Inquiry- and Research-based Teaching in a Course on Model Checking

Sebastian Krings, Philipp Körner, and Joshua Schmidt

Universität Düsseldorf, Hochschule Niederrhein
sebastian.krings@uni-duesseldorf.de

Abstract. In this presentation, we discuss a recent publication on our course on model checking. The course has been shifted from a classical lecture-based format to inquiry and research-based teaching. In the article to be presented, we documented course development, presented some didactic methods used and evaluated the course based on peer review and student feedback. Furthermore, we tried to assert student engagement empirically.

1 Introduction and Motivation

The development and improvement of model checkers [3] for the validation of hard- and software is an ongoing research topic in computer science [4]. Model checking research connects theoretical and practical aspects; new algorithms are often implemented inside well-known model checkers which have been in development for many years. This is seldom taken into account by university courses, which often remain on the theoretical level.

In particular, both complexity and code volume of commonly used model checkers prevent students from coming to grips with internal workings. In consequence, typical courses on model checking stay on a theoretical level. Students often have no way of experiencing the practical aspects of developing a model checker, leading to shortcomings in different areas as discussed in our paper [5].

To improve, we remodeled our course on model checking as published at SEUH 2019 [5], moving it from classic lectures to active learning techniques for an improved hands on experience. Furthermore, we noticed that students writing theses at our chair sometimes lack the required knowledge about how research is performed and thus need close supervision and initial training. In order to motivate individual research and to enable students to train their skills, we decided to move from a classic lecture setting to inquiry-based and research-based learning.

Barron et al. [1] identified four important aspects contributing to the success of inquiry-based courses. In the presentation, we will discuss those factors and show how we realized them in our course.

Additionally, the course provides other more long term benefits to the chair, as we can use the resulting software in bachelor's and master's theses, given that it is more easily accessible than other model checkers and can thus help evaluating alternative techniques.

2 Context, Goals and Challenges

Following a discussion of course context and learning objectives, we will describe the remodeled course in detail, starting with participants, goals and milestones of the redesign and the steps performed in preparation, execution and postprocessing. Additionally, we will discuss two teaching methods employed during the course.

Regarding our goals, students should gain more practical access to model checking and how it connects to software engineering. Furthermore, the course should be connected to recent research results. Additionally, we want to encourage independent research. In particular, this includes supporting students throughout the common research phases from finding a suitable hypothesis to publication.

Switching from a classic lecture to inquiry-based learning resulted in a number of challenges we had to address, e.g., cognitive requirements are higher, as we switch from knowledge reproduction to production. Furthermore, course progression is less linear and project progress will be fluctuating and thus has to be monitored more closely. Of course, adding an additional programming task increases the workload for students. For grading, exams have to be prepared carefully to meet both the didactic requirements and the ones posed by exam regulations, due to the higher amount of group work and cooperation.

The given challenges have been addressed by different means during the course redesign. In the presentation, we will discuss the methods used to address them and how well they played out in the long run.

3 Course Preparation and Execution

We began course preparation by collecting and inspecting existing lecture material such as slides and exercise sheets. Furthermore, in order to gain additional material, the latest literature on model checking was sighted as well.

Reducing existing material to make room for the programming tasks proved difficult for two reasons. First, it was hard to anticipate the speed with which students would progress. Second, we intended for the module to be as open as possible. In particular, we wanted to avoid predetermining techniques and algorithms to be used. However, this made it impossible to anticipate what the students would come up with first.

Following the work by Barron et al. [1], we decided to start our course problem-oriented instead of immediately confronting students with the need to develop a research question themselves[1]. To do so, we followed a four-step approach to introduce model checking: (1) Error- and hazard analysis of an elevator control software, (2) introduce specification languages, (3) collaboratively develop key definitions such as state spaces and transitions and (4) collaboratively develop a simple explicit-state model checking algorithm.

Following, we switched to a more inquiry-based approach: For instance, the reduced input language did not include a way to specify certain system properties.

[1] For a comparison on inquiry-based and problem-based learning see, for instance, [6].

As students were writing their own software models in order to gain test cases, those shortcomings became obvious. In consequence, students had to come up with language extensions and develop new model checking algorithms for them.

The course was mainly driven forward by two methods complementing each other: A hazard collection used to keep the big picture in mind, and a Kanban board, which was used to structure the programming project into parts and to discover and tackle todos in an organized manner.

To motivate the how and why of model checking, we began the first lecture with a simplified hazard analysis. Working in groups, students were supposed to describe behavior scenarios of an elevator that performs as bad as possible. Scenarios were collected and sorted, first by grouping corresponding ideas such as "the elevator never opens its doors" and "the elevator never closes its doors".

We kept the collection around during the semester, removing those cards representing errors our model checker was able to detect. In consequence, the hazard collection was driving the inquiry-based learning process: While some errors were easy to detect using simple explicit-state model checking techniques, others made more involved techniques necessary, e.g., "If the elevator is moving, doors are closed" is comparably simple to verify, whereas "At some point in time the elevator is moving" is more involved due to time-based reasoning.

To manage programming tasks and how they are distributed between the participants we used a method based on simplified Kanban boards. Every occurring task is collected on a (virtual) flashcard. The method proved easy to learn for students and efficient in handling programming process.

The teaching methods used for the problem-oriented introduction as well as those used during the later sessions have been documented thoroughly in our article [5] and will be part of the presentation in greater detail.

To improve constructive alignment [2], we implemented a combination of formative and summative exams able to verify whether the learning objectives stated above have been reached. Both Kanban boards and Git history served as a documentation of participation that was considered for grading.

4 Course Evaluation

We followed three different approaches to course evaluation. First, we used peer review. Two sessions, one for research and development as well as one dedicated to knowledge synchronization, were monitored and reviewed by other faculty members and members of the didactics department. In the presentation, we will discuss the weaknesses discovered and how we adaptet the course to improve.

Second, we relied on classical course evaluations by students. However, due to the small number of participants, no official evaluation was performed by the university. For the same reason, there are no former evaluations we could compare to. Instead, we performed several intermediate evaluations using short online surveys. Feedback was very positive and encouraging. In particular, the course was described as both interesting and challenging at the same time. Furthermore, students evaluated their own learning achievements as high. The main point

of criticism was the volatile speed of progression during the lectures. As an immediate remedy, we started to monitor progress closely and intervened earlier.

Last, we tried to empirically evaluate student engagement, as one of the main challenges was to keep students motivated to participate in research and programming. Given that we used Git to record all changes made to the source code, we used the data collected for statistics on participation and engagement. Apparently, we were able to motivate participants to consistently and autonomously work towards the course goals, as we will show in the presentation by discussing different statistics performed on the data collected.

5 Conclusion

In summary, we believe that our new "model checking" course meets our self-set goals. Realization was more hassle-free than anticipated. In particular, we had no problems motivating students to active participation. We only had to intervene with speed and progress direction seldomly.

One of the key challenges proved difficult however. Due to the scale of the programming project, students had to work on different aspects, e.g., some were working on verification algorithms while others were working on performance improvements. However, all students were supposed to take the same exam.

As planned, we had regular meetings dedicated to synchronizing the different groups. However, specialist knowledge remains and is not distributed equally throughout the participants. While the exam and its results show that all participants reached the desired learning outcomes, we think that with a better focus on knowledge transfer an even better outcome could have been reached.

We believe that a few insights can be carried over to other courses both in model checking and in general and would thus like to present our work to a more general audience than before [5]. In particular, we believe that a strong connection between teaching and research is highly beneficial and can be achieved with little overhead. Depending on the scenario, a switch to inquiry-based learning is feasible and leads to high motivation among participants, without causing too much reduction in course content. However, focus has to be given to the synchronization of participants, i.e., teachers have to ensure that nobody is left behind.

The actual publication process, was not part of the course for several reasons. Nevertheless, three students were interested in writing a follow-up paper and presenting their work to the community. Even though we were unable to provide credit points, all three of them were very enthusiastic about the idea of writing their first article. To our joy, the paper was accepted for the *18th International Workshop on Automated Verification of Critical Systems* [7]. In consequence, our course shows that with the right motivation and proper support, inquiry-based learning methods can produce research results on a very high level.

References

1. B. J. Barron, D. L. Schwartz, N. J. Vye, A. Moore, A. Petrosino, L. Zech, and J. D. Bransford. Doing With Understanding: Lessons From Research on Problem- and Project-Based Learning. *Journal of the Learning Sciences*, 7(3-4):271–311, 1998.
2. J. Biggs. Enhancing Teaching through Constructive Alignment. *Higher Education*, 32(3):347–364, 1996.
3. E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.
4. O. Grumberg and H. Veith, editors. *25 Years of Model Checking: History, Achievements, Perspectives*. Springer-Verlag, Berlin, Heidelberg, 2008.
5. S. Krings, P. Körner, and J. Schmidt. Experience Report on An Inquiry-Based Course on Model Checking. In *Tagungsband des 16. Workshops zu Software Engineering im Unterricht der Hochschulen*, volume 2358 of *CEUR*, 2019.
6. A. Oguz-Unver and S. Arabacioglu. A comparison of inquiry-based learning (IBL), problem-based learning (PBL) and project-based learning (PJBL) in science education. 2:120–128, 07 2014.
7. J. Petrasch, J.-H. Oepen, S. Krings, and M. Gericke. Writing a Model Checker in 80 Days: Reusable Libraries and Custom Implementation. *ECEASST*, 2018.