How teaching conceptual modeling to robotics students changes their perception of software engineering

Meenakshi Manjunath, Jeshwitha Jesus Raja and Marian Daun

Center for Robotics, Technical University of Applied Sciences Würzburg-Schweinfurt, Schweinfurt, Germany

Abstract

Teaching software engineering to non-software engineering students is a challenging task. Often, limitation of time dedicated to software engineering education, limits the subjects to be taught significantly. Furthermore, non-software engineering students, typically equate software engineering with programming, thus focussing their learning on code-related aspects. To counter this, we developed a course setup making extensive use of teaching conceptual modeling to educate students in software engineering. This shall support students in developing abstract-thinking skills and understanding the need for conceptualization and planning in early development phases. Thus, the students can experience a different perspective on software development than they are used to. In this paper, we report an interview study with international bachelor students in a robotic degree program. The study aims at investigating students' perception of software engineering and conceptual modeling. We compare two different student groups, one group attended a more traditional programming focused software engineering course, while the other group attended a conceptual modeling focused software engineering course. Results indicate that conceptual modeling education can indeed improve students' abstract thinking skills as well as their appraisal of abstract thinking, conceptualization, as well as planning software development in advance.

Keywords

Software Engineering, Conceptual Modeling, Student Perception, Education

1. Introduction

The importance of teaching conceptual modeling as part of computer science and software engineering curricula has been widely acknowledged [1, 2]. However, nowadays, software engineering is a vital part not only of computer science and software engineering curricula, but also of other disciplines like data science, social sciences, or technical engineering disciplines [3]. As software engineering is becoming an important part not only of educating future software engineers but also technical engineers with more and more need for developing software intensive systems, a challenge arises in teaching these students conceptual modeling as they are often not used to abstract thinking and model usage from the backgrounds of their degree programs.

ER2023: Companion Proceedings of the 42nd International Conference on Conceptual Modeling: ER Forum, 7th SCME, Project Exhibitions, Posters and Demos, and Doctoral Consortium, November 06-09, 2023, Lisbon, Portugal

meenakshi.manjunath@study.thws.de (M. Manjunath); jeshwitha.jesusraja@study.thws.de (J. Jesus Raja); marian.daun@thws.de (M. Daun)

0009-0005-6421-1450 (M. Manjunath); 0009-0008-7886-7081 (J. Jesus Raja); 0000-0002-9156-9731 (M. Daun)

Commonly, software engineering education in these areas is typically limited to one course focussing on the core aspects of developing software. This often involves programming to show students how concepts materialize in code. This approach suffers a huge shortcoming. It does not prepare students for abstract thinking and collaborating in the design phases of software. Thus, we employed an approach to teaching software engineering to robotic students that particularly focuses on teaching abstract thinking by using conceptual modeling. Thus, we do not teach conceptual modeling in addition to a fundamental software engineering education, but use conceptual modeling as a medium to teach the core principles of software engineering.

Another challenge for teaching conceptual modeling is related to various backgrounds and needs of students. This, is particularly true for very heterogeneous student bodies. Thus, teaching conceptual modeling to suit the specific requirements of international bachelor students from various different countries can pose challenges, such as the need for individual feedback and addressing cultural differences [4]. By exploring existing research and examining the approaches used in prior studies, this paper aims to provide insights into effective instructional practices for teaching conceptual modeling to international bachelor students. Drawing on the research findings of Bayman and Mayer [1], Kayama et al. [2], and Daun et al. [4], this study seeks to identify promising strategies that can enhance the understanding and application of conceptual modeling principles among this student population.

In our case, we report from a robotics degree program, where we find two particular challenges for teaching conceptual modeling:

- Limited exposure to abstract or out-of-the-box thinking due to the predominantly technical nature of the other courses in the degree program
- International student body, consisting of various backgrounds with differing needs due to their primary education system

To address these two challenges, we focused our software engineering education of these students on teaching conceptual modeling to develop abstract thinking skills. In this paper, we contribute an interview study, investigating international robotics students' opinion on using conceptual modeling as major medium for software engineering education. To examine teaching methods for the course of software engineering, specifically focusing on conceptual modeling, as taught to international students in a robotics bachelor degree program, this paper compares two approaches employed in different semesters. The results of the study indicate that students highly value the creative aspect of conceptual modeling and appreciate the fact that there is no definitive right or wrong solution. They find joy in exploring different possibilities and embracing the freedom to think outside the box.

The structure of the paper is as follows: Section 2 provides an overview of related work and key findings on teaching conceptual modeling. Section 3 describes the study setup, outlining the two distinct approaches implemented at the university for teaching the same subject to two batches of bachelor students, as well as the interview setting. In Section 4, we present a review of our major findings based on interviews conducted with the students. Finally, in Section 5, we conclude the paper by discussing the implications of our findings and highlighting avenues for future research.

2. Related work

Over the last decades, various teaching approaches have been established to teach software engineering for various learner groups [5]. Commonly used learning techniques are gamification [6], use of industrial case studies [7, 8], simulation [9], inverted/flipped classrooms [10], etc.

Among these, the use of conceptual models to teach software engineering concepts has been investigated and proven useful. Conceptual modeling is commonly introduced in the course of requirements engineering education [11] but is also used to transport other concepts. For instance, the paper by X Franch [12] has shown that the use of goal models can foster the understanding of data structures. For this idea, Bogdanova et al. have shown that challenges do exist for teaching conceptual data modeling [13]. Particularly, challenging for students is the abstract nature of conceptual models as well as data structures. Such challenges often transcend the boundaries of conceptual modeling and are relevant to the broader field of software engineering education [14]. Factors such as students' prior knowledge, their ability to grasp abstract concepts, and the complexity of the subject matter can pose difficulties regardless of the specific topic being taught. In particular, it has shown that it is difficult for students to rate their competencies with respect to conceptual modeling, as it considerably differs from other computer science subjects [15].

Conceptual modeling education is often taught majorly to postgraduate or advanced bachelor students [16, 17]. At these later stages it is often seen as easier, to introduce complex abstract concepts that are challenging on a cognitive level. However, the introduction of abstract thinking in earlier stages is also found to have considerable advantages for the evolution of students' capabilities throughout their studies [16, 17].

For a recent literature review on educational approaches for conceptual modeling, refer to [18]. In the study, Rosenthal et al. found 121 contributions with respect to conceptual modeling education, concluding that, among others, there is a need for further investigating learner needs and learner perception of modeling processes.

Based on their experiences, Buchmann et al. report in [19] on their experiences from teaching conceptual modeling in education. In particular, they identify six common oversimplifications students make, when it comes to conceptual modeling. Although the background of the students is not revealed, from our experience, this matches with common oversimplifications made by computer science and software engineering students. Based on these findings, Buchmann et al. propose in [20] an educational approach to counter these oversimplifications and allow students to gain deeper insights avoiding these oversimplifications.

In contrast, in our case, we do not report on our experiences but survey the students themselves for their perception. Furthermore, we investigate robotics students that are trained in software engineering and not software engineering students. Thus, as we will see student's perception of conceptual modeling (without proper training) differs considerably, as we do not see individual oversimplifications but in the beginning neglecting the usefulness of conceptual modeling at all. Thus, our teaching approach does also not aim at avoiding these misinterpretations but rather focuses on using conceptual modeling to train students in software engineering without drifting into very detailed programming problems but by keeping the big picture of software engineering education [21, 22] alive.

3. Study setup

3.1. Research goal and research questions

The primary objective of this study was to investigate students' opinion on teaching conceptual modeling in a software engineering course for international robotics students. Therefore, we developed a questionnaire to interview students that had attended a software engineering course specifically tailored towards conceptual modeling and students who attended a more traditional software engineering course focusing on the relation of software engineering concepts and code fragments. By investigating these two teaching methods and their perceived outcomes, the study aimed to identify the instructional approach that optimally supports the learning needs and achievements of international students in this domain.

3.1.1. RQ1: Expectations towards conceptual modeling

We wanted to know what the students' expectations towards conceptual modeling were before the course and how these have changed by the course. In particular, we want to see whether there is a difference between students taking the first course using models to accompany code and the second course emphasizing conceptual modeling.

3.1.2. RQ2: Perceived learning outcome

We wanted to know what the students see as their major learning outcomes of the course. In particular, we wanted to see how the perceived learning outcomes differ between students attending different courses.

3.1.3. RQ3: Course perception

We wanted to know how the students perceived the course setup. What instructional approaches for teaching conceptual modeling were deemed particularly useful, what else do students imagine can help their learning.

3.2. Course design

The software engineering course is offered in the third semester of the robotics degree program. We interview participants of two different editions: winter term 2021 and winter term 2022. These courses were given by different professors employing different teaching approaches that influenced students' perception of software engineering. The course was structured into two weekly sessions, each spanning 90 minutes. Students were evaluated through a written examination at the end of the semester, contributing to the award of 5 credit points for the module.

The first installment concentrated on a wide variety of software engineering topics. Conceptual models were used to describe code snippets in an abstract, easier to understand way. However, the focus of the instruction was on developing software by programming. The focus was primarily on individual comprehension of modeling through a code-oriented approach, with limited group work involved.

The second installment employed an approach with more emphasis on conceptual modeling. Here, students did not only see conceptual models related to code snippets, but were actually tasked with developing conceptual models on their own. As further contrast, the course was limited to the major non-programming aspects of software engineering: Software development processes, requirements engineering, architecture design, and quality assurance. A key goal was to teach the fundamental aspects aside from programming, as we noticed that students developed a very code-centric habit and neglected planning and testing phases when developing robotic software. Therefore, we used conceptual modeling to show a completely new side of software engineering. As modeling languages, we chose modeling languages from all three major perspectives (cf. [23]). In detail, we used UML Class Diagrams and ER (Entity-Relationship) models to represent the structure, Data Flow diagrams and UML Use Case diagrams to depict functional aspects, and State Charts and Finite Automata to illustrate the dynamic behavior.

3.3. Participants

The participants in the study are bachelor students from the robotics degree program from two different semesters:

- Group 1: sixth semester students that attended the course in winter 2021
- Group 2: fourth semester students that attended the course in winter 2022

Each student was invited to participate in an interview without prior knowledge of the interview topic. The purpose of this approach was to ensure that their responses were spontaneous and unaffected by pre-preparation or reliance on internet knowledge. To ensure unbiased answers, none of the professors giving the courses were involved in the interviewing process.

3.4. Procedure and materials

The selection of students for the interviews was based on their regular attendance in the course and their demonstrated understanding of the core concepts, prioritizing those who exhibited a genuine interest beyond exam-focused studying. To ensure a comprehensive understanding of the students' perspectives, individual interviews were conducted over a period of approximately ten days. To maintain the integrity of the interviews, interviewees were specifically instructed not to share the questions or discuss the interview content with other participants. As mentioned above, the participants were told about the topic of the interview on spot to avoid ingenuity in their answers. The questions of the interview are as follows:

1. What do you think conceptual modeling is? Have you done modeling before? What modeling languages do you know?
2. What are your expectations from a software engineering course regarding conceptual modeling?
3. In what ways do you believe conceptual modeling can enhance and contribute to the development process of a software system?
4. What potential challenges or difficulties do you foresee in comprehending and applying conceptual modeling techniques? How do you propose the course can effectively address these challenges to facilitate better understanding and application of the concepts?

5. What was your initial opinion of the course and conceptual modeling before attending the lectures, and how did your perception change by the end of the semester?
6. From your perspective, what teaching methods or approaches do you believe would be the most effective in facilitating your understanding and application of conceptual modeling concepts? In your experience, what aspects of the course or instructional methods were helpful, what aspects were not, and what specific factors made a significant impact on your learning journey?
7. How important do you think it is to integrate real-world examples and industry practices into the course content?
8. In addition to the course materials provided, are there any supplementary resources or learning materials that you think would enrich your learning experience in terms of understanding the core concepts and preparing for exams?
9. What types of assessment and evaluation methods do you believe would be appropriate and effective for a course on conceptual modeling?

4. Results

Table 1 summarizes the most common student answers. Therefore, we have categorized the major responses to the respective aspects of the questions asked. For each concern, the three most prominent themes in responses are summarized for both groups of students, group 1 being the students attending the traditional software engineering lecture and group 2 being the students attending the model-based course. Next, we take a detailed look at the individual responses with respect to the research questions.

4.1. RQ1: Expectations towards conceptual modeling

4.1.1. Previous experience with conceptual modeling

Among the students who took part in the study, it was found that only a quarter of them had previous exposure to conceptual modeling, either through their experience with programming UML class diagrams or their familiarity with data flow diagrams. On the other hand, the majority of the students were initially unaware of the specific modeling languages associated with conceptual modeling. However, they expressed a high level of receptiveness and enthusiasm to acquire new knowledge and understanding in this domain.

4.1.2. Anticipations from the course

Among the students who had prior experience with conceptual modeling, which accounted for 25% of the participants, their expectations were aligned with utilizing modeling techniques such as UML, state charts, data flow diagrams, and similar tools. Conversely, the students without prior experience in conceptual modeling held different expectations, anticipating a focus on programming or the utilization of pseudocode as part of the learning process.

Table 1
Summary of interview results

Question	*Category 1*		*Category 2*		*Category 3*	
	Group 1	*Group 2*	*Group 1*	*Group 2*	*Group 1*	*Group 2*
Conceptual Modeling Understanding	Previous experience in some form		Some personal idea about CM		No idea about CM	
	3	4	1	1	0	1
Expectations from the Course	Provide basic understanding of CM		An extension to the programming courses		Preparing for real life use cases and scenarios	
	1	2	2	3	2	2
Contribution of Conceptual Modeling	Design improvement		Better communication		Efficient and optimized development	
	2	3	0	4	0	1
Challenges and Solutions	Better understanding, i.e. more examples		Real life application, i.e. hands on modeling		Time consuming in lecture, i.e. extra sessions	
	1	4	2	2	0	1
Effective Teaching Methods	Discuss real world problem and solve in class		Additional sessions with programming in line		More group discussions and brainstorming	
	2	2	1	2	1	1
Supplementary Resources	Referenced books		Online tutorials		Workshop/additional sessions	
	1	3	0	1	2	5
Assessment Methods	Written exam		Practical assignments		Group projects	
	1	2	1	1	1	4

4.1.3. Expectations after the course

Students taking the first installment tend to believe that software engineering should equal programming and overhead for planning should be reduced to a minimum. In contrast, the students taking the second installment have a far more positive view on conceptual modeling and the usefulness of models. The students recognize the significance of conceptual modeling and expresses a preference for initiating any project with this approach rather than diving straight into it without proper planning, conceptualization and error handling.

4.2. RQ2: Perceived learning outcome

4.2.1. Software engineering is not programming

Initially expecting another programming course, the students were surprised by the focus on conceptual modeling. However, as the semester progressed, they recognized the course's impact in expanding their skill set. Results showed improved performance and increased enthusiasm. From the students' perspective, the course became a core pillar in the field of robotics, providing students with a deeper understanding of modeling and valuable skills for their future. Despite the initial surprise, the emphasis on conceptual modeling proved transformative, broadening horizons and establishing a strong foundation for their pursuit in robotics.

4.2.2. Conceptual modeling helps abstract thinking

Throughout the semester, the students gradually embraced the concept of abstract and creative thinking. They recognized the significance of software engineering and conceptual modeling in the multidisciplinary realm of robotics. The course provided them with a valuable foundation in understanding software-intensive systems, aligning their learning journey with the demands of the field. As the semester unfolded, the students began to appreciate the role of conceptual modeling in advancing their skills and knowledge in robotics.

4.2.3. Communication is important

During the initial stages of the course, it became apparent that the student groups faced challenges in effectively communicating their ideas and perspectives during problem-solving exercises. However, through the process of collaborative conceptual modeling, students were encouraged to express their thoughts freely and creatively. This experience not only nurtured their communication skills, but also highlighted the importance of effective collaboration in the industry, preparing them for future professional endeavors.

4.3. RQ3: Course perception

Drawing conclusions from the interviews, we can concretely state that the participating students regarding the perception of the course.

- Students, with no prior knowledge, were unaware of the actual course content (before the start of the semester) as they built up the assumption that the Software Engineering course would be completely based on programming.
- All the students share the opinion that there should be an increased provision of examples and exercises to illustrate how a model should be constructed from the professor's perspective. They found the syntaxes of the models initially challenging to memorize at the start of the course, but over time, they gradually became accustomed to them.
- All the students agreed that the course material provided sufficient understanding of the concepts of conceptual modeling. However, they suggested that the material could be further improved by incorporating annotated slides specifically designed to aid exam preparation, making it easier to comprehend and retain the information.
- The students stressed the significance of including real-world examples and industry practices on a smaller scale in the course, deeming them essential for practical relevance.
- Ultimately, all students expressed a preference for having additional exercises with solutions to practice with. This preference stemmed from the understanding that conceptual models can be interpreted in various ways, and practicing with diverse exercises would enhance their ability to apply and interpret models effectively.

4.4. Inferences

The different teaching approaches for software engineering had a profound impact on students' understanding and perspective of software engineering, highlighting the importance of using conceptual modeling for shaping students' ideologies and experiences.

It has shown that, from a students' perspective, incorporating a course on conceptual modeling into multidisciplinary programs like robotics can support software engineering education considerably. Thus, the focus of education is shifted from programming to aspects particular to software engineering. Particularly, students understand the need for conceptualization during development and the need to place adequate emphasis on the design phases rather than just beginning to program right away. Thus, the significance of conceptual modeling in shaping and nurturing young minds is underscored. It fosters the development of abstract thinking, enabling students to effectively tackle a wide range of future challenges and problem-solving scenarios.

When talking about problem-solving, it is a necessity to have structured problem-solving approach which is innately taught by conceptual modeling. Additionally, conceptual modeling emphasizes that there is rarely a single perfect solution in software engineering. It encourages software engineers to generate and evaluate multiple solution ideas, considering their advantages and disadvantages. This fosters critical thinking and enables informed decision-making for effective and efficient software development.

5. Conclusion and future work

In this paper, we investigated students' perception of a software engineering course based on teaching conceptual modeling to instruct robotics students in software engineering. The findings shed light on the significance of conceptual modeling in the field of software engineering, highlighting its role in fostering skills such as planning, abstract conceptualization, and effective communication. The study revealed that students recognized the need to discuss programs not only with stakeholders but also with other developers. However, they also acknowledged the challenges associated with talking about code. To address this, the use of easily understandable models emerged as a valuable tool for enhancing communication and facilitating a deeper understanding of software-intensive systems. Conceptual modeling allowed students to think abstractly about these systems, enabling self-reflection and the generation of their own ideas during the development process.

Overall, the study highlighted the transformative role of conceptual modeling in shaping students' ideologies and experiences in the field of software engineering. It emphasized the need for a comprehensive teaching approach that goes beyond programming and encompasses the various stages and aspects of software engineering. By equipping students with the necessary skills and promoting collaboration, conceptual modeling contributes to their success in the multifaceted field of robotics, which requires a diverse range of skill sets.

In terms of future work, it is recommended to explore the long-term impact of different teaching approaches on students' career prospects and professional development. Further research could also investigate the integration of industry collaborations and internships to provide practical experience in applying conceptual modeling techniques.

References

[1] P. Bayman, R. E. Mayer, Using conceptual models to teach basic computer programming., Journal of Educational Psychology 80 (1988) 291.

[2] M. Kayama, S. Ogata, K. Masymoto, M. Hashimoto, M. Otani, A practical conceptual modeling teaching method based on quantitative error analyses for novices learning to create error-free simple class diagrams, in: IIAI 3rd Int. Conf. on Advanced Applied Informatics, IEEE, 2014, pp. 616–622.

[3] M. Daun, Software engineering education for technical engineering degrees, IEEE Software (2023).

[4] M. Daun, J. Brings, P. A. Obe, K. Pohl, S. Moser, H. Schumacher, M. Rieß, Teaching conceptual modeling in online courses: Coping with the need for individual feedback to

modeling exercises, in: IEEE 30th Conf. on Software Engineering Education and Training (CSEE&T), IEEE, 2017, pp. 134–143.
[5] M. R. Marques, A. Quispe, S. F. Ochoa, A systematic mapping study on practical approaches to teaching software engineering, in: IEEE Frontiers in Education, IEEE, 2014, pp. 1–8.
[6] M. M. Alhammad, A. M. Moreno, Gamification in software engineering education: A systematic mapping, Journal of Systems and Software 141 (2018) 131–150.
[7] C. Wohlin, B. Regnell, Strategies for industrial relevance in software engineering education, Journal of Systems and Software 49 (1999) 125–134.
[8] M. Daun, A. Salmon, T. Weyer, K. Pohl, B. Tenbergen, Project-based learning with examples from industry in university courses, in: 2016 IEEE 29th Int. Conf. on Software Engineering Education and Training (CSEET), IEEE, 2016, pp. 184–193.
[9] M. B. Blake, A student-enacted simulation approach to software engineering education, IEEE Transactions on Education 46 (2003) 124–132.
[10] G. C. Gannod, J. E. Burge, M. T. Helmick, Using the inverted classroom to teach software engineering, in: 30th Int. conference on Software Engineering, 2008, pp. 777–786.
[11] M. Daun, A. M. Grubb, V. Stenkova, B. Tenbergen, A systematic literature review of requirements engineering education, Requirements Engineering 28 (2023) 145–175.
[12] X. Franch, M. Ruiz, Goal-oriented models for teaching and understanding data structures, in: Conceptual Modeling: 40th Int. Conf., ER 2021, Springer, 2021, pp. 227–241.
[13] D. Bogdanova, M. Snoeck, Camelot: An educational framework for conceptual data modelling, Information and software technology 110 (2019) 92–107.
[14] O. Pastor, A. Pierantonio, G. Rossi, Teaching modeling in the time of agile development, Computer 55 (2022) 73–76.
[15] M. Daun, J. Brings, P. A. Obe, V. Stenkova, Reliability of self-rated experience and confidence as predictors for students' performance in software engineering, Empirical Software Engineering 26 (2021) 80.
[16] F. Dalpiaz, J. Horkoff, J. Lockerbie, X. Franch, E. Yu, J. Mylopoulos, et al., Teaching goal modeling in undergraduate education, in: 1st Int. iStar Teaching Workshop at 27th Int. Conf. on Advanced Information Systems Engineering (CAiSE), 2015, pp. 1–6.
[17] M. Ruiz, F. B. Aydemir, F. Dalpiaz, et al., Using conceptual models in research methods courses: An experience using istar 2.0, in: 2nd Int. iStar Teaching Workshop at 36th Int. Conf. on Conceptual Modeling, 2017, pp. 48–57.
[18] K. Rosenthal, B. Ternes, S. Strecker, Learning conceptual modeling: structuring overview, research themes and paths for future research, in: ECIS, 2019.
[19] R. Buchmann, A.-M. Ghiran, V. Döller, D. Karagiannis, Conceptual modelling in education: a position paper, in: 4th Workshop on Managed Complexity, 2019.
[20] R. A. Buchmann, A.-M. Ghiran, V. Döller, D. Karagiannis, Conceptual modeling education as a "design problem", Complex Systems Informatics and Modeling Quarterly (2019) 21–33.
[21] M. Shaw, Software engineering education: A roadmap, in: Conf. on the Future of Software Engineering, 2000, pp. 371–380.
[22] C. Ghezzi, D. Mandrioli, The challenges of software engineering education, in: 27th Int. Conf. on Software Engineering, 2005, pp. 637–638.
[23] A. M. Davis, Software requirements: objects, functions, and states, Prentice-Hall, 1993.