Noname manuscript No.
(will be inserted by the editor)

We Do Not Understand What It Says – Studying Student Perceptions of Software Modelling

Shalini Chakraborty · Grischa Liebel

Received: date / Accepted: date

Abstract Background: Despite the potential benefits of software modelling, developers have shown a considerable reluctance towards its application. There is substantial existing research studying industrial use and technical challenges of modelling. However, there is a lack of detailed empirical work investigating how students perceive modelling. **Aim:** We investigate the perceptions of students towards modelling in a university environment. **Method:** We conducted a multiple case study with 5 cases (5 courses from 3 universities) and two units of analysis (student and instructor). We collected data through 21 semi-structured interviews, which we analysed using in-vivo coding and thematic analysis. **Results:** Students see some benefits of modelling, e.g., using models for planning and communicating within the group. However, several factors negatively influence their understanding of modelling, e.g., assignments with unclear expectations, irregular and insufficient feedback on their models, and lack of experience with the problem domains. **Conclusions:** Our findings help in understanding better why students struggle with software modelling, and might be reluctant to adopt it later on. This could help to improve education and training in software modelling, both at university and in industry. Specifically, we recommend that educators try to provide feedback beyond syntactical issues, and to consider using problem domains that students are knowledgeable about.

Keywords Software Modelling, Case Study, UML, Education

S. Chakraborty
Reykjavik University
Menntavegur 1, 102 Reykjavík, Iceland
ORCID: 0000-0002-9466-3766
E-mail: shalini19@ru.is

G. Liebel
Reykjavik University
Menntavegur 1, 102 Reykjavík, Iceland
ORCID: 0000-0002-3884-815X
E-mail: grischal@ru.is

1 Introduction

Software modelling has the potential to improve on several factors in software and systems engineering, such as productivity [2] or cost [20]. While these benefits seem to be appreciated in certain areas of software and systems engineering [26], software modelling is not widely adopted in software and systems engineering as a whole [14].

The reasons for a lack of adoption and use have been studied in depth, revealing issues such as poor quality code generation, lack of tool support, and lack of guidance or training [10,53,52,29,27,26]. Additionally, it has been put forward that a perceived lack of usefulness or too high effort prevents engineers from using models [14,48].

In the educational domain, substantial work exists on how to teach modelling in a university curriculum, e.g., [50,6,21,32,51,45,25]. Existing work typically comes in the form of suggested course designs, e.g., [42,50], experience reports that discuss challenges or good practices, especially tool-related,, e.g., [6,21,32,51], or papers that report quantitative studies of student opinions, e.g., [45,25]. This body of work forms a rich source of experience and inspiration for teaching modelling.

However, there is a lack of in-depth studies that aim to understand how students perceive modelling and why this is the case. Experience reports commonly suffer from various biases, as they rely on individual experiences and lack rigorous methods of data collection and analysis. Similarly, existing quantitative studies aim to provide a broad picture and can, therefore, not provide detailed explanations [46]. Finally, experience reports by researchers that have a particular focus on software and systems modelling risk being unrepresentative of instructors without such a focus. Since students will after graduation become professionals, it is important to understand their perceptions. Understanding existing challenges they face, but also perceived benefits of using models can help improve modelling education and facilitate an increased uptake in industry.

To address this gap, this paper aims to investigate students' perceptions of modelling, both in terms of benefits of and challenges with modelling. We pose the following two research questions (RQs) to address our aim:

- RQ1: What are students' perceptions about modelling?
- RQ2: What are the main challenges students face while modelling?

To answer these questions, we conducted a qualitative case study, interviewing a total of 21 subjects from 5 university courses that cover modelling. We interviewed both students and instructors, and analysed course assignments.

Our findings show that several of the perceived benefits and challenges align well with those reported in existing literature. Students find it beneficial to use models as a means of communication and handling complexity. They report difficulties such as obtaining good and fast feedback on a model. We further find that the lack of technical skills and domain knowledge in the

student population prevents them from connecting their models and judging their quality to any known domain. This finding gives a better explanation of why certain problem domains might be problematic in any given context. Further, it highlights the importance to carefully tailor modelling courses to the experience of the audience, and consider the role of modelling courses in the curriculum.

The rest of the paper is structured as follows. In Section 2 we discuss the related work on modelling in education and industry, and the importance of taking students' perception into account. In Section 3, we present the method we applied for our case study, followed by the results in Section 4 and a discussion thereof in Section 5. Finally we conclude the paper in Section 6 with a summary and plans for future work.

2 Related Work

In the following, we describe related work that investigates the use of models in industry, and work on modelling education. Finally, we describe work that aims to connect industry and education.

2.1 Models in Industry

There is substantial work on the adoption and the challenges of software modelling in industry, e.g., [19,17,18,52,26,27,48,28,29,4].

In an early study at Motorola, Baker et al. [4] find that models increase productivity and reduce defects. However, they also find that modelling tools are insufficient and lack interoperability, and that MBE does not scale sufficiently.

Mohagheghi et al. [28,29] highlight the potential of using models for simulation and testing, while also mentioning tool issues and model complexity as challenges of adopting MBE.

Hutchinson et al. [19,17,18,52] conduct several studies assessing the state of practice of MBE in industry. The authors confirm that modelling tools and model complexity are problematic, but also highlight organisational factors. Among others, they find that a lack of training hinders the adoption of modelling.

In a survey in the Italian software developing industry, Torchiano et al. [48] find that developers mainly use models for informal sketches and communication. They further report that inexperience among developers limits the use of models.

In a survey among systems engineering professionals, we find that models provide substantial benefits, e.g., in terms of increases in productivity [26]. However, several challenges in technical and non-technical areas remain, e.g., lack of training and guidance, tool shortcomings, and a lack of tool interoperability. Following up with an in-depth qualitative study at two automotive companies, we report that models are used primarily for communication and

to handle complexity. Stakeholders further prefer sketches and informal models [27]. System models often use in other domains, such as healthcare, for communication and to provide better organisational support. In [22], authors discuss the application of event-based models such as Petri nets to capture the various operations running in a hospital. Although found helpful, the authors also highlight certain challenges like needing more understanding in adopting the models from healthcare associations. Considering UML specifically, the healthcare domain has multiple evidence of using UML models to design the domains [49,35] and human-centric tasks [5]. However, in the latter, the authors mentioned a need for more explicit representation from the models. In the healthcare domains where the cognitive behaviour of the actors is highly involved in the process, software models that are used for designing those processes should provide the ability to include the behaviours.

2.2 Models in Education

Research on models in education exists primarily in three forms: (i) solution proposals on how to teach modelling, including tools tailored to an educational context, (ii) experience reports on modelling education, and (iii) surveys among students.

An example of the first category is the practical approach to teaching model-driven software development proposed by Schmidt et al. [42]. In the proposed course, students develop a code generator using standard software development tools. Similarly, Westphal [50] describes the design of an SE course that focuses heavily on modelling. In [12], authors proposed a course where they used students both as language designers and its users to evaluate the usability of software language engineering (SLE). Gonnord et al. [13] try a reverse approach, and with the students, they journey from low-level C code to designing modelling language workbench. In all cases, evaluation of the proposed course design relies on student feedback.

Numerous experience reports exist related to modelling education, e.g., [3, 32,21]. Akayama et al. [3] share their experience and opinion on tool use in software modelling education. The paper describes different approaches taken by the individual authors and provides a discussion of factors such as modelling tools vs pen and paper, the conflict between the concepts of design and programming, and how to measure the quality of models. Paige et al. [32] discuss what they consider to be bad practices of teaching modelling. They name bad practices such as covering a too broad range of modelling-related topics, and focusing on syntax instead of semantics. Similarly, Kolovos and Cabot [21] present a corpus of use cases for courses teaching MBE. The authors state that modelling courses regularly suffer from the use of uninteresting or irrelevant examples, and therefore propose a list of use cases suited to teach modelling. Daniel L. Moody [30] mentions a need for more effort in designing visual syntax for notations. Therefore, propose a tutorial that defines a " set of principles for designing cognitively effective visual notations: ones that are

optimised for human communication and problem solving". Ciccozzi et al. [8] present a survey among educators on how modelling is taught. The authors find that educators see the focus on tools critical, as it might prevent students from understanding the core principles of modelling.

Several empirical studies exist on modelling education, e.g., [36,45,1,24, 15]. Reuter et al. [36] study students' problems with UML diagrams over two modelling courses. As a result, the authors present a catalogue of problems with UML diagrams. In a similar direction, Stikkolorum et al. [45] study student problems and modelling strategies in UML Class diagrams by presenting students with a specialised UML editor that incorporates feedback mechanisms. The results reveal four distinct strategies, and a number of problems such as choosing the right syntax elements. Agner et al. [1] conduct a survey on modelling tool use among 117 students. The authors report issues such as a lack of feedback, difficulties in drawing the diagrams and tool complexity. Hammouda et al. [15] compare the use of modelling CASE tools to pen and paper use. Using a survey, they evaluate students' perceptions of the differences, finding no clear advantage for either approach. In the context of modelling tools in education, the tool Umple needs to be highlighted [11]. Umple is a tool that allows to create UML models in a textual concrete syntax close to object-oriented languages like Java. Furthermore, code in many different general-purpose programmming languages can generated directly from Umple models, thus allowing for direct feedback from models. Surveys with students have shown positive results on the use of Umple [23,24]. Umple is successfully used by several instructors teaching modelling. However, to our knowledge, it is currently not widely used on a global scale. Finally, we conducted several case studies on tool use in modelling education [24,25]. We find that students can use industrial modelling tools successfully, but require substantial coaching in how to use the tools, with a dedicated tool champion present in the course. We further find that the tools' inability to provide adequate feedback impacts the tool acceptance. In an attempt to connect industry practice to education, Whittle and Hutchinson [51] present essential differences between industry practice and education concerning software modelling. The authors find that modelling education is more UML-centric, whereas industry is placing more emphasis on abstract models. Furthermore, in industry it is more common to use a bottom-up approach to adoption, whereas modelling is typically taught in a top-down fashion.

3 Research Method

Our goal is to investigate students' perceptions, especially their challenges in learning and applying software modelling in a university environment. To do so, we conducted a multiple-case study. Runeson at al. [39] define a case study as an "empirical investigation of a software engineering phenomenon within its real-life context" where evidence can be drawn from multiple sources, more importantly from real-life experiments involving human participants [54].

The phenomenon we study is the experience of students learning software modelling at university. In total, we interviewed 16 students from 3 different universities and 5 different courses. Additionally, we interviewed instructors of the 5 courses we considered. Three courses were at bachelor level and two were from masters level. In addition to the interviews, we consulted assignment details for the courses.

Whether students can or should be used as study subjects is a long-debated question in SE [38, 41, 44, 9]. In particular, students can be valid study subjects under certain conditions, e.g., if novice software engineers are studied [9]. In our case, we aim to study how students perceive modelling, in contrast to existing studies that investigate industrial cases. Therefore, the choice of students as study subjects is valid.

The study setup is described in detail in the following sub-sections.

3.1 Case Study Design

3.1.1 Pilot Study

Before starting the actual study, we conducted a pilot study with three interviewees, two doctoral students and one bachelor student. Of the two doctoral students, one has prior knowledge and experience of software modelling. The other student has no experience in software modelling. We selected the combination because we wanted to check how our questions would be received by different persons with various software modelling background, as we are dealing with students from different countries, courses and backgrounds. Then we conducted another interview with a bachelor student who had taken a modelling course at our university. Our focus was to check the interview guide. Also, the pilot study helped us to estimate the overall interview time.

3.1.2 Cases and Recruitment

We designed a multiple *embedded* case study [39] with five cases, each case is represented by a course-university pair. Table 1 shows the cases and the interviewees per case (using anonymous identifiers). Initially, we contacted the instructors of the courses based on our own contacts, and once they agreed, we advertised the study to students of that course. C1 was selected through convenience sampling, as it is a course at our home university. It is a typical "UML course" in the sense that basic UML diagrams are introduced. The course starts with user interface methods, later students learn about software modelling, including the UML use case, class, sequence, and activity diagrams. Assignments related to these diagrams are given. The course has two instructors, where I2 was responsible for the modelling part. However, both I1 and I2 were in charge of the exams. Therefore, we interviewed both instructors. Neither of the two instructors is active in modelling research. C2 was selected as a comparative case, as it was a basic UML course taught by the same instructor

I2 as C1, at the same university. C2 concentrates mainly on software modelling, with more focus on UML in the assignments. Similarly, the difficulty level is higher. C1 and C2 are both bachelor courses. However, C2 is located in the software engineering program and C1 in computer science. C3 to C5 were selected as revelatory cases, as they exhibit one or several differences to C1 and C2. One of the main differences is the change of university/country. C2 and C3 are similar in terms of the course syllabus. Both courses teach system design and UML from scratch. C4 and C5 are courses in the same university and are master courses. While, in C4, UML is taught from scratch, C5 focuses on using UML diagrams for architecture and testing. Apart from C1, all four courses include implementation tasks.

Participation in the interviewees was voluntary and instructors were not aware of who participated in the interviews. We asked interested students to contact the researchers via mail, and once students initiated the contact, we sent them an interview meeting invitation and a consent form. All interviews were online. To appreciate students' participation, we offered them movie vouchers or did a donation to charity on their behalf.

Table 1 Case Summary

Code	Level	Course Description	Students	Instructors
C1	Bachelor	Software analysis and prototype design using user interviews	C1_(S1, S2, S3, S4, S5)	I1, I2
C2	Bachelor	Software requirement analysis and design	C2_(S1, S2, S3)	I2
C3	Bachelor	Software requirement specification and design	C3_(S1, S2)	I3
C4	Master	System specification, design and testing	C4_(S1, S2, S3)	I4
C5	Master	Software architecture design and quality analysis	C5_(S1, S2, S3)	I5

3.2 Data Collection

Based on the pilot study, we decided to conduct interviews for approximately 40 minutes. The student interviews were divided into two set of questions: **introductory question** and **model questions**. In the **introductory questions**, we added questions about a student's background, previous work experience with software models. The **model questions** includes students' personal experiences during courses, challenges in learning modelling, completing assignments and overall perception of software modelling. The instructor questionnaire includes details about the course structure, assignments, feedback from students and the instructor's challenges with the course. All interviews conducted remotely through Zoom or Microsoft Teams, and their build-in recording features. Before recording, we asked permission from each interviewee and received their signed consent form. Interviewees could additionally give their consent that we would publish their anonymised interview tran-

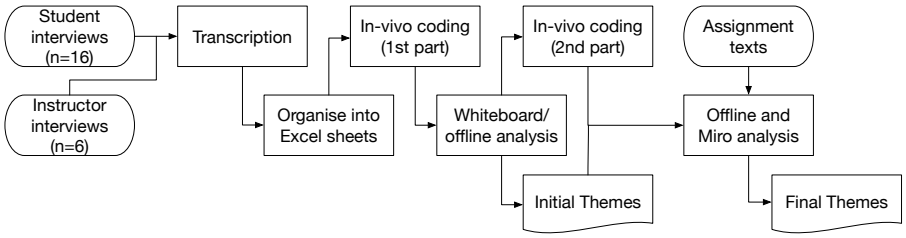

Fig. 1 Data Analysis Process. Rounded rectangles denote data sources, regular rectangles denote activities, rectangles with wavy bottom line denote artefacts, and arrows denote information flow.

script. 13 interviewees consented to publishing. The interview guides and the transcripts can be found on Zenodo [7].

3.3 Data Analysis

The data analysis process is depicted in Figure 1. After conducting the interviews (steps 1a and 1b in Figure 1), we transcribed the interviews using transcription services[1]. In case of automated transcriptions, we post-processed the transcripts to improve their quality. The two researchers then applied in-vivo coding separately on each interviewee transcript. We used in-vivo coding [40], as it helps to highlight participants' opinions by using the actual spoken words. In our case, interviewees were from different countries and cultural backgrounds, and thus used varying vocabulary to describe their personal experiences. After conducting the first five interviews (C1_S1, C1_S2, C1_S3, C1_S4 and C2_S1), we coded and then jointly discussed the resulting codes, grouping them into initially 10 themes. These themes related to benefits and challenges of modelling, as well as general categories refering to course feedback and concerns. We refined the themes as analysis progressed and more interviews were added.

Once we finalised the themes based on student data, we coded the instructor data and compared it with the existing themes. We used the instructor interviewees for three purposes, namely (a) to understand the course context, (b) to better understand the student perceptions and to check whether the instructors had the same views and (c) to obtain ideas for potential best practices. For sorting quotes and themes, we used the online whiteboard tool Miro[4].

As a second data source for our analysis, we used course assignments to cross-check themes and quotes that related to, e.g., assignments, work load, or grading. We contacted the instructors after their respective interviews for assignment details, which included the assignment structure, time, instructions, and point distribution. Unfortunately, only three instructors (C1/I1, C3/I3

[1] We used Konch[2] and Go Transcript[3].

[4] https://miro.com/

and C4/I4) responded. However, these three courses consist of the majority of the students. For the other two courses, we used the instructor interview data, and tried to gather as much information about course timing, assignments, and grading (see questions 3 and 4, in Appendix A2).

We checked the assignments for the problem motivation, the actual tasks, the required problem domain knowledge, clarity of instructions and the total time dedicated to modelling tasks. We then compared these aspects to statements made during the interviews.

The total time of data collection and analysis was approximately one year. The resulting themes related to modelling benefits are listed in Table 2, with example quote for each theme. Similarly, Table 3 lists the themes related to modelling challenges.

Table 2 Final themes related to modelling benefits, with a short description and a sample quote associated with each theme.

Theme	Description	Example statement
B: Same Page	Benefits of modelling in communication and coordination within teams.	*I think it's important for documentation, as well as getting everyone on the same page*
B: Better Planning	Modelling helps in planning ahead, for understanding the requirements, potential design alternatives, but also to plan implementation.	*But the thing I liked this is when you feel like you have a plan and you know what are you gonna do*
B: Better Understanding of Code	Benefits of modelling to help understanding a code base without reading the code in detail.	*I really like the idea of seeing it as a language to talk about code, because I will never read someone's code if they ask for feedback-*
B: Maintenance/ Documentation	Benefits of modelling as a way to help maintaining a system and to document what it does.	*Usually, it's other people who will maintain software, so they need to read what you make if they need to understand.*
B: Personal Preferences	Benefits related to individual preferences.	*Yeah, yeah we used that because it was um..prettier, than this is the whole perfectionism thing with um, it was prettier than coding*
B: Doubtful Benefits	Doubts regarding any benefits of modelling.	*[..]]you can't be mastering everything and [I am] more into implementation rather than modelling things*

3.4 Validity threats

We conducted an exploratory [39] case study, where we primarily collected data through interviews. For the analysis, we leaned towards interpretivism. An interpretivist approach means that *"humans construct knowledge as they interpret their experiences of and in the world; rejecting the objectivist notion that knowledge is simply there to be identified and collected"* [16, 33]. In

Table 3 Final themes related to modelling challenges, with a short description and a sample quote associated with each theme.

Theme	Description	Example statement
C: Unclear Expectations	Confusion on what the purpose of modelling is, often as expectations are not clearly communicated.	*At the end it's like 'Oops, maybe I should have designed', but then I wouldn't know how to design it.*
C: Irregular and Unclear Feedback	Feedback is provided too seldom, is not clear, or is restricted to formalities, such as the diagram syntax.	*Writing for hours and then a teacher be like no this is wrong*
C: Lack of Expertise in the Problem Domain	Students are provided with assignments in a domain they are unfamiliar with. Therefore, they struggle to relate their learning to something known.	*we didn't have much experience in programming and designing and modeling the app, I think the biggest challenge was in the first week that we were just flowing with our ideas how the app should look like without actually knowing how the end picture of the class diagrams should look like*
C: Time and Repetition	Modelling consumes time, and requires repetition to master. These two factors are often in conflict in university courses, and students struggle to see the value of repeatedly improving their models..	*Later on, we had to fix it at least 10 times. After writing the app, we had to go back to our diagrams and redo them how our final vision was*
C: Notation	Struggles with the complexity of the UML notation.	*One thing that comes up is with the state diagram and activity diagram, which one is supposed to do what*
C: Tooling	Modelling tools can cause numerous difficulties, such as poor usability.	*I hate PlantUML. I can't read the diagram that comes out of it. It's all over the place*
C: Lack of Cooperation	Challenges due to difficulties in collaborating in teams, and due to lack of professionalism in students' attitudes.	*[..] one or two would make the model and the rest maybe don't understand. Somehow, I don't know how to solve this problem.*

our work, adopting to interpretivism is suitable as we seek answers for our RQs through the interviewees' perspective, based on their knowledge of the addressed subject and cultural background. The understanding of our knowledge is therefore relative to the person and their personal experience.

Following the work of Petersen et al. [34], we present the validity threats of our work and the measures we have taken to mitigate them in the following.

3.4.1 Transferability

Transferability describes to what extent results from the study can be transferred to cases that resemble the case under study [34]. Cultural differences, the influence of teachers and their teaching practices, and the course subjects limit transferability in our study. We aim to ensure a substantial level of transferability by basing our analysis on five cases. Nevertheless, all five cases

teach modelling for analysis tasks on a high level of abstraction, i.e., for requirements, architecture and design purposes. Specifically, none of the courses covers model transformation or other tasks that require formal models. Furthermore, all five courses are located in Northern European Universities, thus potentially limiting the applicability to other countries. Finally, perceptions of modelling are tightly connected to perceptions of the course. Specifically, the quality of teaching and the instructor's expertise on the topic could positively or negatively affect the studied perceptions. To avoid this, we selected both cases where the instructors do and do not have a research background in software modelling. Pedagogical quality is harder to assess. However, we did not get the impression from students that individual courses had a low level of pedagogical quality. Nevertheless, this might be a threat to transferability.

To further allow for transferability, we conducted a substantial number of interviews (21). We reached a saturation point in our themes after the 16 student interviews, and hence stopped including further cases or interviewees. That is, in the last batch of interviews (n=6) we analysed, no new themes emerged that we had not yet included in our analysis.

3.4.2 Credibility

Credibility describes to what extent findings have been distorted by the researchers [34].

We tried to avoid distortion of the reported findings by grounding the analysis in in-vivo codes directly taken from the verbatim interview transcripts. Furthermore, we performed member checking to get feedback on the extracted themes from our interviewees. Only few interviewees answered this call, but those confirmed the credibility of the found themes.

All interviews were recorded, and data analysis performed on the verbatim transcripts. Additionally, we publish the anonymised transcripts of the 13 interviewees who consented to this. This should ensure credibility of the findings.

3.4.3 Confirmability

Confirmability describes the extent to which conclusions made by researchers follow from the observed data [34].

To allow for confirmability, we presented the way of coding in depth. Furthermore, we give example quotes for each theme, and publish anonymised student transcripts. Note that this does not necessarily ensure reliability, i.e., that analyses conducted by other researchers would yield precisely the same results. We discussed themes and codes and aimed to find consensus in our analyses. Nevertheless, we allowed for some disagreements to account for subjective impressions or opinions, as is common in interpretivist research.

4 Results

In this section we report and discuss the findings of our case study.

4.1 Student Perception of Software Modelling (RQ1)

Based on the interview data, we observe that students see specific benefits of modelling, primarily for areas where informal models might be sufficient, such as obtaining a system overview or conceptual understanding of the problem domain. However, many are skeptical as to what value modelling has for detailed system design. In the following, we discuss the students' perceptions and provide adequate interview quotes that support the categories. An overview of how many students mentioned the perceived benefits is depicted in Figure 2.

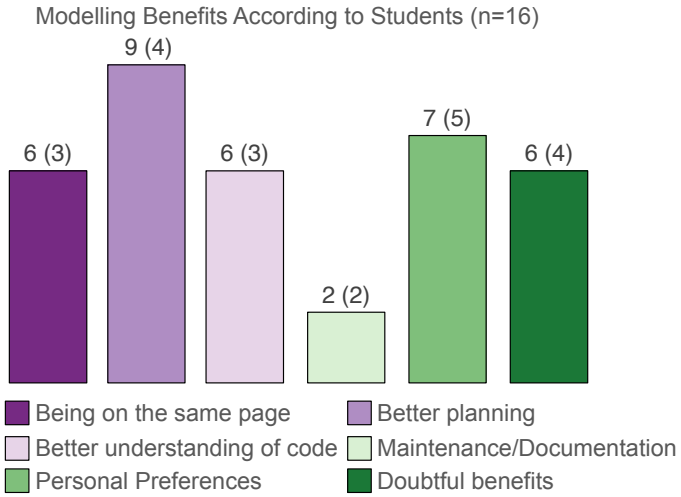

Fig. 2 Benefits as supported by the student interviewees. Numbers over the bars represent the total amount of students mentioning the benefit, and the number of cases in which students mentioned the benefit in parenthesis.

4.1.1 Being on the Same Page

Several students find that modelling helps their groups communicate better with each other. Students mention modelling helps them to express their ideas, share work and make decisions as a group. For example, students stated:

> *"I think it's important for documentation, as well as getting everyone on the same page"* — C2_S1

> *"It's a lot easier to work together when you have the diagrams."* — C1_S5

> *"It helps every group member to know what exactly they want"* — C3_S2

4.1.2 Better Planning

Students find modelling helpful in planning the development, i.e., designing systems. Students appreciate that they can visually map the system and plan the development through modelling.

> *"But the thing I liked this is when you feel like you have a plan and you know what are you gonna do"* — C1_S2

> *"Class diagrams are really, really helpful. Before starting your project...because you can see and have the view of your classes"* — C5_S3

4.1.3 Better Understanding of Code

In relation to programming, some students found models helpful to get an overview of the code on a higher level of abstraction. That is, they stated:

> *"I really like the idea of seeing it as a language to talk about code, because I will never read someone's code if they ask for feedback"* — C3_S1

> *"If you have a diagram for it, we would get a better understanding of the code itself."* — C2_S3

An aspect of this is a top-down modelling approach, where a better understanding of the system is obtained through breaking down the system in steps, as noted by some of our interviewees.

> *"If you are making a program that has more than 200 lines of code then you probably gonna have to model it."* — C1_S1

> *"that was a very nice experience of seeing how you start with a kind of vague idea and then start to break it down more concrete classes"* — C3_S1

4.1.4 Maintenance/Documentation

Students find modelling to be helpful to prepare for future tasks, such as maintaining and documenting the product. However, in our interview data only students with industry background expressed this benefit.

> *"Usually, it's other people who will maintain software, so they need to read what you make if they need to understand."* — C1_S3

> *"Unless you don't have the documentation, you will end up with just a total mess"* — C4_S2

While looking at the assignment details and interview data regarding that, we realized none of the courses explicitly mention maintenance/documentation. The assignments ask for an overview of the diagrams, but there aren't any requirements for documentation. The importance of maintaining a document or how to do that still needs to be clarified for students.

4.1.5 Personal Preferences

In addition to the benefits stated above, some students use models in terms of personal preference. For example:

> *"Yeah, yeah we used that because it was um..prettier, than this is the whole perfectionism thing with um, it was prettier than coding"* — C1_S2

> *"It helps me basically get my idea out there a lot better. Basically, when I'm starting a project, I like modeling the higher level and just going deeper and deeper and deeper."* — C2_S1

4.1.6 Doubtful Benefits

Despite experiencing benefits, several students are doubtful whether the benefits of modelling outweigh the issues, and if they will apply models in the future. We received several statements of students saying they will most likely not apply models in the future.

> *"I think it's a great experience that we can have this now. Not in the future, in our jobs"* — C3_S2

> *"[..]]you can't be mastering everything and [I am] more into implementation rather than modelling things"* — C4_S2

Interestingly, both statements above were made by students with industrial software development experience. The same students however admitted that in their experience they were tasked with implementation only, and not with high-level tasks such as system design or requirements engineering.

One of the instructors confirmed that many of their students would question the application of modelling in industry:

> *"It's the same experience that I have when people talk about this course, they're asking, 'Is anyone using it?"'* — I2

We further discuss the challenges with modelling in the next section.

4.2 Modelling Challenges (RQ2)

Despite benefits observed by the majority of the students, they experience several challenges related to modelling. These relate, among others, to tooling, how to choose the right notation, what to express in the models, and how to apply it to unfamiliar domains. We will discuss these challenges in the following, focusing on the student lens, and complementing their views with the instructor perspective. Overall, we extracted 8 types of challenges. The support by the different interviewees is depicted in Figure 3 and Figure 4.

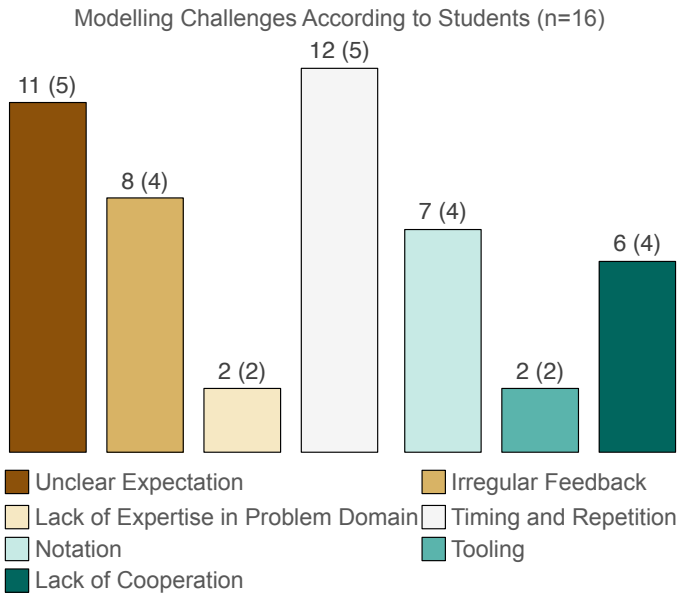

Fig. 3 Student challenges as supported by the interviewees. Numbers over the bars represent the total amount of students mentioning the challenge, and the number of cases in which students mentioned the challenge in parenthesis.

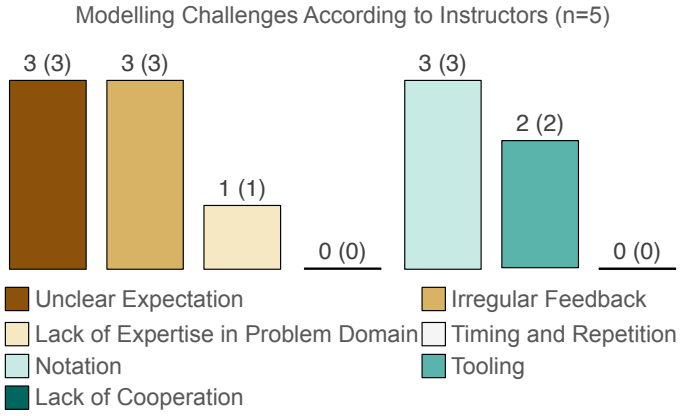

Fig. 4 Student challenges as supported by the instructor interviewees. Numbers over the bars represent the total amount of instructors mentioning the challenge, and the number of cases in which instructors mentioned the challenge in parenthesis.

4.2.1 Unclear Expectations

In university modelling courses, students typically receive assignments to create models. However, they often struggle to understand what is expected of them, and how to create the models.

> *"We started coding and at the end of the day, we didn't know what we did"* — C4_S3

In particular, this is caused by a lack of knowledge in programming, software architecture, and other practical skills necessary to envision a "good" design:

> *"At the end it's like 'Oops, maybe I should have designed', but then I wouldn't know how to design it. "* — C3_S1

Additionally, assignments are often formulated in a way that is in contrast to how they are assessed, e.g., by asking students to create a *prescriptive* model, which is then assessed by how closely it resembles the final code, i.e., in a *descriptive* way. Students quickly pick up this discrepancy in grading and optimise their efforts towards the grading. That is, they take shortcuts initially, and simply update the model later to reflect their actual code.

> *"we had to, we had to basically just create a class diagram out of the code that they wrote, instead of writing a code out of the class diagram"* — C1_S1

4.2.2 Irregular and Unclear Feedback

Students are dissatisfied with the type of feedback they receive. The evaluation criteria often demand *"Diagrams are well structured and provide good overview"*- assignment details from C3, *"Explanation of good design suggestions"*-assignment details from C1, which is vague and do not leave space for constructive, detailed feedbacks. As modelling is often a qualitative task, teachers often revert to giving feedback on objective things such as the diagram syntax. However, this is not perceived as useful feedback.

> *"So no one can say anything to us. It's good or not. Just we should follow some rules about the diagram, for example, what is solid line, what is dashed line."* — C5_S1

Instead, students would like more "holding hands" with regular feedback.

> *"Maybe because it's the first year, this is the first introduction to the subject, so people miss a lot of points, but if it had another level, teach people more, and take their hand more to do these diagrams themselves, that would be good"* — C1_S3

In particular, since there is no automated feedback as, e.g., in programming, students require timely and regular feedback.

> *"writing for hours and then a teacher be like no this is wrong"* — C1_S1

> *"Reply to emails more than five or six hours"* — C4_S1

The regularity and quality of feedback was further complicated by Covid-19, as all 5 cases used remote teaching.

4.2.3 Lack of Expertise in the Problem Domain

Deep knowledge of the problem domain is necessary to perform many software engineering tasks in a satisfactory manner, e.g., programming [31] or program understanding [37]. If this domain knowledge is lacking or entirely missing, software engineering tasks cannot be performed well. Allowing students to work with a known problem increases their interest and participation. Several students stated that, in addition to just learning modelling, they were also lacking domain knowledge and programming experience. Therefore, they were unable to relate their models to familiar concepts.

> *"we didn't have much experience in programming and designing and modeling the app, I think the biggest challenge was in the first week that we were just flowing with our ideas how the app should look like without actually knowing how the end picture of the class diagrams should look like"* — C1_S4

Compared to other introductory topics, such as programming, this makes modelling particularly complex: Students are expected to learn a new concept (modelling), neither having problem domain nor software design/architecture experience. In addition, the models used in the courses we studied do not provide any kind of automated feedback, such as compiler/runtime error messages in programming. This leads to feelings of "just drawing something" in students, without an anchor to connect their models to.

In a similar direction, the chosen problem domain might also affect how representative the learning is for actual systems in industry, or how suited modelling is for the given problem. For instance, one student noted:

> *"we got to know all the basics and the whole idea and how it's supposed to look like, but not how to use it in the future on the bigger picture and the scale of the app"* — C1_S4

4.2.4 Time and Repetition

Students complain that modelling takes time, and courses are tightly scheduled. Students feel pressure due to the bulk of assignments, especially when they are not fully aware of what is expected from them. Given that pressure, they try to prioritise and minimise effort where possible. However, arguably, learning how to model will require the students to make many mistakes and correct them in iterations. Together with the lack of feedback mentioned above, students perceive these iterative improvements as a waste of time.

> *"You run it once and then probably change. It's just a waste"* — C3_S1

> *"we designed something, it didn't work, so we needed to change it "* — C2_S1

> *"Later on, we had to fix it at least 10 times. After writing the app, we had to go back to our diagrams and redo them how our final vision was"* — C1_S4

Interestingly, these students did not perceive that iterative changes helped them understand the problem domain better, but rather saw it as a burden with unclear benefit. One of the reasons is courses contain individual assignments without a continuation or connection between themselves for some cases

(C2 and C3). For each problem students are instructed to draw a specific diagram and then move on to something else. An exception we observed for C4 and C5 since, in C5, students are applying what they learned in C4.

4.2.5 Notation

Keng et al. [43] reported different learning challenges with UML over a decade ago. In our cases, we observe similar challenges. Students find it hard to remember the syntax of different diagrams and their purpose, pointing to a lack of prior knowledge of UML.

> *"many different types of diagrams to represent the same thing"* — C1_S1

> *"The hardest part about drawing UML diagrams is always remembering what exactly the syntax is"* — C2_S1

Specifically, our interviewees are most often pointing out UML state machine and activity diagrams as confusing.

> *"I have this image in my head where the state diagram is not the UI actions. I can't wrap my head around it."* — C2_S3

> *"One thing that comes up is with the state diagram and activity diagram, which one is supposed to do what. "* — C1_S5

The instructors confirm this view.

> *"here's a bit of struggle in between the concepts that they express in a diagram and the particular diagram types"* — I3

> *"they've been mixing those pretty well up [state and activity diagrams]"* — I2

One instructor offered the explanation that this relates to a lack of experience with object-oriented programming in general.

> *"they don't have a lot of experience using object-oriented programming."* — I2

This quote reinforces our discussion above, that students are exposed to various unknown activities when learning how to model, and therefore lack knowledge they can anchor their modelling experiences to.

4.2.6 Tooling

None of the 5 cases we investigated had any requirements for using a particular modelling tool. Correspondingly, students did not voice any tool challenges. Students used multiple tools for their assignments, e.g., Draw.io, PlantUML, and Lucidchart. Also, some students used plain pen and paper for drawing and communicating their diagrams. Students appreciate this choice.

> *"I think actually giving people the freedom to do what they want with the tools is really nice, because some people actually liked drawing it on an iPad or something"* — C2_S1

In particular, some students also showed initiative at choosing different tools depending on the situation.

> *"When I did the prototype of my app, I used the Figma app on the internet. It's really good to visualize the prototype. Then we used the Lucidchart and draw.io, I think it's diagrams.net now. This is for the diagrams. Then we just used the whiteboard to draw it if we needed to because we just gathered together and we'd just throw it on the whiteboard and then we just translated it into diagrams or Lucidchart"* — C1_S4

While offering choice in tooling clearly addresses many tool issues, it has the limitation that it only works if the course and the course assignments do not rely on specific tool features, such as code generation or simulation capabilities. This is a limitation in our study, as all our cases introduced UML only as a means to document, plan, and to communicate, without any automated processing of models in the form of model transformation or other facilities.

4.2.7 Lack of Cooperation

In all 5 cases, students had to cooperate in teams. In fact, modelling was in all cases also intended as a way to facilitate communication in the group. However, in practice they struggled to cooperate and to appreciate the use of models for communication purposes.

> *"We are six, for example, but one or two would make the model and the rest maybe don't understand. Somehow, I don't know how to solve this problem."* — C1_S3

> *"Some people don't like to draw., but we're supposed to work together."* — C5_S1

4.2.8 Summary

Overall, we observe that students struggle with various aspects of models (as depicted in Figures 3 and 4). Apart from classic issues such as learning an unknown notation (be it modelling or not), we find that students lack a connection to previous knowledge. That is, due to a lack of domain, design and programming knowledge, and a lack of automated feedback from modelling tools, they cannot create an anchor. Essentially, they produce a model that they cannot judge on any other level than notation. One student summarised this sentiment as follows:

> *"The diagram is supposed to speak for yourself [sic]. But we didn't understand what it was saying"* — RU_G_S1

5 Discussion

Several of our findings that relate both to perceived benefits and challenges of models are specific to the selection of courses we studied. That is, in all five courses, models are created for planning, documentation, design, and communication purposes. In contrast, topics that require formal models, e.g., for code generation, formal verification, or simulation, are not included. This certainly limits how general our findings are with respect to software modelling as a whole. Nevertheless, we believe that a large percentage of computer science

students worldwide get exposed to software modelling through similar courses, i.e., the stereotypical "UML courses". As such, our findings have the potential to be transferable to many students. Similarly, even if students take more advanced courses on software modelling, their beliefs and preconceptions about modelling can be irreversibly shaped by their first exposure to the topic. As such, we believe our findings are particularly relevant.

In a similar direction, there is a potential threat that challenges perceived by students might not so much relate to software modelling as a course topic, but instead by the pedagogical quality of the courses. We did not try to assess the quality of the individual courses, but at least we cover a variety of expertise in modelling on the lecturers side, ranging from instructors that do not work on models in research, to instructors that publish in software modelling venues and have or have had a particular focus on the topic. Therefore, we do not believe that the students' experiences can simply be related to issues in teaching.

Modelling tools have been reported consistently as a concern in studies on software modelling, both related to industry and education. In contrast, we did not find much reported challenges with tooling. To some extent, we believe the reason for this absence of tooling challenges is the focus on "drawing" diagrams rather than formal modelling, and the freedom of choosing a modelling tool of the students' choice. None of the cases used tools that are specifically designed for education, such as the Umple modelling tool [11]. Since surveys with students have shown positive results [23], we believe studying such a course in depth using qualitative methods could yield further insights that are complementary to ours. However, we also believe that the use of software modelling tools tailored towards education is currently not representative.

While this study focused purely on the educational context, it is interesting to discuss potential similarities and differences to industrial practice.

First, we observe that models are appreciated for communication purposes and to handle system complexity in our cases, something that is also reported in industry [14,26,47]. Other benefits reported in industry, such as simulation or verification capabilities do not apply in our cases, since all five courses used modelling only on an informal level to express models for planning, communication, and coordination. Similarly, it is hard to reason about improvements reported from industry, e.g., in terms of productivity [4,28].

Tooling issues are a frequent topic in industry, e.g., reported in [53,52,19, 17,53,26,27]. In contrast, none of our interviewees raised tool issues. One explanation for this observation is clearly that none of our courses mandated a CASE tool for modelling, but instead left that choice to the students. Similarly, while models were often assessed for semantic correctness, the specified purposes of the models did not require syntactically or semantically correct models. Together with a lack of requirements for interoperability between tools, this removes most of the issues contemporary modelling CASE tools have. Nevertheless, it is positive to observe that our interviewees did not finish their courses on modelling with the perception that modelling tools are bad, as is

commonly reported in literature on modelling education, e.g., in [25,3]. Such a negative perception could lead to a reduced uptake of modelling in industry.

Our interviewees raise several challenges that relate to a lack of guidance, feedback, and clarity when it comes to modelling. While they primarily perceive this as an issue in how assignments are set up and how the courses are designed, the challenges resonate with those in industry. For instance, a lack of training and guidance is raised in several empirical studies on modelling in industry, e.g., [14,48,26]. Whittle et al. [53] highlight that organisational and process factors play a major role in the use of MDE. Similar issues are raised by our interviewees in the educational context, namely that modelling assignments need suitable processes that allow for iterations and quick feedback loops. This is especially important as models were not used for automated tasks in any of our cases, i.e., there was no automatic feedback generated.

An important difference of the educational setting to industry is that students often lack both the technical background, e.g., in terms of programming experience, and the domain knowledge required for the example domains. Therefore, they struggle to contextualise the value and the quality of their models, often leading to a perception of doing a meaningless drawing task. Publications on modelling education sometimes argue for the use of realistic examples and caution against toy examples, e.g., [21]. However, also realistic examples have their pitfalls. While unrealistic domains are just that, unrealistic, they can also expose students to a known domain, or at least a low level of complexity due to a domain that is artificial and potentially more controlled and restricted. From this point of view, toy examples can be a suitable tool for modelling education. Given our findings in this study, we would advise to use primarily example domains the students have sufficient knowledge of. Whether or not these domains are then simplified artificially or not should depend primarily on context factors in the course, e.g., the time dedicated to understand and work with the domain, the depth of the covered modelling concepts, and the student level. Another option is to let students choose a domain on their own, which can potentially increase their motivation. However, they run the risk of selecting a domain which is particularly unsuited given the course assignments.

Finally, we observe that several benefits and challenges voiced by our interviewees are directly reflected in industrial surveys on the use of models, e.g., the usefulness of models for communication [26,14,48], or the sentiment that models are not worth the effort [14,48]. While we are not able to answer this based on our study, we would like to highlight the possibility that these opinions are shaped during university education and then maintained later on. That is, addressing these issues while educating modellers could lead to a higher uptake and appreciation of modelling in industry.

6 Conclusion

We conducted a case study investigating the perceptions students have of software modelling. To do so, we conducted 21 interviews with students and instructors in five courses at 3 universities, and consulted assignment descriptions of three of the five courses. The results offer rich insights into how students perceive modelling. While related work on the topic exists, it is usually in the form of survey studies, opinion and experience papers, and industrial case studies.

Several findings from related work are confirmed in our study, e.g., that while seeing the benefit of modelling for planning or communication, many students perceive modelling as not worth the effort or as an outright waste of time. They struggle with the fast pace of courses and a lack of repetition and in-depth feedback that goes beyond the diagram syntax. However, we additionally find differences to existing literature, or deeper explanations of phenomena observed in related work. For instance, our interviewees reported no tooling-related challenges, potentially as they were allowed to choose their tool of choice in all cases, and as none of the five courses used any model-based techniques that would require a specific CASE tool. Finally, we find that students struggle to understand modelling, as they are often at the same time lacking knowledge in areas related to the course projects, i.e., the problem domain, the intended task (such as design or architecture), object orientation, and programming skills. As there is often no automatic feedback from modelling tools, this means students cannot anchor their models to any known domain.

Our study is of interest to university researchers, educators and professionals. Specifically for educators, it confirms many existing beliefs, but also highlights new details that can be considered when teaching modelling, i.e., a more nuanced view on how the choice of a problem domain affects the students' perceptions of modelling. For modelling researchers, it can open up new directions on how to improve guidance and training in modelling, something that also professionals regularly report as challenging. Finally, professionals can use our findings to better understand what experiences university graduates have with modelling, which benefits they perceive, and why this is the case.

Acknowledgement

We would like to thank the interviewees for participating in the study.

Declarations

Competing Interests

The authors declare that they have no conflict of interest.

Data Availability

We publish 13 out of the 21 anonymised interview transcripts on Zenodo, `https://doi.org/10.5281/zenodo.6913780`. We did not receive consent to publish the remaining 8 transcripts.

References

1. Agner, L.T., Lethbridge, T.C., Soares, I.W.: Student experience with software modeling tools. Software and Systems Modeling **18**(5), 3025–3047 (2019). DOI 10.1007/s10270-018-00709-6
2. Agner, L.T.W., Soares, I.W., Stadzisz, P.C., Simão, J.M.: A brazilian survey on UML and model-driven practices for embedded software development. Journal of Systems and Software **86**(4), 997–1005 (2013). SI : Software Engineering in Brazil: Retrospective and Prospective Views
3. Akayama, S., Demuth, B., Lethbridge, T.C., Scholz, M., Stevens, P., Stikkolorum, D.R.: Tool use in software modelling education. In: EduSymp@ MoDELS (2013)
4. Baker, P., Loh, S., Weil, F.: Model-driven engineering in a large industrial context—motorola case study. In: International Conference on Model Driven Engineering Languages and Systems, pp. 476–491. Springer (2005)
5. Bernonville, S., Kolski, C., Beuscart-Zephir, M.C.: Contribution and limits of uml models for task modelling in a complex organizational context: case study in the healthcare domain. In: Internet and Information Technology in Modern Organizations: Challenges & Answers, Proceedings of The 5th International Business Information Management Association Conference, pp. 119–127. IBIMA (2005)
6. Burgueño, L., Vallecillo, A., Gogolla, M.: Teaching uml and ocl models and their validation to software engineering students: an experience report. Computer Science Education pp. 1–19 (2018). DOI 10.1080/08993408.2018.1462000
7. Chakraborty, S., Liebel, G.: Dataset: We Do Not Understand What It Says – Studying Student Perceptions of Software Modelling (2022). DOI 10.5281/zenodo.6913780. URL `https://doi.org/10.5281/zenodo.6913780`
8. Ciccozzi, F., Taentzer, G., Vallecillo, A., Wimmer, M., Famelis, M., Lambers, L., Mosser, S., Paige, R., Pierantonio, A., Rensink, A., Salay, R.: 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, pp. 122–129 (2018)
9. Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., Oivo, M.: Empirical software engineering experts on the use of students and professionals in experiments. Empirical Software Engineering **23** (2018). DOI 10.1007/s10664-017-9523-3
10. Forward, A., Badreddin, O., Lethbridge, T.C.: 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) (2010)
11. Garzón, M.A., Aljamaan, H., Lethbridge, T.C.: Umple: A framework for model driven development of object-oriented systems. In: 2015 ieee 22nd international conference on software analysis, evolution, and reengineering (saner), pp. 494–498. IEEE (2015)
12. Gilson, F.: 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, pp. 98–105 (2018)
13. Gonnord, L., Mosser, S.: 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, pp. 106–113 (2018)
14. Gorschek, T., Tempero, E., Angelis, L.: On the use of software design models in software development practice: An empirical investigation. Journal of Systems and Software **95**, 176–193 (2014)

15. Hammouda, I., Burden, H., Heldal, R., Chaudron, M.R.: Case tools versus pencil and paper. In: ACM/IEEE 17th Int. Conf. on Model Driven Engineering Languages and Systems–Educators Symposium (2014)
16. Hiller, J.: Epistemological foundations of objectivist and interpretivist research. Barcelona Publishers (2016)
17. Hutchinson, J., Rouncefield, M., Whittle, J.: Model-driven engineering practices in industry. In: 2011 33rd International Conference on Software Engineering (ICSE), pp. 633–642 (2011). DOI 10.1145/1985793.1985882
18. Hutchinson, J., Whittle, J., Rouncefield, M.: Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. Science of Computer Programming **89**, 144–161 (2014). DOI https://doi.org/10.1016/j.scico.2013.03.017. Special issue on Success Stories in Model Driven Engineering
19. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of mde in industry. In: 2011 33rd International Conference on Software Engineering (ICSE), pp. 471–480 (2011). DOI 10.1145/1985793.1985858
20. Kirstan, S., Zimmermann, J.: 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" (2010)
21. Kolovos, D.S., Cabot, J.: Towards a corpus of use-cases for model-driven engineering courses. In: EduSymp/OSS4MDE@ MoDELS, pp. 14–18 (2016)
22. Kopach-Konrad, R., Lawley, M., Criswell, M., Hasan, I., Chakraborty, S., Pekny, J., Doebbeling, B.N.: Applying systems engineering principles in improving health care delivery. Journal of general internal medicine **22**(3), 431–437 (2007)
23. Lethbridge, T.C., Mussbacher, G., Forward, A., Badreddin, O.: Teaching uml using umple: Applying model-oriented programming in the classroom. In: 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), pp. 421–428. IEEE (2011)
24. Liebel, G., Badreddin, O., Heldal, R.: Model driven software engineering in education: A multi-case study on perception of tools and uml. In: 2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T), pp. 124–133 (2017). DOI 10.1109/CSEET.2017.29
25. Liebel, G., Heldal, R., Steghöfer, J.P.: Impact of the use of industrial modelling tools on modelling education. In: 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), pp. 18–27 (2016). DOI 10.1109/CSEET.2016.18
26. Liebel, G., Marko, N., Tichy, M., Leitner, A., Hansson, J.: Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. Software and Systems Modeling **17**(1), 91–113 (2018). DOI 10.1007/s10270-016-0523-3
27. Liebel, G., Tichy, M., Knauss, E.: Use, potential, and showstoppers of models in automotive requirements engineering. Software and Systems Modeling (2018). DOI 10.1007/s10270-018-0683-4
28. Mohagheghi, P., Dehlen, V.: Where is the proof? - a review of experiences from applying mde in industry. In: I. Schieferdecker, A. Hartman (eds.) Model Driven Architecture - Foundations and Applications, *Lecture Notes in Computer Science*, vol. 5095, pp. 432–443. Springer Berlin Heidelberg (2008)
29. Mohagheghi, P., Gilani, W., Stefanescu, A., Fernandez, M.A., Nordmoen, B., Fritzsche, M.: Where does model-driven engineering help? experiences from three industrial cases. Software and Systems Modeling **12**(3), 619–639 (2013)
30. Moody, D.L.: The "physics" of notations: a scientific approach to designing visual notations in software engineering. In: 2010 ACM/IEEE 32nd International Conference on Software Engineering, vol. 2, pp. 485–486 (2010). DOI 10.1145/1810295.1810442
31. Oliveira, K., Regina, A., Rocha, C., Travassos, G., Menezes, C.: Using domain-knowledge in software development environments. Tech. rep., CiteSeerX (2000). URL `https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.2007`
32. Paige, R.F., Polack, F.A., Kolovos, D.S., Rose, L.M., Matragkas, N.D., Williams, J.R.: Bad modelling teaching practices. In: EduSymp@ MoDELS, pp. 1–12 (2014)
33. Pascale, C.M.: Cartographies of knowledge: Exploring qualitative epistemologies. Sage Publications (2010)

34. Petersen, K., Gencel, C.: Worldviews, research methods, and their relationship to validity in empirical software engineering research. In: 2013 joint conference of the 23rd international workshop on software measurement and the 8th international conference on software process and product measurement, pp. 81–89. IEEE (2013)
35. Raistrick, C.: Applying mda and uml in the development of a healthcare system. In: International Conference on the Unified Modeling Language, pp. 203–218. Springer (2004)
36. Reuter, R., Stark, T., Sedelmaier, Y., Landes, D., Mottok, J., Wolff, C.: Insights in students' problems during uml modeling. In: 2020 IEEE Global Engineering Education Conference (EDUCON), pp. 592–600 (2020). DOI 10.1109/EDUCON45650.2020.9125110
37. Rugaber, S.: The use of domain knowledge in program understanding. Annals of Software Engineering **9**(1), 143–192 (2000)
38. Runeson, P.: Using students as experiment subjects - an analysis on graduate and freshmen student data. Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering (2003)
39. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering – Guidelines and Examples. John Wiley & Sons Inc. (2012). DOI 10.1002/9781118181034
40. Saldaña, J.: The coding manual for qualitative researchers. SAGE Publications (2015)
41. Salman, I., Mısırlı, A.T., Juristo, N.: Are students representatives of professionals in software engineering experiments? In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 1, pp. 666–676 (2015). DOI 10.1109/ICSE.2015.82
42. Schmidt, A., Kimmig, D., Bittner, K., Dickerhof, M.: Teaching model-driven software development: Revealing the" great miracle" of code generation to students. In: Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148, pp. 97–104 (2014)
43. Siau, K., Loo, P.P.: Identifying difficulties in learning uml. Information Systems Management **23**(3), 43–51 (2006)
44. Sjöberg, D.I., Anda, B., Arisholm, E., Dyba, T., Jorgensen, M., Karahasanovic, A., Koren, E.F., Vokác, M.: Conducting realistic experiments in software engineering. In: Proceedings international symposium on empirical software engineering, pp. 17–26. IEEE (2002)
45. Stikkolorum, D.R., Ho-Quang, T., Chaudron, M.R.: Revealing students' uml class diagram modelling strategies with webuml and logviz. In: 2015 41st Euromicro Conference on Software Engineering and Advanced Applications, pp. 275–279 (2015). DOI 10.1109/SEAA.2015.77
46. Stol, K.J., Fitzgerald, B.: The abc of software engineering research. ACM Transactions on Software Engineering and Methodology (TOSEM) **27**(3), 1–51 (2018)
47. Störrle, H.: How are conceptual models used in industrial software development? a descriptive survey. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, pp. 160–169 (2017)
48. Torchiano, M., Tomassetti, F., Ricca, F., Tiso, A., Reggio, G.: Relevance, benefits, and problems of software modelling and model driven techniques—a survey in the italian industry. Journal of Systems and Software **86**(8), 2110–2126 (2013)
49. Walderhaug, S., Stav, E., Mikalsen, M.: Experiences from model-driven development of homecare services: Uml profiles and domain models. In: International Conference on Model Driven Engineering Languages and Systems, pp. 199–212. Springer (2008)
50. Westphal, B.: 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), pp. 690–699 (2019). DOI 10.1109/MODELS-C.2019.00105
51. Whittle, J., Hutchinson, J.: Mismatches between industry practice and teaching of model-driven software development. In: International Conference on Model Driven Engineering Languages and Systems, pp. 40–47. Springer (2011)
52. Whittle, J., Hutchinson, J., Rouncefield, M.: The state of practice in model-driven engineering. IEEE Software **31**(3), 79–85 (2014). DOI 10.1109/MS.2013.65

53. Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., Heldal, R.: Industrial adoption of model-driven engineering: Are the tools really the problem? In: International Conference on Model Driven Engineering Languages and Systems, pp. 1–17. Springer (2013)
54. Wohlin, C.: Case study research in software engineering—it is a case, and it is a study, but is it a case study? Information and Software Technology **133**, 106,514 (2021). DOI https://doi.org/10.1016/j.infsof.2021.106514

A Interview Guides

We present two interview guides in this section used for students and instructors. We scheduled 40 minutes for student interviews and 30 minutes for instructor interviews. Interviews were recorded with permission from the interviewees.

A.1 Student Interview Guide

The interview starts with a small introduction of 5 minutes presented by the interviewer which consists the following points:

- Asking permission to record the interview.
- Introduction of the interviewer.
- Explain the purpose of the study, the research questions.
- Explain the rules of the interview.

Following the introduction, the interview starts. It has two parts, first the *Introductory Questions*. The duration of this part is 10 minutes.

1. I1: Which degree are you currently pursuing? (in which major?)
2. I2: Have you worked in industry? If yes, what role/domain?
3. I3: Do you have any experience with modelling? What kind of experience do you have with modelling?
4. I4: Go in detail with the experience, ask about syntax, diagrams that they had used before.

In the second part, *Modelling Questions* we go in detail with the modelling challenges and the duration is a maximum of 25 minutes. Before going into the modelling questions, the interviewer explains software modelling and her research interest in software modelling. The reason is to make sure that the interviewee has a clear idea and can give concrete answers.

1. M1: What experience do you have in software modelling?
2. M2: What did you like from your experience?
3. M3: What challenges did you face during your experience?
4. M4: In your opinion what are the reasons behind these challenges?
5. (If needed): When you are using a modelling notation what typically troubles you?
6. M5: How was your experience with modelling tools? Tell me about the tools you have used in the past.
7. M6: What did you like and dislike about each of those tools?
8. M7: For which purposes/in which situations (in SE) do you think modelling is useful? When is this not the case?
9. M8: In your case can you tell me of an experience where you found modelling useful?
10. M9: What advantages and disadvantages do you see in modelling?
11. M10: Did you get enough knowledge about modelling from the courses you have taken on the topic?
12. M11: What was missing from those courses?
13. M12: How would the perfect modelling course look like?
14. M13: Do you have any comments or suggestions?

A.2 Instructor Interview Guide

1. Could you shortly describe which topics you cover in your course?
2. Regarding modelling, what are you covering in the course?
   (a) Are you teaching modelling notation (e.g., UML diagrams)?
   (b) Are you teaching how to draw those models (example: identify nouns/verbs in a text to draw class diagrams)?
   (c) Are you teaching how the models relate to code?
   (d) Are you teaching semantics of modelling notations?
3. Roughly how much time is spent on modelling in your course?
4. Regarding modelling, what is covered in assignments or exam?
5. How do you think students perceive the modelling part of the course?
6. What do you think are the things that students struggle with (with respect to modelling)?
7. What challenges are you facing when teaching modelling?
8. How do you think this could be improved?