amount of Deletes in the beginning of the diagram shows that students were less confident with sequence diagram.

One reason behind this inconsistency is how students interpret the modelling problem. In our previous studies (Chakraborty and Liebel 2023, 2024), we noted that there is no systematic method taught to students for practising the reading of a modelling problem, which is a significant issue. The consequence of this challenge is that students' understanding of the correct diagram can be misleading. As seen in Fig. 5a and b, two different diagrams from RU show the same functionality of creating a profile in the app, but each of them employs a different design. A similar situation happens in the diagrams in Fig. 6 from UMA, where we can see that different students go into different levels of detail in the diagrams. We understand that, since modelling is a creative task, having different solutions is to be expected.



(a) Pair considering the like and the creation of the chat



(b) Pair only considering the likes

Fig. 6 Sequence Diagrams from UMA about giving a "like" to someone

However, guidance is needed to explain to students why they are modelling in different ways.

4.2.2 Challenges in Interpreting a Modelling Problem

Our study setting allowed students to read the problem for five minutes before beginning the modelling task. Although students had both tasks (class and sequence diagrams) from the start, the majority focused on reading the problem to create the class diagram first. We observed *refer_problem* and *refer_diagram* two codes where students expressed problem interpreting the modeling task at hand and willingness to go back to either read the problem description or read the class diagram. The later part is interesting as a common aspect we observed is that students preferred reading the problem description to solve the class diagram and then referred to the class diagram to solve the sequence diagram. Consequently, if they missed something in the class diagram, the subsequent sequence diagram was affected. Students realised the importance of reading the problem thoroughly to solve both tasks only later, while finishing the sequence diagram, and they acknowledged their oversight. Figure 7 illustrates how frequently participants referred back to the problem statement while working on the class diagram, and how often they referred to the class diagram while constructing the sequence diagram. This figure highlights a significant challenge in interpreting problem requirements, as it suggests that students often need to revisit earlier artifacts to clarify their understanding, particularly when transitioning between different modeling tasks.

Figure 7 is a summary of what we recorded through each pairs interaction while modelling. Following are some of the examples:

> "N: Let's not do that. We should have fixed that in class diagram, I think it was mentioned in the problem"

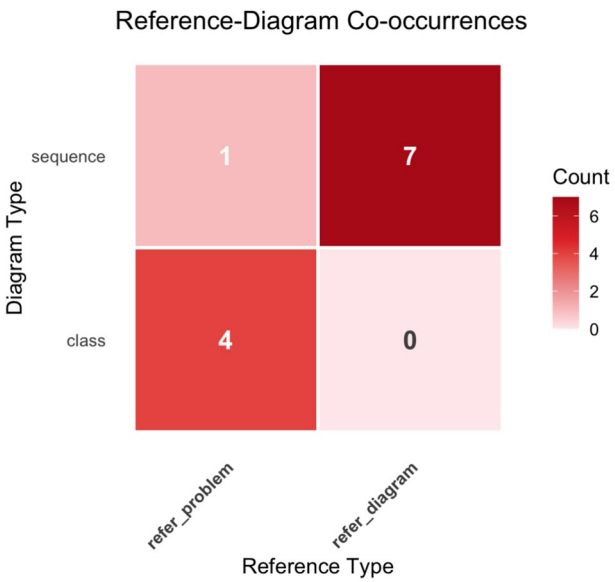


Fig. 7 Occurrences of referring to problem vs class diagram during modeling

RU_Student[8]"*D: But how can I move that though? Like maybe we have to deactivate. N: Should read first*"

RU_Students

"*D: Can we change the class diagram at this stage (while realising the sequence diagram)? N: I don't think so*"

UMA_Students

"*D: We should have spent more efforts when depicting the class diagram. N: I agree*"

UMA_Students

"*D: I think the thread should have been a class. N: I agree, but that's already done, now we cannot leave it like this in the sequence diagram.*"

UMA_Students

Following a conversation piece between a pair while solving a sequence diagram task where students had to model the requirements, *a user can "like" another user if they share the same geographic location, a user has two days to confirm or decline a "like" from another user"*. This piece of conversation shows struggles and lack of systematic approach to read a modelling problem/model.

"N: It's not like this, because this arrow indicates User1. So maybe we have to put another arrow from User1 to User2 or something. D: No, no, it doesn't matter. N: Wait, what if both share the same? Oh, no, no. I was reading this wrong. D: So okay, so we get a notification we can skip the arrow that goes back to user one, right? N: Oh no no, they don't share the same location type"

RU_Students

The challenge here highlights students' lack of training in reading problems and diagrams. Having an unsure attempt while drawing such a diagram is understandable. However, a structured way of reading a problem needs to be added here, furthering a method to create this model systematically. We can see students employing both approaches to solve a modelling task, but due to insufficient guidance, they struggle with each.

4.2.3 Lack of Knowledge in the Completeness of a Diagram

Students commonly face uncertainty regarding task completion, experiencing doubt after both class and sequence diagram assignments. The challenge lies in the difficulty students encounter in verifying the accuracy of their modelling steps, leading to uncertainty about whether the diagrams effectively address the given tasks. The absence of a clear validation mechanism during the drawing process contributes to this post-drawing doubt among students.

[8] *D* and *N* refer to *Driver* and *Navigator*, respectively

"*D: Whatever. We are done, I hope! We just have to fix this later, maybe. N: I think, this is alright(?)*"

RU_Student

"*D: I don't think there is much more we can do.*"

UMA_Student

"*D: There's only 50 seconds left. N: Yeah, I'm not sure about the delete action, let's just leave it like this*"

UMA_Students

5 Discussion

In this section, we discuss our findings, particularly the issues identified through our results and their implications for modelling education.

5.1 Different Modelling Preferences

Stikkolorum et al. (2015) conducted a similar study with 20 first year bachelor's students in software engineering during their second semester.Unlike our study, these students were inexperienced with UML modelling and only created class diagrams having learned class diagramming prior to the task. The authors recorded four activities from their class diagram modellingcreate, move, set, and remove. They used their own tool, WebUML, to log these activities. The authors report four strategies for modelling class diagrams in Stikkolorum et al. (2015). We found similar preferences for class diagrams among our students. However, our study, supported by students' conversations and the use of two different tools, provides a more in-depth picture behind these strategies. Three factors influence students' preferences: the modelling tool, the way they read the problem, and their personal style. We observed two main preferences for connecting different classes in the diagram. The first preference involves creating all classes initially and establishing connections later, while the second preference entails sequentially adding classes and their associations. Notably, all five pairs using PlantUML, a text-based modelling tool, follow the first preference. In contrast, all eleven pairs using MagicDraw, a more visually-oriented tool, favour the second preference. This strongly suggests that the tool has a significant influence on these choices. Our study also involves sequence diagrams, which helped us notice different preferences among students based on how they read the problem or the class diagram prior to modelling the sequence diagram. Additionally, including sequence diagrams shows the difficulties students have in connecting multiple UML diagrams, e.g., by using class diagram elements in their sequence diagrams. Finally, since we employed a method similar to pair programming, where students alternated their roles, we observed variations in personal style. Some students favoured a more compact layout for the sequence diagram, while many followed the exact wording provided in the problem description, and others chose their own vocabu-

lary. Unlike the class diagram, there was no noun identification method for the sequence diagram. In Fig. 8, we can see two different sequence diagrams for creating a profile in the app, modeled by students from UMA using MagicDraw. The pair in Fig. 8a mostly writes full sentences for the messages, while the pair in Fig. 8b tries to follow the camel case notation and write what could be operation names—although such operations were not added in the class diagram.

Consequently, individual thinking influenced certain preferences among the students using different tools. In PlantUML, copying and pasting was a common trend, and during brainstorming, we observed that students often paused their actions. In contrast, students using MagicDraw frequently utilised many edit and delete actions, playing around with the interface by dragging parts of the diagram, especially in the sequence diagrams. An interesting question for future work is whether any of these approaches is more or less suitable for various modelling tasks, such as design exploration.

Our study explores whether we observe differences in modelling styles among students. However, the study design does not allow us to reason about the impact on various qualities, e.g., whether one modelling style leads to better designs, is more maintainable, or is easier to understand than other observed styles. This is a limitation in the study design in favour of broad exploration of these styles. Hence, future studies need to investigate suitability for various tasks and precise measurement of the resulting modelling quality.

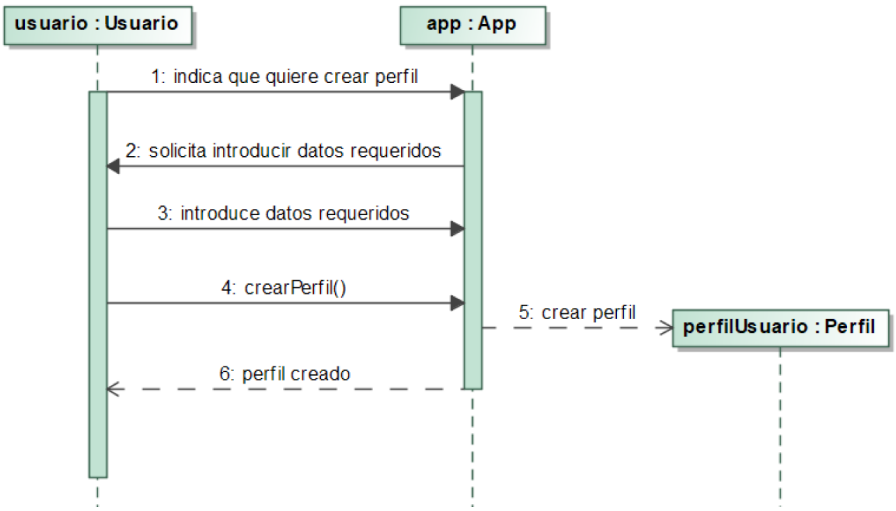
5.2 Structural vs Behavioural Modelling

Students find class diagrams relatively easy to model. Two primary reasons may explain this: the problem description explicitly stated "model the domain with a class diagram", which is less complex than modelling two functionalities with a sequence diagram. Another significant reason is that students find class diagrams more relevant because they can better visualise the diagram and its various relationships to programming. For instance, at RU, the course instructor demonstrated this relevance using programming in lectures. Students often struggle to understand and model the behaviour of the system, finding it more challenging than creating the structural components. The class diagram provides the basic building blocks, which the sequence diagram then has to adhere to. In theory, one could start by designing all the interactions first, making the subsequent class diagram more difficult. However, in practice, students do not start with a "clean sheet" when working on sequence diagrams; they frequently refer to the class diagram to construct the sequence diagram.

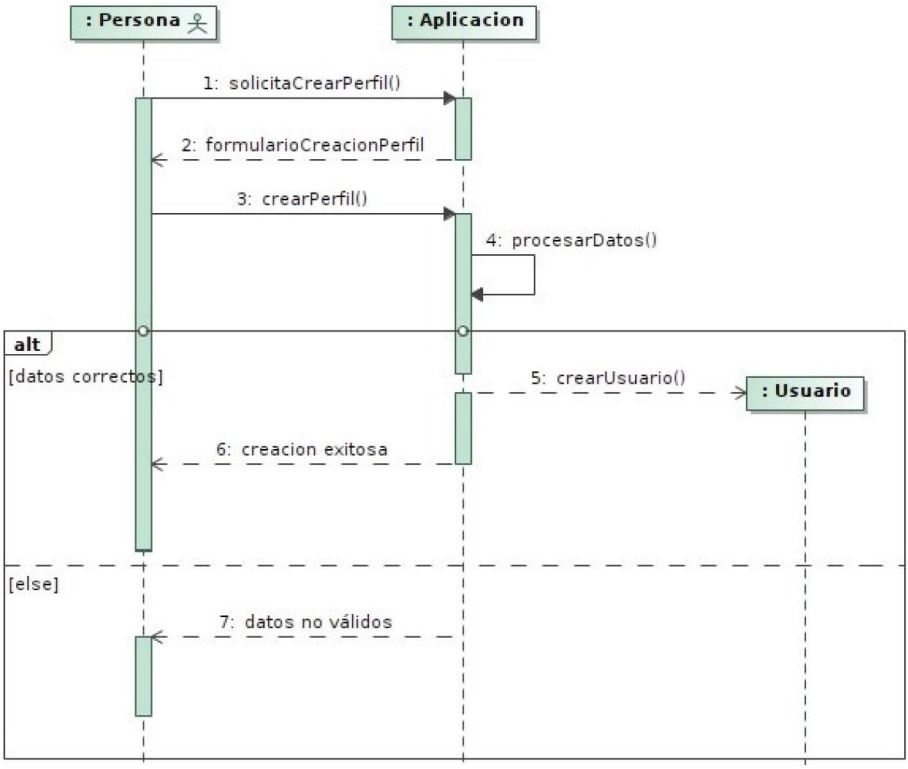
Various studies (Reuter et al. 2020; Lopes et al. 2019) have addressed challenges with UML diagrams, particularly highlighting the difficulties associated with sequence diagrams. One study mentioned that students find it hard to imagine the sequence of events. In our previous research (Chakraborty and Liebel 2023, 2024), students reported that sequence diagrams were challenging and the sense of a complete diagram was vague. We observed a similar trend in this study; conversations indicated that students felt more confident about finishing the class diagram than the sequence diagram.

5.3 Influence of Modelling Tools

Tool-related challenges are not new in software modelling education (Liebel et al. 2016; Reuter et al. 2020). The various designs and interfaces of modelling tools have created com-



(a) Pair writing mostly sentences in messages



(b) Pair trying to use camel case notation for messages

Fig. 8 Sequence Diagrams for creating a profile in the app (UMA)

plexities in learning modelling. Pourali and Atlee (2018) show several challenges modellers faced while modelling class and state machine diagrams with eight different modelling tools. However, studies like (Bimonte et al. 2013; Paige et al. 2014; Robbins and Redmiles 2000) have investigated and explored different ways a tool can assist modellers; although, these are industry-level studies and do not focus on training students in modelling education. Our study supports modelling tool's influence on modelling style, or *preferences*. In our study, two different modelling tools were involved: MagicDraw and PlantUML. While MagicDraw has a more "drag and draw" functionality, PlanUML is code-based. The distinct nature of these two tools influenced our students to follow a certain individual preferences during modelling. While creating a class diagram, students using PlantUML created all the classes first and then connected them with the necessary associations. The reason is that, in PlantUML, one needs to run the code in order to get the visual product, and students found it helpful to see all the classes on the screen before completing the connection between them. In MagicDraw, the tool interface allows one to create diagrams visually and graphically by selecting elements from the palette and dragging them to the main panel. Since students could visualise the model in the making, students chose to complete a small part of the class diagram by creating connections between two classes and then moving forward.

In sequence diagrams, we observed more modification frequency than in class diagrams, i.e., more Delete and Edit actions. The tool's interface influenced here, too. Since PlantUML is code-based, students frequently used "copy-paste" and, hence, used Edit actions. Not having consistent terminologies in sequence diagrams and the code-based nature of PlantUML made students more prone to errors. Comparatively, with MagicDraw, there were more Delete actions, as deleting an element or a thread of messages was easy with a click.

It is well established that the choice of modelling tool influences the modelling process, shaping both the approach and the final outcome. While granting students the freedom to choose their own tools is important for fostering independent learning, this flexibility should be accompanied by proper guidance. Even experienced modellers can struggle with industry-level tools, raising concerns about how students, who often receive little structured exposure, can be expected to adapt seamlessly in professional settings. If we do not guide students in exploring different tools and understanding how tools influence modelling styles, we risk leaving them unprepared for real-world challenges.

5.4 Implication to Education

Based on our findings, we emphasise the importance of training students to interpret a modelling problem and a model, to understand the correctness of a model, and to use modelling tools to foster creativity and an individual style. Before teaching modelling, it is crucial to incorporate instruction on interpreting both a modelling problem and a model. This practice does not only provide students with a foundational understanding of the modelling concept, but also instills a sense of correctness in their approach. Training students to read a modelling problem helps them gather the necessary information and map a mental model based on the problem and chosen diagram type. Studies in the past have demonstrated the difficulties of interpreting models. Studies like Kuzniarz et al. (2004) and Zayan et al. (2014) explore the involvement of stereotypical examples to enhance the comprehension of UML models. Kutar et al. (2002) investigate whether users' understanding of information in sequence diagrams differs from their understanding of collaboration diagrams. We have observed

students reading the class diagram to model the sequence diagram, highlighting the necessity of an individual's ability to comprehend a model. When students read a model, they can understand the reverse journey from creation to the problem, identifying mistakes reflected in the model. This knowledge becomes invaluable when students create their own models. Additionally, reading a model imparts valuable insights into the visual aspects of modelling, helping students understand what the end-user can and should infer from the model. Consequently, students develop a nuanced comprehension of the information conveyed through the model, fostering a more comprehensive and insightful modelling process.

Unlike programming, modelling does not provide immediate error feedback. To address this, students should understand when their model is complete and correct, understanding by correct that the resulting model fulfills all the criteria mentioned in the problem description A potential approach is to teach students how to interpret a modelling problem into specific modelling concerns (Liebel 2016). Figure 9 shows five modelling concerns that should be considered while modelling. In the early stages, understanding the objective and purpose is essential. These concerns also assist in interpreting an existing model. In this study, we focused on the purpose and objective of modelling. Students were provided with a problem description and a list of requirements. We observed their modelling actions based on their understanding of the modelling task's purpose and objectives. As students selected their own modelling tools, we did not initially account for any tool-related effects. However, one of our key findings was the significant influence of tools on students' modelling preferences. While previous studies have highlighted how tools contribute to either challenges or strengths, our study reveals that a tool can profoundly shape a modeller's thought process, potentially leading to the development of a permanent modelling style. Although the notation was standardised in our study, we did not incorporate stakeholders, which we plan to address in future research. For educators, we suggest that these modelling concerns need to be explicit, or exploring the concerns needs to be part of the task, as the solution space otherwise grows.

A key limitation of using students with complex UML diagrams lies in the constrained environment of classroom settings, where the design and flexibility of experiments are often restricted. Similarly, in their study (Planas and Cabot 2019), Planas et al. found no signifi-

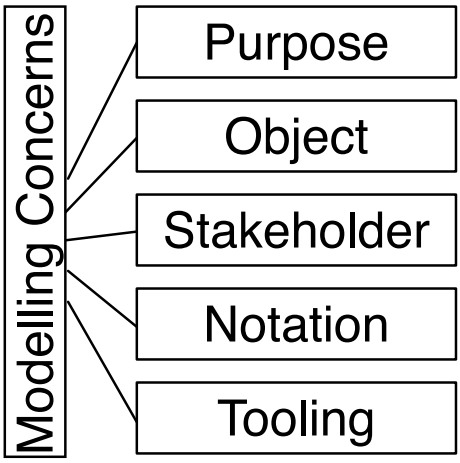


Fig. 9 Classification of Modelling Concerns. Adapted from Liebel (2016)

cant differences in global modelling strategies among students when creating class diagrams. The authors attributed this behaviour partly to the unrestricted approach of modelling tools, which allow users to create diagrams freely. One of the critical revelations from our results is that students' approaches to modelling vary based on the tools they use. Modelling tools are often considered challenging by students (Nóbrega et al. 2007; Pourali and Atlee 2018). However, beyond this challenge, modelling tools also shape students' modelling actions. Studies like (Bimonte et al. 2013; Pati et al. 2017; Robbins and Redmiles 2000) have investigated and explored different ways a tool can assist modellers; however, these are industry-level studies and do not focus on training students in modelling education. Additionally, the selection of a modelling tool is often left to the students. We believe it is beneficial to let students choose their own modelling tool, but we strongly recommend demonstrating the different interfaces of several modelling tools and allowing students to explore these tools with different assignments. In our experiment involving both sequence and class diagrams, we observed that students exhibited greater confidence with class diagrams, as reflected by fewer Edit and Delete actions. We have previously noted that students find structural diagrams easier to work with compared to behavioural diagrams. This may be because, in a classroom setting, students do not have the opportunity to observe the implemented behaviour of a system, making it challenging for them to confidently model behavioural diagrams, such as sequence diagrams. Educators should take note of this, as our recommendation to focus on reading models can be particularly helpful. By studying how system behaviours are modeled before attempting to create behavioural diagrams themselves, students can gain a clearer understanding and build confidence in modelling such diagrams.

We further investigated by creating a set of modelling recommendations and applying them in two 12-week bachelor courses related to software design (Chakraborty and Liebel 2024). Feedback was gathered from students (via surveys) and instructors (through interviews) regarding the impact of the applied recommendations. To assess students' actual understanding of modelling following the implementation of these recommendations, we analysed both their assignments and their feedback. Additionally, a focus group study with students, teaching assistants, and experts from modelling and education was conducted to obtain early feedback on the overall impact of the recommendations. Students and teaching assistants reported improved communication, and students indicated that the recommendations helped clarify assignment expectations, allowing them to manage their workload and learning pace more effectively. Our study establishes robust guidelines for modelling, which we believe will have a greater impact on education and better prepare students for industry roles.

6 Conclusion

Our study offers insights into students' actions and interactions while solving a modelling problem.

We conducted a modelling study with 32 students from two universities and countries. The study involved solving a modelling task that included two types of UML diagrams: class and sequence. Students participated in the study in pairs, and we recorded their screen activities and conversations. After analysis, we observed several modelling preferences and challenges among the participants. These observations revealed modelling preferences that

could evolve into distinct modelling styles with more detailed and broader investigation. By examining students' modelling actions alongside their descriptions of the modelling experience, we gained a deeper understanding of modelling as a creative task and identified several issues in modelling education that hinder its later adoption. These issues include inconsistent strategies, a lack of training in interpreting modelling problems, and the absence of methods to verify one's final model. We strongly recommend incorporating guidance into modelling education for interpreting modelling problems and creating models to address these issues effectively. The modelling preferences found through our study are promising and encourage further studies with different diagrams and a diverse range of problems. The results from these studies, in addition to the findings presented here, can lay the groundwork for identifying individual modelling styles, thereby guiding individuals towards more informative modelling practices. While our study provides valuable insights, we acknowledge certain limitations, such as the relatively small dataset and the lack of characterisation of participants' prior modelling experience. In the future, we aim to include a larger and more diverse participant pool to validate our findings and explore the generalisability of modelling preferences across different contexts. To avoid selection bias and improve generalisability, we also plan to perform further studies following different strategies for participant selection such as stratified random sampling. Expanding the study to include more complex modelling tasks and additional UML diagrams will further enrich our understanding of how modelling styles evolve and how best to support students in developing effective modelling practices.

Appendix A Problem Descriptions

We present three problem descriptions used throughout our study setup. We changed the details and timings after two pilots. The main study was for 1 hour, divided into Reading the full problem description (5 minutes), one of the participants solving task 1 with a class diagram (25 minutes), taking a small break (5 minutes), the other participant solving task 2 with sequence diagrams (25 minutes).

A.1 Pilot Study 1

– **Problem Description**: The task is to model the domain of a restaurant food court system using class diagram. The system should facilitate the management of multiple restaurants within a food court, customer orders, and the preparation and delivery of food. Your model should accurately represent the relationships between the entities involved and provide a clear structure for the system's operations. Please consider the following requirements for the system: The system must support multiple restaurants within the food court, each restaurant should have details such as name, cuisine type, menu items, and operating hours, an order should include customer details, a list of ordered items, total price, and order status (e.g., pending, preparing, completed) and finally, the system should support the delivery of orders to customers, including delivery time and delivery personnel details.

- **Modelling Tool**: Please use the Eclipse Papyrus for the task.
- **Time**: 30 minutes

A.2 Pilot Study 2

- **Problem Description** The task involves modelling two problems related to an international dating app. Some information of the system: to create a profile, users must provide their personal information, location, and at least three pictures. Every user should have a valid credentials. The app operates by allowing users to see others within a certain geographic area, defined as a flexible distance (x km) from the user's location, which can be either their current GPS location or a manually set coordinate. To initiate a connection, a user can send a "like" to another user within the same geographic area. The recipient will receive a notification with the sender's profile information and has two days to confirm the like. If there is no confirmation within two days, the like is deleted. The first task is to model the domain using class diagram. The second task is to model two behaviours of the system: user profile creation and initiating connection between two users.
- **Modelling Tool**: Please use a modelling tool of your choice.
- **Switching Roles**: After a brief description of pair programming, students were instructed here to switch roles between tasks.
- **Time Limit**: Total 1 hour. Read the problem: 10 minutes, class Diagram: 20 minutes, small break: 10 minutes and sequence Diagram: 20 minutes.

A.3 Main Study

- **Problem Description**:Your job here is to solve two tasks regarding a dating app. You must maintain the following constraints while solving the tasks: 1. It is an international dating app, not country specific. 2. To open a profile, the user must provide their name, email id, gender preference, location and at least 3 pictures. Every user should have a username and password. 3. How does this app work? Assume User1 and User2 are both users of this app from specific geographic areas. Two things to notice here: "geographic area" and "user's location". A "geographic area" is a certain distance (x km) from the user's location, which is flexible. A "user's location" can be either that user's current GPS location or some other coordinate set by the user manually. For example, if User1's location is Reykjavik, User1 can set his/her/their geographic location within 80 km of Reykjavik. 4. User1 can see all other users of this app with the same geographic location. 5. To initiate a connection an interested user of this app send a "like". User1 can like User2 if both share the same geographic location. User2 will get a notification, including User1's profile information. User2, in this case, has two days to confirm User1's like; a confirmation means User2 also like User1. If there is no confirmation, after two days User1's like will be deleted. 6. If both User1 and User2 send like to each

other, the app initiates a message thread private to both users. If none of the users starts talking after 1 week, the thread will be deleted, including both users' like.
- **Task 1: Class Diagram** Your task is to model the domain of this app, that is, design the domain model using a class diagram. Please mention the attributes of each class and the multiplicities. Try to think about the operations for each class and mention as many as you can. The purpose here is for a pair to decide what should be in the system and what's outside of the system.
- **Task 2: Sequence Diagram** Your task is to model the sequence of the following functionalities: 1. User profile creation: Show step-by-step how a user creates a profile in the app. Try to be as detailed as possible. 2. Users initiate the connection: Now that users have created their profiles, show step-by-step how a connection is made between two users. (Use the description given above to understand the communication between the system and its users, to draw the sequence diagram)
- **Modelling Tool**: Please use a modelling tool of your choice.
- **Switching Roles**: After a brief description of pair programming, students were instructed here to switch roles between tasks.
- **Time Limit**: Total 1 hour. Read the problem: 5 minutes, class Diagram: 25 minutes, small break: 5 minutes and sequence Diagram: 20 minutes.

Acknowledgements We would like to thank all the students who participated in the study.

Author Contributions Shalini Chakraborty and Grischa Liebel (*Department of Computer Science, Reykjavik University, Reykjavik, Iceland*), and Javier Troya and Lola Burgueño (*ITIS Software, Universidad de Málaga, Malaga, Spain*). Correspondence to Shalini Chakraborty.

Funding Open Access funding enabled and organized by Projekt DEAL. This project has been funded by the Spanish Ministry of Science, Innovation, and Universities under contract PID2021-125527NB-I00 and University of Málaga under project JA.B1-17.

Data Availability We publish the problem descriptions we used for pilot studies 1 and 2 and the main study in the Appendix. We didn't disclose the videos of students modelling for privacy reasons. Students modelling actions are provided through an external link: https://doi.org/10.5281/zenodo.14857703. The students names aren't disclosed thus each pair is identified as their university acronym-pair number.

Declarations

Competing Interests The authors have no competing interests to declare that are relevant to the content of this article.

Ethical Approval Our study involved voluntary participation from university students, who had the right to withdraw at any time. Participants were informed about the scope and rules of the study in the beginning.

Informed Consent Participants provided written informed consent, acknowledging the study's purpose, design, and participation rationale. We received signed consent form from participants prior to study.

Conflict of Interest The authors declare that they have no conflict of interest.

Clinical Trial Number Clinical trial number: not applicable.

References

Abbott RJ (1983) Program design by informal english descriptions. Commun ACM 26(11):882–894

Agner LT, Lethbridge TC, Soares IW (2019) Student experience with software modeling tools. Softw Syst Model 18(5):3025–3047

Agner LTW, Soares IW, Stadzisz PC, Simão JM (2013) A brazilian survey on UML and model-driven practices for embedded software development. J Syst Soft 86(4):997–1005. SI : Software Engineering in Brazil: Retrospective and Prospective Views

Akayama S, Demuth B, Lethbridge TC, Scholz M, Stevens P, Stikkolorum DR (2013) Tool use in software modelling education. In: EduSymp@ MoDELS

Andriessen J, Baker M, Suthers D (2003) Argumentation, computer support, and the educational context of confronting cognitions. Confronting cognitions in computer-supported collaborative learning environments, Arguing to learn, pp 1–25

Aniche M, Treude C, Zaidman A (2021) How developers engineer test cases: An observational study. IEEE Trans Software Eng 48(12):4925–4946

Baker MJ (1999) Argumentation and constructive interaction. Foundations of Argumentative Text Processing 5:179–202

Baltes S, Ralph P (2022) Sampling in software engineering research: A critical review and guidelines. Empir Softw Eng 27(4):94

Begel A, Nagappan N (2008) Pair programming: What's in it for me? New York, NY. Association for Computing Machinery, USA

Bimonte S, Boulil K, Pinet F, Kang M-A (2013) Design of complex spatio-multidimensional models with the icsolap uml profile-an implementation in magicdraw. In: International conference on enterprise information systems, 2:310–315. SCITEPRESS

Bruegge B, Dutoit AH (2009) Object-Oriented Software Engineering Using UML, Patterns, and Java. Prentice Hall Press, 3rd edition

Burgueño L, Vallecillo A, Gogolla M (2018) Teaching UML and OCL models and their validation to software engineering students: an experience report. Comput Sci Educ 28(1):23–41

Burgueño L, Vallecillo A, Gogolla M (2018b) Teaching UML and OCL models and their validation to software engineering students: an experience report. Compt Sci Educ, pp 1–19

Carver J, Shull F, Basili V (2003) Observational studies to accelerate process experience in classroom studies: An evaluation. In: 2003 International symposium on empirical software engineering, 2003. ISESE 2003. Proceedings., pages 72–79. IEEE

Chakraborty S, Liebel G (2023) We do not understand what it says-studying student perceptions of software modelling. Empir Softw Eng 28(6):149

Chakraborty S, Liebel G (2024) Evaluating software modelling recommendations: Towards systematic guidelines for modelling. In: Proceedings of the 18th ACM/IEEE international symposium on empirical software engineering and measurement, ESEM '24, page 337–347, New York, NY, USA. Association for Computing Machinery

Chakraborty S, Liebel G (2024) Modelling guidance in software engineering: a systematic literature review. Softw Syst Model 23(1):249–265

Ciccozzi F, Taentzer G, Vallecillo A, Wimmer M, Famelis M, Lambers L, Mosser S, Paige R, Pierantonio A, Rensink A, Salay R (2018) How do we teach modelling and model-driven engineering? a survey. In: Proceedings of the 21st ACM/IEEE international conference on model driven engineering languages and systems: Companion proceedings, pages 122–129

Dekel U, Herbsleb JD (2007) Notation and representation in collaborative object-oriented design: an observational study. ACM SIGPLAN Notices 42(10):261–280

Forward A, Badreddin O, Lethbridge TC (2010) Perceptions of software modeling: a survey of software practitioners. In: 5th workshop from code centric to model centric: evaluating the effectiveness of MDD (C2M: EEMDD)

Gilson F (2018) Teaching software language engineering and usability through students peer reviews. In: Proceedings of the 21st ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, pages 98–105

Gonnord L, Mosser S (2018) Practicing domain-specific languages: from code to models. In: Proceedings of the 21st ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, pages 106–113

Gorschek T, Tempero E, Angelis L (2014) On the use of software design models in software development practice: An empirical investigation. J Syst Softw 95:176–193

Hammouda I, Burden H, Heldal R, Chaudron MRV (2014) Case tools versus pencil and paper. In: ACM/IEEE 17th Int. Conf. on Model Driven Engineering Languages and Systems–Educators Symposium

Hutchinson J, Rouncefield M, Whittle J (2011) Model-driven engineering practices in industry. In: 2011 33rd International conference on software engineering (ICSE), pages 633–642

Hutchinson J, Whittle J, Rouncefield M, Kristoffersen S (2011) Empirical assessment of mde in industry. In: 2011 33rd International conference on software engineering (ICSE), pages 471–480

Kirstan S, Zimmermann J (2010) Evaluating costs and benefits of model-based development of embedded software systems in the car industry–results of a qualitative case study. In: Workshop C2M: EEMDD "From code centric to model centric: Evaluating the effectiveness of MDD"

Kolovos DS, Cabot J (2016) Towards a corpus of use-cases for model-driven engineering courses. In: EduSymp/OSS4MDE@ MoDELS pages 14–18

Kutar M, Britton C, Barker T (2002) A comparison of empirical study and cognitive dimensions analysis in the evaluation of uml diagrams. In: Procs of the 14th workshop of the psychology of programming interest group (PPIG 14), pages 1–14

Kuzniarz L, Staron M, Wohlin C (2004) An empirical study on using stereotypes to improve understanding of uml models. In: Proceedings. 12th IEEE international workshop on program comprehension, 2004., pages 14–23. IEEE

Liebel G (2016) Model-Based Requirements Engineering in the Automotive Industry: Challenges and Opportunities. Chalmers Tekniska Hogskola (Sweden)

Liebel G, Heldal R, Steghöfer J-P (2016) Impact of the use of industrial modelling tools on modelling education. In: 2016 IEEE 29th international conference on software engineering education and training (CSEET), pages 18–27

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. Softw Syst Model 17(1):91–113

Liebel G, Tichy M, Knauss E (2018b) Use, potential, and showstoppers of models in automotive requirements engineering. Softw Syst Modeling

Lopes A, Steinmacher I, Conte T (2019) UML acceptance: Analyzing the students' perception of UML diagrams. In: Proceedings of the XXXIII brazilian symposium on software engineering, pages 264–272

Mangano N, LaToza TD, Petre M, van der Hoek A (2014) How software designers interact with sketches at the whiteboard. IEEE Trans Software Eng 41(2):135–156

Miyake N (1986) Constructive interaction and the iterative process of understanding. Cogn Sci 10(2):151–177

Mohagheghi P, Gilani W, Stefanescu A, Fernandez MA, Nordmoen B, Fritzsche M (2013) Where does model-driven engineering help? experiences from three industrial cases. Softw Syst Model 12(3):619–639

Nóbrega L, Nunes NJ, Coelho H (2007) The meta sketch editor: a reflexive modeling editor. In: Computer-Aided Design Of User Interfaces V, pages 201–214. Springer

Paige RF, Polack FAC, Kolovos DS, Rose LM, Matragkas ND, Williams JR (2014) Bad modelling teaching practices. In: EduSymp@ MoDELS, pages 1–12

Pati T, Kolli S, Hill JH (2017) Proactive modeling: a new model intelligence technique. Software & Systems Modeling 16:499–521

Pinggera J, Soffer P, Fahland D, Weidlich M, Zugal S, Weber B, Reijers H, Mendling J (2013) Styles in business process modeling: an exploration and a model. Softw Syst Modeling, pages 1–26

Pinggera J, Soffer P, Zugal S, Weber B, Weidlich M, Fahland D, Reijers H, Mendling J (2012) Modeling styles in business process modeling. volume 113

Planas E, Cabot J (2019) How are UML class diagrams built in practice? a usability study of two UML tools: Magicdraw and papyrus. Computer Standards & Interfaces 67:103363

Pourali P, Atlee JM (2018) An empirical investigation to understand the difficulties and challenges of software modellers when using modelling tools. In: Proceedings of the 21th ACM/IEEE international conference on model driven engineering languages and systems, MODELS '18, page 224–234, New York, NY, USA. Association for Computing Machinery

Ragnarsson A, Chakraborty S, Liebel G (2021) Modrec: A tool to support empirical study design for papyrus and the eclipse modeling framework. In: 2021 ACM/IEEE international conference on model driven engineering languages and systems companion (MODELS-C), pages 361–369

Ralph P, Baltes S, Bianculli D, Dittrich Y, Felderer M, Feldt R, Filieri A, Furia CA, Graziotin D, He P, Hoda R, Juristo N, Kitchenham BA, Robbes R, Méndez D, Molleri J, Spinellis D, Staron M, Stol K-J, Tamburri DA, Torchiano M, Treude C, Turhan B, Vegas S (2020) ACM SIGSOFT empirical standards. CoRR arXiv:2010.03525

Reuter R, Stark T, Sedelmaier Y, Landes D, Mottok J, Wolff C (2020) Insights in students' problems during UML modeling. In: 2020 IEEE Global engineering education conference (EDUCON), pages 592–600. IEEE, 2020

Reuter R, Stark T, Sedelmaier Y, Landes D, Mottok J, Wolff C (2020) Insights in students' problems during UML modeling. In: 2020 IEEE global engineering education conference (EDUCON), pages 592–600

Robbins JE, Redmiles DF (2000) Cognitive support, uml adherence, and xmi interchange in argo/uml. Inf Softw Technol 42(2):79–89

Rumbaugh J, Blaha M, Premerlani W, Eddy F, WiE L et al (1991) Object-oriented modeling and design, vol 199. Prentice-hall Englewood Cliffs, NJ

Schmidt A, Kimmig D, Bittner K, Dickerhof M (2014) Teaching model-driven software development: Revealing the" great miracle" of code generation to students. Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148:97–104

Stikkolorum DR., Ho-Quang T, Chaudron MRV (2015) Revealing students' UML class diagram modelling strategies with webuml and logviz. In: 2015 41st Euromicro conference on software engineering and advanced applications, pages 275–279

Torchiano M, Tomassetti F, Ricca F, Tiso A, Reggio G (2013) Relevance, benefits, and problems of software modelling and model driven techniques-a survey in the italian industry. J Syst Softw 86(8):2110–2126

Veerman AL, Andriessen JEB, Kanselaar G (1999) Collaborative learning through computer-mediated argumentation

Westphal B (2019) Teaching software modelling in an undergraduate introduction to software engineering. In: 2019 ACM/IEEE 22nd international conference on model driven engineering languages and systems companion (MODELS-C), pages 690–699

Whittle J, Hutchinson J (2011) Mismatches between industry practice and teaching of model-driven software development. In: International conference on model driven engineering languages and systems, pages 40–47. Springer

Whittle J, Hutchinson J, Rouncefield M (2014) The state of practice in model-driven engineering. IEEE Softw 31(3):79–85

Whittle J, Hutchinson J, Rouncefield M, Burden H, Heldal R (2013) Industrial adoption of model-driven engineering: Are the tools really the problem? In: International conference on model driven engineering languages and systems, pages 1–17. Springer

Williams L, Kessler RR (2003) Pair programming illuminated. Addison-Wesley Professional

Williams L, McDowell C, Nagappan N, Fernald J, Werner L (2003) Building pair programming knowledge through a family of experiments. In: 2003 international symposium on empirical software engineering, 2003. ISESE 2003. Proceedings., pages 143–152. IEEE

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer Science & Business Media

Zayan D, Antkiewicz M, Czarnecki K (2014) Effects of using examples on structural model comprehension: a controlled experiment. In: Proceedings of the 36th international conference on software engineering, pages 955–966

Authors and Affiliations

Shalini Chakraborty[1] · Javier Troya[2] · Lola Burgueño[2] · Grischa Liebel[1]

✉ Shalini Chakraborty
shalini.chakraborty@uni-bayreuth.de

Javier Troya
jtroya@uma.es

Lola Burgueño
lolaburgueno@uma.es

Grischa Liebel
grischal@ru.is

[1] University of Bayreuth, Universitätsstraße 30, Bayreuth 95447, Germany

[2] ITIS Software, Universidad de Málaga, Blvd. Louis Pasteur, 29071 Málaga, Spain