Evaluating Software Modelling Recommendations: Towards Systematic Guidelines for Modelling

Shalini Chakraborty
shalini19@ru.is
Reykjavik University
Reykjavik, Iceland

Grischa Liebel
grischal@ru.is
Reykjavik University
Reykjavik, Iceland

ABSTRACT

Background: Despite having several advantages, software modelling remains unpopular for developers. Similarly, university students do not see the benefits of software modelling in the university curriculum. Prior research show the lack of guidance for students to do so.

Aims: We aim to evaluate the effectiveness of four modelling recommendations made in related work to improve student modelling knowledge. Additionally, we aim to discover students' perceptions of software modelling after taking a course with the recommendations included.

Method: We conducted a mixed method study, including interviews with teaching assistants, student surveys, and a focus group study involving students, teaching assistants, and experts from both modelling and education.

Results: We find that the four recommendations overall have a positive impact as they help students better understand the modelling knowledge from the course. Students express that specific recommendations help them grasp the concept of software modelling well. We also extend the recommendations by adding more details specific to software modelling and solidifying the recommendations into systematic guidelines.

Conclusions: The guidelines can potentially enhance education and training in software modelling, catering to both academic settings and industrial environments. Additionally, the guidelines contribute to improved communication between students and the course itself by outlining what students can expect from modelling assignments and the value inherent in each of these assignments.

CCS CONCEPTS

• **Software and its engineering → Unified Modeling Language (UML)**.

KEYWORDS

Software Modelling, UML, Education, Interview, Survey, Focus Group

ACM Reference Format:
Shalini Chakraborty and Grischa Liebel. 2024. Evaluating Software Modelling Recommendations: Towards Systematic Guidelines for Modelling .

ESEM '24, October 24–25, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-1047-6/24/10.
https://doi.org/10.1145/3674805.3686693

In *Proceedings of the 18th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '24), October 24–25, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3674805.3686693

1 INTRODUCTION

Software modelling has a significant influence on software development and product maintenance. Despite having multiple benefits [2, 15], software modelling remains unpopular among developers [12]. The picture is similar in education. Often while learning modelling, university students limit their modelling application to course assignments. Students recognise specific advantages of modelling, particularly in areas where informal models suffice, such as understanding a system overview or conceptual comprehension of the problem domain. However, most students still need to be convinced 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 [5, 20, 30].

In our previous case study [5], we present four recommendations for modelling education in universities after an extensive case study based on three universities. The recommendations are to be applied to modelling courses or courses where modelling is a sub-part. The recommendations included assignment structures, grading instructions and feedback suggestions. In this study, we aim to evaluate the four recommendations and examine students' perception of modelling after taking a course with the recommendations.

Therefore, to address this goal we have the following research questions (RQs):

- RQ1: How effective are the recommendations in improving students' software modelling skills?
- RQ2: What are students' perceptions of software modelling after the implementation of the recommendations in the courses?

We conducted a mixed-method empirical study to answer the RQs. To explore the effectiveness of the recommendations, we applied them in two 12-week bachelor courses related to software design. Afterwards, we continued the empirical study with a qualitative investigation. This involves gathering feedback from students (through surveys) and instructors (through interviews) regarding the impact of the applied modelling recommendations. To delve into students' actual understanding of modelling post-recommendation implementation, we analysed student assignments in addition to their feedback. This analysis focused on assessing the syntax and semantics of UML diagrams created as a part of their assignments, aiming to gain insights into students' understanding of modelling

concepts imparted through the courses. Finally, we conducted a focus group study with students, teaching assistants, and experts from both modelling and education to get early feedback on the overall impact of our recommendations.

Our findings show that the recommendations have a positive impact on modelling education. Students and teaching assistants expressed that specific recommendations helped them communicate better with each other and the course. Students expressed that the recommendations helped them become aware of the assignments' expectations, which allowed them to distribute effort into the assignment and take the knowledge of software modelling at their own pace. Although we observed a few challenges, such as a lack of semantic understanding of the modelling diagrams, students could grasp the modelling knowledge through familiar domains. Apart from the initial evaluation, we added specific details to the recommendations to solidify them into systematic guidelines, which will help students in the long run and better carry the knowledge from university into industry careers.

The rest of the paper is structured as follows: we discuss the related work in Section 2, Section 3 demonstrates our methodology and data analysis strategies. We report the results of our study in Section 4, followed by discussions in Section 5 conclusion in Section 6.

2 RELATED WORK

In the following, we describe related work that discusses research on modelling education, especially how to teach modelling and students' role in empirical studies.

2.1 Research on Modelling Education

Modelling and model-based software development represent crucial components of software engineering [6, 33]. There exists a substantial number of literature describing the inclusion of modelling and modelling languages like UML into general software engineering or software design courses [7, 10, 18, 25, 32, 33].

Rapos [25] describes a course based on model-driven software engineering (MDSE) with a higher emphasis on lab performance and assignments based on students' feedback. Gonnord et al. [11] guide students from low-level C code to crafting a modelling language workbench. Westphal [33] outlines the structure of an SE course heavily centred on modelling. Gilson et al.[9], advocate for a course where students serve both as language designers and users to assess the usability of software language engineering (SLE). Vallino [32] propose and describe a course where they claim that should be emphasized throughout the curriculum. In each instance, student feedback forms the cornerstone of evaluating the proposed course design.

Several empirical studies have discussed modelling and especially UML in education [1, 5, 13, 21, 26, 30]. Agner et al. [1] conducted a survey on the utilisation of modelling tools among 117 students. Their findings highlighted issues such as inadequate feedback, challenges in diagram creation, and tool complexity. Hammouda et al. [13] compared the usage of modelling tools to traditional pen-and-paper methods. Through a survey, they assessed students' perceptions of the differences, ultimately finding no distinct advantage for either approach. Reuter et al. [26] investigated students' challenges with UML diagrams across two modelling courses. In a similar vein, Stikkolorum et al. [30] explored student difficulties and modelling approaches in UML Class diagrams by providing students with a specialised UML editor equipped with feedback mechanisms.

2.2 Students' Role in Empirical Studies

Empirical studies with students, especially in the are of software modelling is not uncommon. Pourali et al. [24] describes a two-phase user study aimed at identifying the primary challenges users encounter while developing UML Class and State-Machine diagrams using UML modelling tools. Badreddin et al. [4] explore the impact of education on students' perceptions of modelling by conducting a survey across three countries. Kuzniarz et al. [19] examine the use of stereotypes examples to enhance the understanding of UML models. Kutar et al. [17] perform an empirical study to assess whether users' comprehension of information in Sequence diagrams differs from that in Collaboration diagrams. Chakraborty and Liebel [5] conduct a multiple case study to explore students' perceptions of software modelling. The study involve conducting 21 interviews with students and instructors across five courses at three universities.

3 METHOD

In this empirical study, we engaged with university students to evaluate the efficacy of four modelling recommendations introduced during their bachelor courses.

3.1 Recommendations

In our previous case study [5], we engaged in 16 interviews with students and conducted 5 interviews with instructors from three different universities to explore students' perceived understanding of modelling and the challenges they encountered.

After studying the benefits and challenges of students, we present four recommendations for modelling education. Below, we discuss the four recommendations and our explanations behind them.

R1 Iterative Projects with Regular Feedback: We recommend iterative assignments/projects to help students practice what they have learned. Along with that, instructors must provide regular feedback. Regular feedback is consistent, timely, specific and contains assessment details of every assignment students were instructed to do. Combining iterative projects and regular feedback can make learning more efficient as students get to implement the feedback during the course time.

R2 Limitation of diagram types: UML constitutes a vast and intricate domain, presenting a substantial challenge for learners. To enhance the learning experience, we advocate for a more realistic approach that acknowledges the limitations of students' time and cognitive capacity. Hence, we propose narrowing down the array of diagram types covered in a course. By being more realistic about what students can feasibly learn within a limited time frame, we aim to prevent overwhelming them with an extensive variety of diagrams. In our study[5], we encounter the challenge of "unclear expectations," where students struggle to align a given problem with the appropriate modelling diagram. We believe that reducing the variety of diagrams becomes crucial to address that challenge,

allowing students to delve deeper into the intricacies of each diagram type and fostering a clearer understanding of their respective purposes.

R3 Grading Rubrics: Along with *irregular feedback*, we observe confusion regarding assessment. Irregular feedback refers to feedback that is provided inconsistently. In our case study, we noticed that students received feedback at varying times rather than in a systematic or scheduled manner. This lead to confusion, uncertainty, and difficulty in understanding how to improve or progress, as they may not know when to expect feedback or may receive it too late to make effective changes. Unlike coding, the instructors must provide students with constructive and regular feedback, as modelling doesn't offer runtime errors as programming. At least not in the early stages when students are not generating code from the models. Therefore, we recommend having grading rubrics for assignments where stated model purposes and grading are aligned, and aligning that with each task is very important for the student's understanding. In the rubric, assignments are broken down into tasks, and priorities and grades for every task are mentioned. Grading rubrics foster better communication between students and instructors, allowing instructors to provide constructive feedback regularly. A sample rubric is attached as a supplementary file[1]

R4 Use of Familiar Problem Domains: Finally, we discussed the challenge of the *absence of expertise* in the specific problem domain. To overcome this hurdle, we recommend educators consider employing familiar problem domains, perhaps ones selected by the students.

3.2 Study Design

Empirical evidence holds significant importance in evaluating various aspects as it provides tangible data and observations derived from real-world experiences or experiments. We selected qualitative analysis of the empirical evidence [28]. Qualitative research methods were primarily developed by educational researchers and other social scientists to delve into the intricacies of human behaviour, focusing more on words than numbers [8, 22]. In our context, qualitative analysis was apt as we were assessing recommendations intended for human application in the creative task of modelling. We incorporated the recommendations in two university courses, focusing on software design and emphasising UML modelling. One of the authors assumed the role of instructor in both instances. We integrated a combination of the suggested recommendations to optimise the learning experience. Both courses were part of the same university and designed for bachelor-level students. For the data collection process, we gathered students' assignments along with their questions throughout the assignment phases. Post-course, we conducted a survey to gain insights into their overall experience. We interviewed teaching assistants (TAs) for their perspectives on the recommendations and overall student participation. We used informed consent to gather all data. Students signed a consent form, consenting to grant access to their assignments and grading information for this study. The post course survey was voluntary for students. We also obtained informed consents from TAs for the interviews. In both surveys and interviews, participants had the right to withdraw at any moment, and they also retained the right to withdraw their consent at any time.

In the next sections, we explain the courses in more detail and show how we used the recommendations in our teaching approach.

C1: Software Analysis and Design The course is given to first-year bachelor students in Computer Science. The curriculum broadly addresses aspects like software prototype design, user interfaces, requirement analysis, and UML modelling. Specifically, the first author took charge of the UML section, covering 30 percent of the entire course. We crafted the entire syllabus for UML, including assignments and grading criteria. In total, the course accommodated 259 students, out of which 50 students granted permission for us to use their assignments via signed consent forms. Additionally, 30 students out of 259 provided feedback through a post-course survey. The survey was anonymous.

We incorporated R1, R2, and R3 in the course design. For R1, we had trouble providing students with regular feedback because it was an extensive course, and we had six different TAs (teaching assistants). As the TAs are responsible for grading assignments and giving feedback, we observed a few communication issues.

We did not select the problem domain for this course, as another instructor started the course with modelling coming later in the course. We aimed for continuity in student engagement with the initial problem domain (following R1), so we adhered to the same domain throughout and tailored assignments accordingly. The domain was Icelandic tourism, and students were working to make software that provides tourists with information about tourist places in Iceland, their availability, weather forecasts and facilities around these places for efficient visits.

C2: System Analysis and System Design The course is given to first-year bachelor students in Software Engineering, placing a heightened emphasis on modelling compared to the previous course. The syllabus encompasses a range of topics, including requirement engineering, diverse interface design methodologies, UML modelling covering 6 types of UML diagrams (use case, class, sequence, activity, state, and component diagrams) and their practical application, and testing of the designs. The first author was in charge of the complete course. Of the 37 enrolled students, 35 granted us permission to use their assignments via consent forms and 31 students contributed to the post-course anonymous survey.

We incorporated all four recommendations in the course design. As a problem domain, we surveyed pre-course to know what students would like to work on and finalised four options from their answers:

StudyBuddyConnect, a free online platform exclusively for university students, where they can enrol, register qualifications, and find study partners.

IcelandThrillQuest, a platform for discovering thrilling physical activities in Iceland.

MoodMusic a system to define tracks one loves and emotions one feels when listening.

DiverseConnection, a nonprofit digital story-sharing platform fostering inclusivity and creativity where individuals of all genders, races, sexes and backgrounds can unite to share their passion for art, music, and dance.

We opted for both of these courses due to the shared instructor, with the same author teaching both. Additionally, both courses

[1] https://doi.org/10.5281/zenodo.11125637

feature modelling in their syllabus, providing a convenient opportunity for us to implement the recommendations. The timing of these courses further worked to our advantage, given that they were scheduled consecutively—C1 in the fall semester (2023) and C2 in the spring (2024).

Table 1 describes the detailed actions we took to address the four recommendations.

In addition to the mentioned actions, we granted students the freedom to select their preferred tools. We firmly believe that this freedom carries its benefits, increasing the likelihood of students encountering a positive initial modelling experience rather than becoming frustrated with tool-related challenges.

3.3 Data Collection and Data Analysis

We collected three types of data from both courses[2]: *student feedback*, *instructor's feedback regarding grading the assignments*, and finally, the *assignments performed by the students*.

Student feedback: For the first data source, *Student feedback*, we aimed to understand how students perceive the newly designed UML lectures and assignments. We surveyed at the end of the course. For C1, the survey was voluntary, and we received 30 survey participants. For C2, we made the survey the last assignment of the course, carrying 5 per cent of the final grade. We received 31 responses. The survey included questions about their learning (challenges and benefits), given materials, applied recommendations, specific diagrams, etc. For C2, we kept the same questions as C1, just added two more questions as we applied all 4 recommendations in C2[3].

Instructor's feedback: For the second data source, *instructor's feedback*, we interviewed teaching assistants (TAs) from both courses[4]. TAs are responsible for interacting with students during the problem-solving sessions[5] and grading the assignments. We interviewed 4 TAs from C1. Of the four TAs, two were involved with the course in the previous academic year (fall semester, 2022), and thus, we asked them a few additional questions to get their opinion for both academic years. We aimed to compare both years and evaluate the effectiveness of our modelling recommendations.

For C2, we had one TA. This person was a TA in C1 as well. We had a slightly different interview guide for C2 due to the different nature and types of assignments.

The instructors are identified as TA_number_course, for example: TA_1_C1, TA_2_C1, TA_3_C1, TA_4_C1 and TA_C2.

Assignments: We structured the assignments to allow students to practice modelling twice. Initially, in both courses, students completed small assignments (typically carrying 2 percent and 5 percent of the overall grade) to draw the diagrams for the first time. Subsequently, we introduced group assignments where students could read diagrams from their peers, share ideas within their groups, and then redraw the diagrams. This collaborative approach helped students gain a deeper understanding of the problem domain and enhanced their diagramming skills.

We gathered the small assignments and group assignments related to UML diagrams from students. In total, we compiled 3 small assignments and 1 group assignment from C1 (from 50 students), as well as 4 small assignments and 2 group assignments from C2 (from 35 students).

Additionally, we conducted a focus group [16] session to evaluate the recommendations. Prior to recording the focus group study, we obtained consent from all participants. Moreover, students received a movie voucher for their participation in the focus group study. The recommendations mentioned in [5] are in the early stages of development. Therefore, employing focus groups is helpful, as they provide an excellent opportunity to gather initial feedback. The specific objective is to evaluate the recommendations within a defined context and explore participants' opinions and experiences regarding their use.

- **Selecting Participants**: We chose 6 participants for the focus group, representing various aspects and experience levels. This included 2 students, one TA (who was a TA in both C1 and C2), one presenter and host of the focus group, one expert in software modelling, and finally, one representative from the education sector.
- **Planning and Conducting the Session**: The session was planned for 1 hour. The host started the session by presenting the research and recommendations. Each participant then presented their opinions on them. The host also acted as a facilitator. Their role during the sessions involved:
  (1) Ensuring that crucial topics were addressed.
  (2) Keeping discussions on track.
  (3) Prompting participants to share their viewpoints.
  However, the facilitator refrained from actively participating in the discussion. Alongside facilitating, they were responsible for taking notes during the discussions. The discussion was semi-structured. There were specific themes. However, the themes helped direct the conversation without constraining participants from having an open discussion.

For data analysis, we chose three approaches. Firstly, we conducted in-vivo coding to analyze the feedback received from both students and instructors. Secondly, for the assignments, we developed an analysis protocol: Diagram Assessment Sheet or DAS. Following this protocol, we thoroughly examined all the assignment submissions, which included diagrams drawn by the students. Finally, for the focus group, both researchers opted for open code analysis.

In-vivo Coding: In-vivo coding, as described in [27], is utilized as one of the initial cycle coding methods for qualitative analysis. It involves directly using the literal spoken words as codes, rather than creating one's own terms (open coding).

Initially, we transcribed all the interview data from C1, which included interviews with the four TAs. Subsequently, both researchers independently coded one interview each, followed by a discussion of the outcomes. Once we reached an agreement, we proceeded to code the remaining interviews. We repeated the same process for

[2]We asked for students' consent in the beginning and collected data only for those students who agreed to our data collection. 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) and the right to publish the anonymised transcripts.
[3]Survey questions are added to: https://doi.org/10.5281/zenodo.11125637
[4]The interview guide and transcripts are added to: https://doi.org/10.5281/zenodo.11125637
[5]Students perform the assignments in problem-solving sessions. In our case, students solve the individual assignments in problem-solving sessions and work on the group assignments.

Recommendations	Actions taken in Courses
R1	The course project was divided into two forms: individual and group assignments. Assignments were designed to allow students to apply what they learned in individual tasks to group projects. This approach enables students to analyse and implement the feedback received on individual assignments in their group work.
R2	We narrowed down the syllabus to focus on a few types of diagrams. Due to time constraints in C1, we concentrated on just three diagram types and created assignments based on them. In C2, we extended the learning period to four weeks, covering six diagram types. Students were given the opportunity to complete four individual assignments and two group assignments related to these six diagram types (In C1, students performed 3 individual assignment and 2 group assignment). In both courses, individual assignments were used for students to initially learn and practice a specific type of diagram. In the group assignments, students were required to model the diagrams they had learned in the individual assignments, but this time collaboratively. This approach allowed students to apply the feedback they received on a particular diagram, work with peers, and engage in collaborative modelling. The problem domain remained consistent across all assignments.
R3	Grading rubrics were employed for both individual and group assignments. In both courses, the instructor introduced grading rubrics in the first lecture and explained how they would work throughout the course. The grading system was structured to align with specific tasks, ensuring that students comprehended the significance of each step in the assignments. Feedback corresponded to the rubric, facilitating easy and systematic analysis for students. We published rubrics at the same time as the assignments. Hence, students knew the evaluation criteria before modelling.
R4	In C1, we couldn't incorporate R4. Nevertheless, we made sure that all modelling assignments revolved around the same problem students had been working on since the start. In C2, we encouraged them to propose their own domains. This approach aimed to actively involve students in the process and enhance their sense of inclusion. We also ensured the students worked on the same domain for the entirety of the course to increase their understanding and comfort.

Table 1: Various actions taken in both courses (C1 and C2) based on four recommendations (R1, R2, R3 and R4)

all 30 student surveys for C1 and the interviews and surveys from C2.

Diagram Assessment Sheet or DAS: We developed an assessment strategy called the Diagram Assessment Sheet (DAS) to assess students' understanding of UML diagrams in the course based on their created models/UML diagrams. DAS evaluates two key aspects of a UML diagram: Syntax and Semantics. It's important to note that the DAS is aligned with the proposed recommendations in [5] and the way modelling was taught in those courses, as outlined in Section 3.1. Therefore, we are assessing students' comprehension of modelling within the framework of the recommendations. Our goal is to evaluate the students' overall knowledge of software modelling based on the recommendations.

The assessment sheet includes students' data for each of their modelling assignments. Each assignment is accompanied by syntax and semantics columns, where the syntax column employs a numbering system to denote incorrect notations with a (-1) marker. In contrast, the semantics column adopts an open coding approach, documenting all semantic issues encountered in the diagrams. Subsequently, both authors employed DAS to analyse two students' assignments individually. Following this initial analysis, the authors discussed their findings, refined DAS and conducted further analysis on an additional ten students' assignments. Once a consensus was reached on the analysis strategies within DAS, the analysis was extended to encompass all 50 students' assignments from C1 and 35 students from C2.

Focus Group Analysis: We recorded the focus group session, transcribed the data and put them into an Excel sheet[6]. We then separated research explanations, clarification points and other informal/non-related communications from the data. We categorised the rest of the communication into three different aspects: benefits of the recommendations, challenges of the recommendations and modifications of the recommendations.

3.4 Validity Threats

We conducted a mixed-method empirical study, primarily collecting data through interviews, surveys and a focus group study. Afterwards, we took an interpretive approach [14, 23] and analysed the data via qualitative measures. An interpretivist approach implies that humans construct knowledge as they interpret their experiences of and within the world. We opted for interpretivism to evaluate the recommendations as it was essential to understand our study participants' experience with modelling recommendations and their understanding of modelling afterwards. A potential threat to our study is that the four recommendations were applied in parts of two courses. As the courses have more to offer, the other parts (such as non-modelling theory/assignments) influence students' experience of the recommendations, which leads to the perception of modelling. To address this issue, we gathered students' opinions and real experiences through surveys and assignments. We also

[6]https://doi.org/10.5281/zenodo.11125637

supplemented the data with the perspectives of teachers by interviewing TAs. Additionally, we held a focus group study to obtain more targeted feedback on the recommendations from students, TAs, and experts in modelling and education. We acknowledge that our data set consists of just two university courses. We tried to mitigate that by selecting two courses from two bachelor fields (Computer science and software engineering).

To ensure the credibility of our findings, we based our analysis on in-vivo codes taken directly from the exact words people said in the interviews. Both authors did the coding. We also had a one-hour focus group discussion to hear different opinions about recommendations, where we included experts in education and modelling who were not part of any of the two courses. Moreover, in a commitment to transparency, we have made anonymised transcripts available for interviewees who granted consent.

One of the authors served as the modelling instructor in both courses. There is a potential for researcher bias when teaching modelling through the recommendations. However, we address this concern by involving TAs in grading assignments. Additionally, TAs were responsible for practical sessions. For instance, R3 (grading rubrics) were employed by students and TAs without any input from the instructor.

In C1, we had over 280 students and seven TAs grading the entire course. The modelling segment occurred towards the end of the 12-week course. With previous assignments accumulating, consistent regular feedback for students was challenging to provide. However, students received consistent feedback for the initial two assignments. When they finally received feedback for the last two assignments, they genuinely appreciated its positive impact. In their post-course survey, they emphasised the importance of regular feedback and how it aided their learning.

4 RESULTS

In the following subsections, we present the answers of two RQs with the results of study.

4.1 RQ1: How effective are the recommendations in improving students' software modelling skills?

Our study indicates that the four recommendations are successful in improving students' overall modelling skill. The recommendations are practical as students showed more engagement in assignments, and the communication was significantly better between students and instructors. In the following, we present each recommendation and the evaluation of those based on the study results.

4.1.1 R1 Iterative Projects with Regular Feedback: We observed positive feedback for R1. All four TAs expressed that students were struggling with syntax. Since this was the students' first attempt at drawing UML diagrams, we anticipated some initial difficulty with syntax. However, thanks to R1, students were able to redraw the diagrams using the same problem domain.

> *"I think a lot of them were handing in diagrams that weren't following the syntax. So I think they struggle with understanding the syntax or using the syntax in the first place. They were sometimes using the syntax wrongly, but insisting on using it nonetheless. It got better in group assignments, at least they were asking more specific question with their mistakes."* — TA_1_C1

Upon finishing the course, students came to appreciate learning the syntax. When asked, *"What aspects of the UML modelling part did you find most beneficial to your learning?"* . The majority of students responded learning the syntax was the most valuable aspect.

> *"The most beneficial aspect to learning it was certainly its logical and strict syntax."* — Student_C1

> *"I think the syntax and repeat them again in group assignments. I understood better after that. "* — Student_C1

We asked students, *On a scale of 1 to 5 (where 5 is the highest), how helpful was it to repeat the diagrams in the assignments?*. 60 percent of students rated 5, 30 percent of students rated 4 (see Figure 1) in C1. In C2, 51 percent rated 5 and 19 percent rated 4 (see Figure 2)

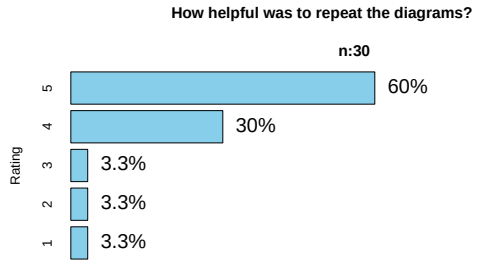

Figure 1: Rating from students of C1: how helpful was it to repeat the diagrams in the assignments

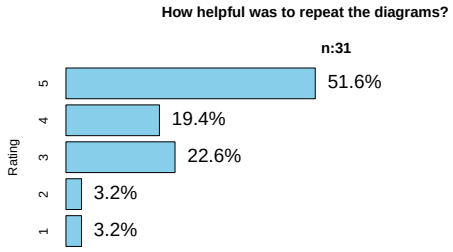

Figure 2: Rating from students of C2: how helpful was it to repeat the diagrams in the assignments

4.1.2 R2 Limitation of diagram types: In C1, due to limited time, we could only include three types of UML diagrams. In C2, we included six types of UML diagrams. We asked students, *On a scale of 1 to 5 (where 5 is the highest), how satisfactory do you think the distribution of UML modelling instruction was?* 45 percent students rated 5 and 41 percent students rated 4 (see Figure 3).

We also asked students of C2, *How do you feel about the time given to learn six diagrams: was it enough for you to understand them well, or do you feel like you needed more time?* Most students mentioned that the time distribution regarding the amount of modelling syllabus was enough for them to get a basic understanding.

> *"I believe that I have acquired a basic understanding of the diagrams that I will build upon in the future through experience."* — Student_C2

> *"But it was enough to get a feeling for each one, so that the next time I see one I at least know what I'm looking at and know what it's telling me."* — Student_C2

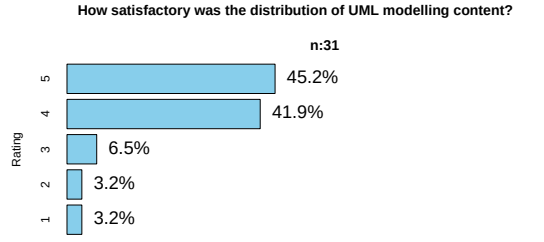

Figure 3: Rating from students of C2: how satisfactory do you think the distribution of UML modelling instruction was

4.1.3 R3 Grading Rubrics. To assess R3, we asked questions to both TAs and students regarding the rubric provided with each assignment. The responses from TAs were overwhelmingly positive, indicating that the rubric aided their comprehension of the grading criteria.

> *"But the rubric was clear. Yeah, okay. I remember it was like, you know, if this is missing the textual mark, if this is missing the text. Yeah, it was very clear in the UML."* — TA_2_C1

We also noticed that the rubric facilitated communication among the TAs. In C1, the rubric helped the six TAs align their grading expectations and ensure consistency.

> *"The rubric made it faster and those meetings [grading meetings among TAs] it was easier to discuss disagreements referring to the instructions"* — TA_4_C1

As modelling involves a level of subjectivity, the rubric may not always offer complete information. Nonetheless, it served as a fundamental guide for TAs to facilitate fair grading.

> *"But then again, like. These types of diagrams have an artistic aspect to them, so it's hard to define a universal rubric. But in general, yeah, like the rubrics had the, the main components and which. Had a direct relation with the assignment description. Please do five of these things. And then like the rubric said one point for each thing four points would be five points. Boom. You can't get any more simple than that. But any ambiguity there is only due to the students' work, not the rubric itself"* — TA_3_C1

For students, the rubric clarified the expectations for each assignment and provided a level of detail that assisted them in prioritising tasks. Additionally, the rubric facilitated feedback, enabling students to formulate questions about its contents. We asked students, *On a scale of 1 to 5 (where 5 is the highest), how helpful were the assignment and grading instructions?* In C1, 63 percent and in C2, 55 percent of students rated 5(see Figure 4 and Figure 5).

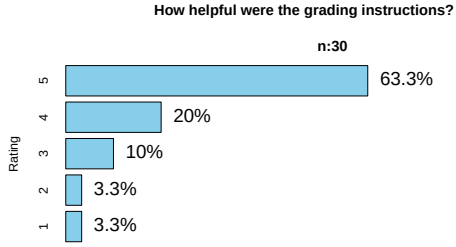

Figure 4: Rating from students of C1: how helpful were the assignment and grading instructions

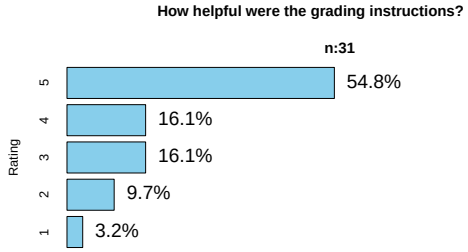

Figure 5: Rating from students of C2: how helpful were the assignment and grading instructions

However, the rubric was not entirely effective when it comes to students assessing their own modelling skills. In the focus group, a student mentioned,

> *"I think it's good for the assignment, but I don't know if you can, like, apply it elsewhere. I think if we don't have the grading rubric, we would probably go a lot more into detail, because we're just doing what's on the grading rubric"* — Student in the focus group

We understood that the rubric served the technical aspects of modelling, which was needed as students were new to modelling and required a mechanical understanding of each UML diagram. However, we felt there is a need to upgrade the rubric so students can use it further to assess their overall modelling skills after the course, along with realistic expectations, which are not limited to modelling assignments but modelling in general.

4.1.4 R4 Use of Familiar Problem Domains. Our findings show that students prefer working with the same problem domain throughout their course. When asked how helpful working with the same problem domain was for all the assignments, students answered positively.

> *"it allowed us to go through the back end development process from gathering requirements to implementing the requirements. It gave us a inside into how the process can be for us in the future"* — Student_C2

Students grew familiar with the domain as the course progressed.

> *"Always when I read the assignment description, I thought to myself how I could implement it with my problem domain."* — Student_C2

Working with the same problem domain help students focus more on UML.

> *"Using the same problem domain highlights the differences between diagrams and such."* — Student_C1
>
> *"That way we could focus better on the actual UML rather than the problem domain"* — Student_C2

4.2 RQ2: What are students' perceptions of software modelling after the implementation of the recommendations in the courses?

We asked students about their overall modelling experience after the course and analysed their assignments to present students' perception of modelling.

4.2.1 Benefits of Modelling. We asked students, *What aspects of the UML modelling part did you find most beneficial to your learning?* Most students mentioned that they found the modelling process and how it can be used in planning and designing software to be most helpful for learning.

> *"Different ways to see a system, helps in planning"* — Student_C1
>
> *"I think all of it. Learning how systems are modelled and designed for the back end"* — Student_C2

Additionally, students found the visual appeal of modelling beneficial for their learning.

> *"Besides understanding how helpful it is for future projects as a student and/or software engineer, I think it also helped me visualise software better and understand how it communicates with itself [different classes] in the big picture"* — Student_C2
>
> *"It was the different ways you build a system and the visual aspect of it"* — Student_C2

Students find that different diagrams provide them with different aspects of software development.

> *"I found learning how to create diagrams for user experience be very beneficial to understanding how software works with its users"* — Student_C1
>
> *"It was fun to connect programming into the diagrams like the class diagram. It was a cool way to see how you could plan your program ahead of time and make a cool diagram so that other people can understand your ideas"* — Student_C1
>
> *"One of the most important UML modelling that I studied in my opinion is the Use case diagram, it gives us an idea on how the system should work which is beneficial when designing a software"* — Student_C2

Further, we asked students, *which specific topics or diagrams within UML did you find most interesting?* to see their preferences and areas of curiosity. We hoped these answers would help tailor students' modelling experiences, leading to higher engagement in the future. The majority of the students mentioned the class diagram to be the most interesting. Interestingly, they provided various reasons for this:

> *"I found the class diagram the most interesting because it gave the best idea of how the program should be programmed. It gave a good visual representation of how the program should function and work"* — Student_C1
>
> *"Most definitely the class diagram. It think it will be very beneficial to know how it works. And to iterate through the process of making a system to update the requirements"* — Students_C2
>
> *"Class diagram, because in the class diagram you have to implement all the functions for each class"* — Student_C2

After class diagram, students find sequence diagram interesting.

> *"The sequence diagrams were really interesting; it was interesting to visualize how the user interacts with the system and what could go wrong"* — Student_C2

We noted a trend towards a more elaborate responses from the students. Notably, students provided more comprehensive insights into individual diagrams. Furthermore, there was an enhanced recognition of modelling as a thinking tool and how it contributes to long-term proficiency and its application in software development.

> *"Different ways to think about the problem and modelling them likewise"* — Student_C1
>
> *"I think the most beneficial aspect of what I learned from UML modelling is to think more like how a software engineer. When developing a software you have to think of every aspect and constantly plan for everything by making diagrams, unit testing and also get to now the target audience better"* — Student_C2

4.2.2 Challenges of Modelling. We asked students, *what challenges or difficulties have you encountered while studying UML?*

Many students were overwhelmed with the UML notation and its vast complexity.

> *"There is no one right way of executing the development process and we have to find what method is best for us"* — Student_C2
>
> *"Understanding the different rules, like the dotted lines and all of those small parts"* — Student_C1
>
> *"There seems to be different types of "syntax". Maybe I was looking at the wrong things, but when looking up images of diagrams they often looked different to each other"* — Student_C2

We noticed several syntactical and semantic mistakes in students' assignments. While syntactical errors were somewhat reduced in the group assignments, students still struggled with semantics. As students were free to choose any tool they preferred, we received varying layouts for the same problem and its solution model. Nonetheless, we observed that students had difficulty with open-ended problems where they needed to select elements for the model, such as different classes for a class diagram or various types of validation required to perform a basic registration service through a sequence diagram. We realised that students needed support in understanding the appropriate level of detail to include in a model. Students raised specific concerns about how much information a model should contain.

> *"Sometimes it was difficult to realise how detailed the models were supposed to be, that was the hardest part"* — Student_C2

5 DISCUSSION

Our research findings highlight the positive impact of the four recommendations on software modelling practices within the university environment. However, despite the assistance provided by these recommendations, challenges persist regarding students' overall grasp of software modelling, particularly in terms of applying this knowledge beyond the confines of the course. It is crucial to explore what students are modelling and what knowledge they acquire from these courses in light of the implemented recommendations.

5.1 What do students model?

Giving students multiple assignments with different diagram types is useful, but stating a clear purpose is necessary for them to understand modelling. Student assignments show that students need help understanding the correctness of a diagram. Students emphasise syntax, which is concerning as most of the modelling experts argue the importance of semantics over syntax. Two factors triggered this aspect among students. First, students perceive correctness primarily as syntactical correctness. Second, the concept of semantics remained unclear to them.

> *"How to use the notations correctly based on our vision of the program because I feel like it is so easy to make a mistake. "* — Student_C1

However, students need help understanding a diagram's semantics regarding a problem description.

> *"The difficulties I have encountered is learning how to understand the analysis of the diagram and try to figure out if you have gather sufficient information. "* — Student_C1

Students often struggle with *unclear expectations* in modelling courses [5]. This arises firstly from students encountering modelling diagrams for the first time and secondly from the overwhelming nature of UML. Learning and practising diagrams within a limited time frame can only provide limited assistance to students in mastering modelling. Since the syntax of UML is complex, their learning tends to be limited to these concerns, not moving beyond mastery of syntax.

In analysing assignments from both C1 and C2, we observed common syntactical mistakes. For instance, incorrect associations between classes were made by 95 percent of students in C1 and

94 percent in C2 during their initial attempts at drawing class diagrams[7]. Similarly, mismatches between direct and return messages in sequence diagrams were seen with 73 percent of students in C1 and 68 percent in C2 making this mistake in their first attempts[8]. While syntax improvements were seen in the second attempts at drawing these diagrams, snytactical issues persisted even in subsequent attempts.

In both C1 and C2, during the group assignments on modelling, students were required to review each other's initial diagrams, received feedback, and then collaboratively drew a second diagram. This approach helped their comprehension of various diagrams and their basic notations. However, as the diagrams all addressed to the same problem domain, students were unable to understand different semantics from other examples. In a focus group study, a student mentioned:

> *"I don't think we ever saw like an actual, fully-formed diagram or something, like, for another problem domain"* — Student in the focus group

Even though students got the syntax and semantics right, sometimes, understanding what should be part of the diagram and what should be beyond the scope was difficult.

> *"I'm not sure how detail you guys want us to go, and, for example, we were doing a sequence diagram recently with a group assignment, and our group was really overthinking the sequence diagram, and we were going into a lot of details when we shouldn't have."* — Student in the focus group

With the grading rubric, we managed to break down the assignments into tasks, aligned purposes with each task and guided students on how to complete a modelling task. However, modelling is a creative process, and it is crucial for students to grasp what to model independently, albeit with the aid of guidelines. Consequently, we adjusted our recommendation R1 by incorporating diagram interpretation. We discuss this further in Section 5.3.

5.2 From university to industry: *How do we carry the knowledge?*

Students expressed that modelling helps them in various ways:

> *"Overall, I thought it was interesting to see these methods for modelling a system in different ways. I hadn't really given much thought to how systems should be mapped out before implementing them, but it's obviously crucial in any practical project"* — Student_C1

> *"I find that all of the diagrams that we have learned to make have helped me gain a better understanding of what it takes to make a software, and great to learn everything for the future "* — Student_C2

> *"I think the most beneficial aspect of what I learned from UML modelling is to think more like how a software engineer. When developing a software you have to think of every aspect and constantly plan for everything by making diagrams, unit-testing and also get to now the target audience better"* — Student_C2

However, several students expressed their doubts as well:

> *"if you look beyond the course, if you would apply this somewhere else now, either in a course or, you know, out in industry, do you feel like it would still be something that would help you to roughly know what you should do?"* — Student in the focus group

The primary challenge identified from our data is guiding students in the right direction and ensuring they retain their modelling knowledge for their future careers. Empirical data shows that lack of guidance or training poses a significant obstacle to the adoption of modelling in industry [12, 21, 31]. While the recommendations prove beneficial to students during their university studies, further guidance is needed to bridge the gap between academia and industry, especially in terms of training. Three aspects pose the greatest threat here: challenges regarding modelling tools, lack of realistic modelling problems, and training in courses similar to the industry.

Students are able to use modelling tools, but need substantial amount of training to understand the differences between tools and produce models efficiently [21]. In university curricula, students are often given the freedom to select their modelling tool, which is good as they can select a simple one. However, it has drawbacks as modelling tools often have different user interfaces and use different sub-sets of the UML (correctly or not), which need to be clarified.

> *"Because like we have, tools and the are four different types of dotted lines and you have like 16 different arrows, some filled, some are not, some are open other than not"* — Student in the focus group

Whittle and Hutchinson [34] find various mismatches between industry and educational modelling practice. The study reveals that, while modelling in education is very UML-centric, industry increasingly prioritises other notations, particularly at the meta-modelling level with bespoke domain-specific languages and code generators. In their study, Akundi et al. [3] conducted a two-phase study to check to which extent model-based systems engineering (MBSE) academic curricula in the US reflect industry workforce hiring requirements. The study provided a pathway to align university curricula with industry needs and strategies to mitigate adoption challenges. While UML can serve as a tool for entry-level courses to modelling, eventually education might have to bridge to industry practice. This applies to handling various tools and notations, as mentioned above, but also to move beyond syntactical issues and use models purposefully to engineer software systems. Note that this requires a certain maturity, and is therefore not suited for entry-level courses, where students have little knowledge of software engineering as a whole.

5.3 Refining Modelling Education Guidelines

We evaluated four recommendations and investigated two research questions: the effectiveness of these recommendations in improving modelling skills and how they influence students' perceptions of modelling. Our study found that the recommendations effectively enhanced students' modelling skills and led to a better learning experience, with improved communication between students and instructors. As students' perceptions of modelling improved, we gained deeper insights into their challenges in learning modelling. Based on the analysis of the responses to both research questions, we refined our recommendations as follows:

R1 Iterative Projects with Regular Feedback: It is essential for students to engage in more reading of diagrams, not just of diagrams created by other student groups, but also of diagrams produced by professionals. It is important to discuss strengths and completeness of a diagram given a problem description, and to teach student to explore what makes a good diagram.

> *"If I show you a class diagram for code generation, it will look ridiculous because it's so detailed, because it needs to be very precise and technical. Whereas, you know, I've seen company diagrams on their architecture, which were literally four boxes with a bunch of random arrows between them"* — Modelling expert in the focus group

Therefore, we propose that, in addition to iterative projects, we should include diagram interpretation as part of the modelling

[7] Data sourced from TA feedback for the assignments
[8] Data collected from TA feedback

assignment. This would serve as an essential precursor, allowing students to grasp important concepts before delving into diagram creation themselves. By starting with interpretation, students can develop a deeper understanding of the purpose and significance of each diagram component. Furthermore, it enhances communication between instructors and students regarding feedback, establishing clear expectations not only for assignments but also for students' participation in any modelling activities. Specifically, we recommend the following three course elements:

- Iterative Projects: Structure modelling courses around iterative projects as a foundational element.
- Model Interpretation: Emphasise the importance of model interpretation as the initial step before commencing modelling activities.
- Consistent Feedback: Provide ongoing and consistent feedback for each stage of the modelling process to facilitate student learning and growth.

R2 Limitation of Diagram Types: We observe from our findings that limiting the diagram types helps students, especially first-year students. The essential thing here is to show students why those particular diagram types are selected. Previous research shows the connectivity between class diagram and object-oriented programming [29]. Similarly, we observe that students can see the connection between class diagrams and object-oriented programming concepts, helping them to bridge new concepts to exisiting knowledge in programming. Further, we encourage instructors to anchor diagram learning with more theoretical or technical concepts. For example, a student mentioned,

> *"I thought the sequence diagram was an interesting way to look at users interacting with the system."* — Student_C1

Therefore, we propose to keep the recommendation as it is:

- Limited Modelling Diagrams: Limit the number of modelling diagrams covered in the syllabus to avoid overwhelming students. Provide a rationale for the selection of each modelling diagram type, explaining why they are relevant and beneficial for the course objectives.

R3 Grading Rubrics: Essentially, modelling is a tool for understanding a domain and planning or designing a system. In industry, these diagrams help comprehend problems and plan solutions. In a course, however, students learn syntactical aspects first, before learning their use. The grading rubric could focus initially on technicality and completeness, then progress to evaluating the effectiveness of the diagrams in addressing a specified purpose. This shift from syntax to purpose is crucial for preparing students for real-world applications. Also, rubrics can be a great tool of communication between TAs, TAs and students and between students themselves. We suggest to use rubric more into that direction. Therefore, in addition to a 'technical rubric', we propose to add am evaluation rubric for students to assess their modelling knowledge:

- Technical Rubric: Develop a technical grading rubric for assignments, focusing on evaluating the correctness and completeness of students' modelling diagrams.
- Evaluation Rubric: Introduce a second rubric for students to assess the effectiveness of the learned diagrams. Include checklists in this rubric to enable students to evaluate their learning against the course's learning outcomes.

R4 Use of Familiar Problem Domain: We advocate for instructors adopting a unified approach by selecting a single domain for the entire project, one that students are relatively familiar with. This approach not only fosters student engagement but also allows instructors to tailor the project to align with students' interests and the objectives of the course. Furthermore, it allows students to spend time on the concepts that are introduced in the respective course, namely software modelling, without too much cognitive load spent for understanding a new domain.

- Consistent Problem Domain: Ensure that all assignments in the course revolve around a single problem domain. Encourage instructors to involve students in selecting the problem domain by highlighting the importance of diverse domains and their unique information and development requirements.

6 CONCLUSION

We conducted a mixed-method empirical study to assess the impact of four modelling recommendations from related work, i.e., to use iterative modelling projects with regular feedback, to limit the amount of diagram types, to use detailed grading rubrics, and to use a familiar project domain throughout the course. We applied the four recommendations in two bachelor-level university courses and gathered data from students, teaching assistants, modelling and educational experts through surveys, interviews, detailed assignment analysis, and a focus group discussion. The results indicate the effectiveness of the recommendations in enhancing students' modelling skills and knowledge. However, we also identified areas for improvement, particularly in guiding students to apply their knowledge in real-world industry settings. As a result, we propose modifications to the recommendations, incorporating specific insights from our study to enhance students' modelling experience in university settings and for long-term effectiveness.

REFERENCES

[1] Luciane TW Agner, Timothy C Lethbridge, and Inali W Soares. 2019. Student experience with software modeling tools. *Software & Systems Modeling* 18 (2019), 3025–3047.

[2] Luciane Telinski Wiedermann Agner, Inali Wisniewski Soares, Paulo Cézar Stadzisz, and Jean Marcelo Simão. 2013. A Brazilian survey on UML and model-driven practices for embedded software development. *Journal of systems and software* 86, 4 (2013), 997–1005.

[3] Aditya Akundi and Wilma Ankobiah. 2024. Mapping industry workforce needs to academic curricula–A workforce development effort in model-based systems engineering. *Systems Engineering* (2024).

[4] Omar Badreldin, Timothy Lethbridge, Arnon Sturm, Waylon Dixon, Abdelwahab Hamou-Lhadj, and Ryan Simmons. 2015. The effects of education on students' perception of modeling in software engineering. CEUR Workshop Proceedings.

[5] Shalini Chakraborty and Grischa Liebel. 2023. We do not understand what it says–studying student perceptions of software modelling. *Empirical Software Engineering* 28, 6 (2023), 149.

[6] Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sebastien Mosser, Richard F Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabi Taentzer, et al. 2018. Towards a body of knowledge for model-based software engineering. In *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. 82–89.

[7] Gregor Engels, Jan Hendrik Hausmann, Marc Lohmann, and Stefan Sauer. 2006. Teaching UML is teaching software engineering is teaching abstraction. In *Satellite Events at the MoDELS 2005 Conference: MoDELS 2005 International Workshops Doctoral Symposium, Educators Symposium Montego Bay, Jamaica, October 2-7, 2005 Revised Selected Papers 8*. Springer, 306–319.

[8] Jane Frances Gilgun. 1992. *Definitions, methodologies, and methods in qualitative family research.* SAGE Publications, Inc.

[9] Fabian Gilson. 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*. 98–105.

[10] Hassan Gomaa and L Burgueño. 2017. Teaching Software Modeling and Design.. In *MoDELS (Satellite Events)*. 543–546.

[11] Laure Gonnord and Sébastien Mosser. 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*. 106–113.

[12] Tony Gorschek, Ewan Tempero, and Lefteris Angelis. 2014. On the use of software design models in software development practice: An empirical investigation. *Journal of Systems and Software* 95 (2014), 176–193.

[13] Imed Hammouda, Håkan Burden, Rogardt Heldal, and Michel RV Chaudron. 2014. Case tools versus pencil and paper. In *ACM/IEEE 17th Int. Conf. on model driven engineering languages and systems–educators symposium*.

[14] James Hiller. 2016. Epistemological foundations of objectivist and interpretivist research. (2016).

[15] Sascha Kirstan and Jens Zimmermann. 2010. Evaluating costs and benefits of model-based development of embedded software systems in the car industry–results of a qualitative case study. In *Proceedings Workshop C2M: EEMDD „From code centric to model centric: Evaluating the effectiveness of MDD" ECMFA*.

[16] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. 2008. The focus group method as an empirical tool in software engineering. In *Guide to advanced empirical software engineering*. Springer, 93–116.

[17] Maria Kutar, Carol Britton, and Trevor Barker. 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)*. 1–14.

[18] Ludwik Kuzniarz and Miroslaw Staron. 2005. Best practices for teaching UML based software development. In *International Conference on Model Driven Engineering Languages and Systems*. Springer, 320–332.

[19] Ludwik Kuzniarz, Miroslaw Staron, and Claes Wohlin. 2004. An empirical study on using stereotypes to improve understanding of UML models. In *Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004.* IEEE, 14–23.

[20] Grischa Liebel, Rogardt Heldal, and Jan-Philipp Steghöfer. 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)*. IEEE, 18–27.

[21] Grischa Liebel, Nadja Marko, Matthias Tichy, Andrea Leitner, and Jörgen Hansson. 2018. Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. *Software & Systems Modeling* 17 (2018), 91–113.

[22] Sharan B Merriam et al. 2002. Introduction to qualitative research. *Qualitative research in practice: Examples for discussion and analysis* 1, 1 (2002), 1–17.

[23] Kai Petersen and Cigdem Gencel. 2013. 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*. IEEE, 81–89.

[24] Parsa Pourali and Joanne M Atlee. 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*. 224–234.

[25] Eric J Rapos. 2018. We'll make modelers out of'em yet: introducing modeling into a curriculum. In *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. 130–134.

[26] Rebecca Reuter, Theresa Stark, Yvonne Sedelmaier, Dieter Landes, Jürgen Mottok, and Christian Wolff. 2020. Insights in students' problems during UML modeling. In *2020 IEEE Global engineering education conference (EDUCON)*. IEEE, 592–600.

[27] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. SAGE Publications.

[28] Carolyn B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on software engineering* 25, 4 (1999), 557–572.

[29] Martina Seidl, Marion Scholz, Christian Huemer, and Gerti Kappel. 2014. *UML@ classroom: An introduction to object-oriented modeling*. Springer.

[30] Dave R Stikkolorum, Truong Ho-Quang, and Michel RV Chaudron. 2015. Revealing students' uml class diagram modelling strategies with webuml and logviz. In *2015 41st Euromicro conference on software engineering and advanced applications*. IEEE, 275–279.

[31] Marco Torchiano, Federico Tomassetti, Filippo Ricca, Alessandro Tiso, and Gianna Reggio. 2013. Relevance, benefits, and problems of software modelling and model driven techniques—A survey in the Italian industry. *Journal of Systems and Software* 86, 8 (2013), 2110–2126.

[32] James Vallino. 2006. If you're not modeling, you're just programming: modeling throughout an undergraduate software engineering program. In *International Conference on Model Driven Engineering Languages and Systems*. Springer, 291–300.

[33] Bernd Westphal. 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)*. 690–699. https://doi.org/10.1109/MODELS-C.2019.00105

[34] Jon Whittle and John Hutchinson. 2012. Mismatches between industry practice and teaching of model-driven software development. In *Models in Software Engineering: Workshops and Symposia at MODELS 2011, Wellington, New Zealand, October 16-21, 2011, Reports and Revised Selected Papers 14*. Springer, 40–47.