Qualitative Requirements Elicitation of Student Requirements for Tool-supported Teaching of UML Diagrams

Florian Huber
Kempten University of Applied Sciences
Kempten, Germany
florian.huber@hs-kempten.de

Tobias Eigler
Kempten University of Applied Sciences
Kempten, Germany
tobias.eigler@hs-kempten.de

Georg Hagel
Kempten University of Applied Sciences
Kempten, Germany
georg.hagel@hs-kempten.de

Christian Wolff
University of Regensburg
Regensburg, Germany
Christian.Wolff@informatik.uni-regensburg.de

ABSTRACT

Learning how to model software systems or components is considered to be a central part in the education of future computer scientists. Students are usually introduced to this topic in software engineering courses. As de facto standard, most of them teach the *Unified Modeling Language* (UML) for creating diagrams in specific contexts. This task regularly presents students with significant challenges, as has been widely discussed in literature. Therefore, it is not surprising that there are some tool-supported approaches provided by educators. These are often based on observations. However, comprehensive research on actual students' requirements and aspirations for such tools has so been missing. The contribution of this paper will be a qualitative requirements elicitation for students requirements for tools to support teaching of UML diagrams in software engineering education.

CCS CONCEPTS

• **Social and professional topics → Software engineering education**; • **Software and its engineering** → *Software creation and management*.

KEYWORDS

Software Engineering Education, UML, Tool-Supported Teaching, Students Requirements

ACM Reference Format:
Florian Huber, Tobias Eigler, Georg Hagel, and Christian Wolff. 2023. Qualitative Requirements Elicitation of Student Requirements for Tool-supported Teaching of UML Diagrams. In *ECSEE 2023: European Conference on Software Engineering Education (ECSEE 2023), June 19–21, 2023, Seeon/Bavaria, Germany.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3593663.3593673

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ECSEE 2023, June 19–21, 2023, Seeon/Bavaria, Germany
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9956-2/23/06…$15.00
https://doi.org/10.1145/3593663.3593673

1 INTRODUCTION

In software engineering education, students usually receive textual exercises from educators. These contain requirements descriptions from which UML diagrams are to be generated. Because there are multiple diagram types with individual and in some cases similar looking elements, remembering the diagrams' components can be a first hurdle for students. However, applying those to meet the textual requirements is an even more challenging task for them [13, 16, 17]. Correcting students' diagrams and giving individual feedback is often impossible for educators, because students' solutions can differ from each other and from a sample solution, but can still be correct solutions[6, 7]. Furthermore, due to time restrictions, it is not possible to look at and discuss all of them. Therefore, several tools have been developed to support students in their learning process and provide individual feedback [7]. Many of them are unavailable for a wide range of students at different universities or require the use of specific additional tools (like certain modeling tools) [7]. The heterogeneity of these applications illustrates that different aspects have to be considered to support the learning process of students. So far, a comprehensive research on students requirements for such tools has come up short. Since too little information can be extracted from existing literature, this publication reports on qualitative interviews in which challenges and related requirements from a student's perspective have been collected. Therefore, a guideline-based expert interview was developed and conducted.

The remainder of this paper is structured as follows: section 2 gives an overview about related work. The methodological procedure and a description of the interview participants is outlined in section 3. Finally, the results of the expert interviews are presented in section 4 and discussed in section 5.

2 RELATED WORK

As already outlined in the previous section, there are several approaches for a tool-supported teaching of UML diagrams in software engineering education. Authors in [7] conducted a systematic literature review to provide an overview about those tools. Their functionalities were developed regarding the observations of educators and do not provide a qualitative survey about students' requirements. The authors of [7] also present the five most mentioned challenges in the listed publications that students face according to educators when learning modelling with UML. However, the research does not specifically address the challenges and resulting requirements for such tools.

From an educator's perspective, the challenges of software engineering education have been widely discussed in the literature (e.g. [3, 5, 14]). The most common mistakes by students in software modeling have been extracted by [4]. The authors in [15] have a similar goal. They conducted a systematic literature review about the problems students face during the modeling process. Furthermore, a catalog of challenges and a study are presented. The literature review shows that the students' difficulties with applying the UML are documented in literature. The study reveals problems students encounter solely during the modeling process.

In [8] a multi-case study on the perception of UML modeling tools in education is presented. It points out that there is a need for tailoring those tools to educational purposes. Since this publication does not deal with modeling tools, it differs thematically. Students' perception of UML diagrams is also studied by [9] and can be distinguished for the same reasons.

To the best of our knowledge, so far, a qualitative investigation of challenges and requirements for tool-supported teaching of UML diagrams from a student's perspective has come up short.

3 RESEARCH DESIGN

The development of the research design was carried out according to the specifications of [1] and [2]. The following subsections describe the research interest and translate it into two research questions. Since it is not possible to answer these from existing literature, a qualitative expert interview was conducted with relevant stakeholders. Preliminary quantitative research would be too restrictive, as not all possible requirements can be captured in advance on, for example, a questionnaire. Afterwards, the group of interviewed experts is described. Finally, it is highlighted how the interview results are extracted and evaluated.

3.1 Research Interest and Questions

The overall goal of this paper is to elicit students' challenges and related requirements for a software-based tool for teaching UML diagrams. Not solely the modeling process is considered, but the entire process of such exercise sessions.

In a first step, students should describe the challenges they face in creating UML diagrams based on textual requirements. As described in section 2 these already have been studied in literature. While these are presented as results in section 4, the main focus is on the related requirements that must be met if students are to be supported by software. Before these were discussed in the interviews it was important to talk about challenges and think of software based solutions for them.

Subsequently, students are to be asked about possible tool-supported assistance during the requirements extraction, the modeling phase and the feedback process afterwards. This represents the central points in the interview. The aim is to find specific functionalities that should be met, in order to support them in their learning process.

Based on the described research interest, two questions arise:

- **RQ1:** What challenges do students face when learning modeling with UML in software engineering education?
- **RQ2:** What software-based assistance can help students overcome their challenges in creating UML diagrams from textual exercises?

3.2 Qualitative Guideline-based Expert Interviews

Since knowledge is to be obtained from a specific group of people (students), an expert interview was chosen as methodical approach. Following the definition of [2], students are therefore considered as "experts" because they see themselves confronted with the described problem context and therefore have important insights which only they can reveal - they are experts for their specific software engineering learning situation.

Based on the descriptions of [1] and [2], an interview guideline was developed to answer the outlined research questions.

To provide an easy entry point into the interview, as suggested by [1], the experts are asked three personal questions:

- How old are you?
- Which semester are you in?
- Which course of study do you belong to?

These questions do not add any content value to the interview. They merely provide an overview about the study participants and are a simple start to the interview. The research questions are then asked in the order described. To keep the conversation flowing, some sub-questions are written down in the guideline. These shall serve as additional stimuli, if required.

The duration of the interview is scheduled for about 30 minutes, but can be adapted depending on the interview situation.

3.3 Participants

The qualitative guideline-based expert interview was conducted at the Kempten University of Applied Sciences, Germany. Students in a bachelor's degree program with a computer science related course of study were considered as possible participants. It was a basic requirement to only question students that already have taken the software engineering course and have participated in the exercise sessions. If this would not have been the case, the students would not be considered as experts by the definition of [2]. The expert interview was scheduled in the winter semester 2022 / 2023.

3.4 Evaluation and Information Extraction

As suggested by [2], a qualitative content analysis is a suitable methodological approach to analyze the contents of such guideline-based expert interviews. It is a strictly regulated and qualitatively oriented evaluation method for texts, which where collected in scientific contexts [12]. The process has several stages with different steps that need to be executed [2, 10–12]. In order to support and facilitate the qualitative content analysis, the web application *QCAmap* has been developed by Mayring and Fenzl (please see: https://www.qcamap.org/ui/en/home) to lead researchers through the individual steps. It has also been used to generate the results in section 4.

Table 1: Categories of challenges

Category	%
Object-orientation	100
Application of UML	100
Complexity of UML syntax	80
Processing the textual exercises	80

Table 2: Challenges of object-orientation

ID	Challenge	%
RQ1-10	Abstractness of UML components	60
RQ1-1	Learn and apply object-oriented concepts	40
RQ1-3	Abstraction of texts to UML components	20
RQ1-11	Linking UML components with program code	20

As presented in [12], the two research questions were taken as deductive top categories. Within them, inductive subcategories were developed according to the answers of the participants.

4 RESULTS

4.1 Overview of the Participant Group

Five students agreed to participate as experts. The age distribution ranges from 25 to 31. All of them are in between the fifth and tenth semester of a computer science related bachelor's degree program at the Kempten University of Applied Sciences, Germany. Either the students were enrolled in the *Business Information Systems* or the *Computer Science* course of study. Due to the distributions in age, semester and course of study, it can be assumed that the participants have gained different experiences and can deliver diverse answers to the individual research questions.

4.2 RQ1: What challenges do students face when learning modelling with UML in software engineering education?

For the top category "Challenges", four categories were formed by an inductive process (please see: [2, 10–12]). Those are presented in table 1 and ordered by percentage. The latter refers to the number of times the category (or in the following the individual challenge or requirement) has been named over all interviews.

The four categories:

- Object-orientation (referred to in 100% of the interviews)
- Application of UML (referred to in 100% of the interviews)
- Complexity of UML syntax (referred to in 80% of the interviews)
- Processing the textual exercises (referred to in 80% of the interviews)

were derived from the individual challenges listed in table 2 - table 5.

Table 3: Challenges of application of UML

ID	Challenge	%
RQ1-8	No individual and direct feedback	80
RQ1-4	Variety of modeling possibilities	60
RQ1-19	Finding a strategy / approach	60
RQ1-14	Transfer of the mental model into an UML diagram	40
RQ1-6	Complexity of larger diagrams	20
RQ1-7	Complexity of modeling tools	20
RQ1-16	Identification of the correct design pattern	20
RQ1-17	Correct application of a design pattern	20

Table 4: Challenges with the complexity of the UML syntax

ID	Challenge	%
RQ1-5	Variety of components to be learned	60
RQ1-9	Differentiation of the components	60

Table 5: Challenges with processing the textual exercises

ID	Challenge	%
RQ1-2	Extraction of requirements / components	80
RQ1-13	Identification of relations of the diagram components	60
RQ1-18	Availability of exercises	20

Table 6: Categories of requirements

Category	%
Provision and processing of the textual exercise	100
Feedback and improvement	100
Modelling	40

4.3 RQ2: What software-based assistance can help students overcome their challenges in creating UML diagrams from textual exercises?

Based on the second research question, the top category "Requirements" emerged. By clustering the individual answers of the interviews, the three inductive categories:

- Provision and processing of the textual exercise (referred to in 100% of the interviews)
- Feedback and improvement (referred to in 100% of the interviews)
- Modelling (referred to in 40% of the interviews)

were formed. They are visible alongside the percentage in table 6. The individual requirements that were clustered in these categories are listed in table 7 - 9.

Table 7: Requirements for the provision and the processing of textual exercises

ID	Requirement	%
RQ2-17	Larger amount of exercises	100
RQ2-6	Highlight important components	100
RQ2-5	Different levels of difficulty	60
RQ2-7	Help text	40
RQ2-10	Collaborative work	20
RQ2-11	Step by step guide through tasks	20
RQ2-13	Different assistance for different levels of difficulty	20
RQ2-15	Strategy guide	20
RQ2-16	Reversing exercises	20

Table 8: Requirements for feedback and improvements

ID	Requirement	%
RQ2-3	Software-supported individual feedback	100
RQ2-2	Sample solution	60
RQ2-14	Linking further literature	20

Table 9: Requirements for modeling

ID	Requirement	%
RQ2-8	Linking the diagram to program code	20
RQ2-9	UML editor	20
RQ2-12	Proposal for subsequent step	20

5 DISCUSSION

5.1 RQ1: What challenges do students face when learning modeling with UML in software engineering education?

The four categories illustrate which areas of teaching UML diagrams in software engineering education are particularly challenging for the surveyed students. For teachers, this result can serve as an indicator to pay special attention to the mentioned topics in teaching.

As acknowledged in literature (e.g. [7]), object-orientated concepts and their application are challenging for students. Therefore, it is not surprising that this category has been formed inductively. One reason for this, for example, is the abstractness of the UML components. It is difficult for the students to convert the descriptions contained in the textual exercises into an UML structure without making mistakes (e.g. confuse attributes as classes). Furthermore, some interview participants stated that they struggled to understand how to convert the components in their diagrams to program code.

Students also encounter hurdles in the practical application of the knowledge they have learned in theory. The lack of direct feedback on their individual diagrams is seen as particularly challenging. Some participants named a sample solution for each exercise in that context as a requirement. However, regarding their learning success, the participants were telling that individual feedback would have been a greater contribution than a sample solution. The latter can differ from a students' diagram. But that does not mean that they had made a mistake in their models. Especially for novices, the multitude of modeling options can be difficult.
Although all participants had access to a professional modeling tool (for free with their student license) only one stated to have used it. Besides the complexity of these tools, they preferred to draw the diagrams by hand because they cannot use software tools during the exam.
When modeling UML diagrams from textual exercises, careful reading and understanding the text is critical. The participants indicated that they gradually developed a strategy on how to extract all information from the given texts and transfer it from their own mental model into an actual UML diagram. It is this cognitive process that the authors consider to be one of the most important and critical parts, as it determines the students' success. A good strategy can be developed through guidance, combined with a variety of exercises that challenge students, but do not overwhelm them at their current level of knowledge.

The third category contains all challenges students face regarding the UML syntax. The variety of diagram types and their individual components can be overwhelming for novices. Also, many of the diagram components look similar but have different meanings, which can be even more confusing.

Processing the textual exercises is the fourth category. Extracting all information within the textual exercises and understanding the relations of the components is regarded difficult by the participants. One stated that the limited availability of exercises was difficult. However, most participants didn't name it as a challenge but named that topic as an requirement for tool-supported teaching of UML diagrams.

5.2 RQ2: What software-based assistance can help students overcome their challenges in creating UML diagrams from textual exercises?

The previously described setting during the classes' exercise sessions can be divided into three main phases. The inductively developed categories reflect these. At first student acquire the theoretical knowledge and extract information from the given texts. Afterwards they start modeling. Ultimately, they expect feedback to understand if their solution is correct or contains mistakes. It is interesting that only 40% of the participants want software-based support during the modeling phase. According to the findings of the interview, they prefer to generate a first diagram without any interruption. They see subsequent individual feedback as extremely helpful for their learning process.

Most requirements for tool-supported assistance were named in the category for provision and processing textual exercises. A larger amount of texts was seen as one of the most helpful requirements. However, according to the participants a sample solution, or preferably individual feedback is required alongside with them. Otherwise it is hard to determine whether a valid UML diagram was generated from the textual descriptions. The participants stated

that such digitally provided exercises should come with different levels of difficulty, which match their individual state of knowledge. Highlighting important information (e.g. class names) within the texts is another very important requirement. This and other functionalities like help texts could, according to the results, be available only for specific levels of difficulty. Because students often prepare in groups for an upcoming exam, collaboration has also been mentioned as desirable.
As stated previously, a good strategy when gathering all important information from the textual exercises is vital. The participants could imagine a step-by-step guide or a strategy guide that walks them through the exercise.
One participant even imagined a reversal of the exercise process. According to this, students would see an UML diagram and be required to write the requirements text. Usually in the professional world, customers, stakeholders, etc. approach computer scientists with their ideas. Therefore, the tasks are deliberately set in this way in academia. However, it could contribute to the learning success of the students to turn this process around for an individual exercise.

The second cluster concentrates on the tool-supported assistance during the feedback process. Individual and direct feedback by a software was named in all of the interviews. The participants regarded it as the most vital point that could improve their learning success. Especially in the beginning it is difficult for them to compare their own diagrams to a sample solution and determine, whether they made a mistake or not. Feedback which is individually tailored in combination with specific suggestions for further literature could boost the students' success.

Cluster three contains all the requirements that were named to assist during the modeling process. According to the interview answers, linking the UML components to program code could be helpful to reduce the abstractness. A simple UML editor with reduced complexity was also called for. An interesting requirement is the proposal for subsequent steps. However, during the interview the participant told us that this might be a feature that distracts to much and that feedback after finishing the modeling process might be more helpful.

6 CONCLUSION AND FURTHER WORK

In this paper, we have presented results from guideline-based expert interviews in order to collect the requirements of students on tool-supported assistance in the teaching of UML diagrams in software engineering education. The two research questions reveal which challenges students face when modeling UML diagrams from textual exercise descriptions and what requirements they have for a software that helps them overcome these. With the results, educators are given an overview of subject areas which students seem to struggle with and can adapt their teaching efforts accordingly. Furthermore, there is a great number of software tools which aim to support educators in software engineering education [7]. The results presented above can serve as basis for further developments in this area.

All results and statements refer to the conducted interviews. A complete picture in terms of challenges and requirements is difficult to achieve due to students' individual differences and experiences. Results presented in literature (e.g. [7, 15]) can deliver additional insights. The educators' side must also be considered and might be a topic for future work.

REFERENCES

[1] Nina Baur and Jörg Blasius. 2014. *Handbuch Methoden der empirischen Sozialforschung.* Springer Fachmedien Wiesbaden, Wiesbaden. https://doi.org/10.1007/978-3-531-18939-0

[2] Alexander Bogner, Beate Littig, and Wolfgang Menz. 2014. *Interviews mit Experten: Eine praxisorientierte Einführung.* Springer Fachmedien Wiesbaden, Wiesbaden. 1–105 pages. https://doi.org/10.1007/978-3-531-19416-5

[3] Barbara Bracken. 2003. Progressing from Student to Professional: The Importance and Challenges of Teaching Software Engineering. *J. Comput. Sci. Coll.* 19, 2 (dec 2003), 358–368.

[4] Stanislav Chren, Barbora Buhnova, Martin Macak, Lukas Daubner, and Bruno Rossi. 2019. Mistakes in UML Diagrams: Analysis of Student Projects in a Software Engineering Course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).* 100–109. https://doi.org/10.1109/ICSE-SEET.2019.00019

[5] Carlo Ghezzi and Dino Mandrioli. 2006. The Challenges of Software Engineering Education. In *Software Engineering Education in the Modern Age*, Paola Inverardi and Mehdi Jazayeri (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 115–127.

[6] Florian Huber and Georg Hagel. 2020. Work-in-Progress: Towards detection and syntactical analysis in UML class diagrams for software engineering education. In *2020 IEEE Global Engineering Education Conference (EDUCON).* 3–7. https://doi.org/10.1109/EDUCON45650.2020.9125244

[7] Florian Huber and Georg Hagel. 2022. Tool-supported teaching of UML diagrams in software engineering education - A systematic literature review. In *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO).* 1404–1409. https://doi.org/10.23919/MIPRO55190.2022.9803560

[8] Grischa Liebel, Omar Badreddin, and Rogardt Heldal. 2017. 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).* 124–133. https://doi.org/10.1109/CSEET.2017.29

[9] Adriana Lopes, Igor Steinmacher, and Tayana Conte. 2019. UML Acceptance: Analyzing the Students' Perception of UML Diagrams. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering* (Salvador, Brazil) *(SBES 2019).* Association for Computing Machinery, New York, NY, USA, 264–272. https://doi.org/10.1145/3350768.3352575

[10] Philipp Mayring. 2000. *Qualitative Inhaltsanalyse. Grundlagen und Techniken.* Weinheim: Deutscher Studien Verlag.

[11] Philipp Mayring. 2010. *Qualitative Inhaltsanalyse.* VS Verlag für Sozialwissenschaften, Wiesbaden, 601–613. https://doi.org/10.1007/978-3-531-92052-8_42

[12] Philipp Mayring and Thomas Fenzl. 2014. *Qualitative Inhaltsanalyse.* Springer Fachmedien Wiesbaden, Wiesbaden, 543–556. https://doi.org/10.1007/978-3-531-18939-0_38

[13] Juan Carlos Muñoz Carpio, Michael Cowling, and James Birt. 2018. Framework to Enhance Teaching and Learning in System Analysis and Unified Modelling Language. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE).* 91–98. https://doi.org/10.1109/TALE.2018.8615284

[14] Sofia Ouhbi and Nuno Pombo. 2020. Software Engineering Education: Challenges and Perspectives. In *2020 IEEE Global Engineering Education Conference (EDUCON).* 202–209. https://doi.org/10.1109/EDUCON45650.2020.9125353

[15] 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).* 592–600. https://doi.org/10.1109/EDUCON45650.2020.9125110

[16] D. R. Stikkolorum, F. Gomes de Oliveira Neto, and M. R. V. Chaudron. 2018. Evaluating Didactic Approaches Used by Teaching Assistants for Software Analysis and Design Using UML. In *Proceedings of the 3rd European Conference of Software Engineering Education* (Seeon/ Bavaria, Germany) *(ECSEE'18).* Association for Computing Machinery, New York, NY, USA, 122–131. https://doi.org/10.1145/3209087.3209107

[17] Sylvia Stuurman, Harrie Passier, and Erik Barendsen. 2016. Analyzing Students' Software Redesign Strategies. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '16).* Association for Computing Machinery, New York, NY, USA, 110–119. https://doi.org/10.1145/2999541.2999559