A lightweight collaborative approach for teaching software project labs with industry partners

Christian Plewnia, Andreas Steffens, Nils Wild and Horst Lichter

Research Group Software Construction, RWTH Aachen University
https://www.swc.rwth-aachen.de

Abstract

Software project labs are an important part of computer science education. In 2014, we started to run labs together with industry partners to implement further learning objectives. In this article, we present these learning objectives and our organizational approach to realize a joint lightweight teaching with an industry partner. Furthermore, we report on our experiences with this approach, including the students' point of view. Finally, we discuss what we adapted to conduct our software project lab online during the Covid-19 pandemic.

Keywords

computing education, cooperative learning, industry cooperation

1. Introduction and related work

Universities educate computer science students so that they are well equipped for a successful career in industry or research. Accordingly, the Bachelor's program also and especially imparts practical skills. Courses such as Introduction to Programming and Software Engineering are a central part of the curriculum. In order to gain more practical experience, students typically complete at least one software project lab (SPL). It has special learning objectives and places special demands on the organization and implementation. In order to achieve these objectives, we have developed a lightweight and collaborative SPL format that introduces real world problems of industrial partners.

Prior research shows that students benefit from an involvement of industry partners in teaching of practical computer science courses [1, 2, 3]. Four often reported advantages for students are: (1) they have the opportunity to find an interesting employer afterwards through contact with industry [4]; (2) the motivation of the students may increase if the produced results are relevant for the industrial partner [4]; (3) they learn development techniques that are only applicable to larger systems and are difficult to teach [5]; (4) they learn to apply their theoretical knowledge to real problems [6].

Not only the students benefit from joint teaching, but also the industry partners. For example, the companies can attract future employees, prototypes for new ideas are developed, or new technologies are evaluated. Additionally, the companies benefit from the students' creativity [7].

The paper is structured as follows: First, we present the general and our specific learning objectives intended with SPLs. Next, we give an overview of our SPL format and describe its organization. In section 3, we discuss our experiences within the last seven years with this SPL format. Afterwards, our experience with online SPLs during the Covid-19 pandemic is presented and the results of student evaluations are discussed. Finally, we give a conclusion of our key findings.

Joint Proceedings of SEED & NLPaSE co-located with APSEC 2020, 01-Dec, 2020, Singapore
EMAIL: plewnia@swc.rwth-aachen.de (C. Plewnia); steffens@swc.rwth-aachen.de (A. Steffens); wild@swc.rwth-aachen.de (N. Wild); lichter@swc.rwth-aachen.de (H. Lichter)
ORCID: 0000-0003-0221-9842 (C. Plewnia); 0000-0002-3440-1238 (H. Lichter)

2. Learning objectives of the software project lab

The SPL is typically placed after the Programming and Software Engineering lectures in the Bachelor's curriculum. Usually, it is conducted during one semester, i.e. the students attend other courses in parallel, and has a workload of six credit points, which corresponds to 180 hours of work. At many universities, the SPL is organized decentrally. In this case, each chair defines its specific SPL based on the general learning objectives described in the module handbook. Students can then choose from all offered labs. This decentralized approach allowed us to tailor the format of the SPL.

SPLs should achieve multiple equally important general learning objectives. These are

- O1 the application of software engineering methods and techniques,
- O2 the application of programming languages and technologies,
- O3 the development of a larger software system,
- O4 the application of modern software development tools and environments,
- O5 the documentation and presentation of produced results, and
- O6 team-based planning and coordination of tasks.

Although these general SPL learning objectives are challenging, a very important aspect is missing: working in a realistic environment including communication with an (external) client. We believe that this also contributes very strongly to achieving the general objectives, especially in the areas of communication, planning, and presentation.

Therefore, we developed our own SPL concept in 2014 based on already published SPL teaching approaches and all learning objectives. The overall goal was to develop a collaborative joint teaching approach with industry partners without causing significant additional collaboration effort for neither the partner nor us. This distinguishes our approach from approaches that require more involvement of the industry partner, such as described by Katz [8].

With our approach for the implementation of SPLs we pursue the following special learning objectives that go beyond the general ones:

- S1 Students should learn from each other in the sense of peer learning as described by Gogus[9].
- S2 Students should learn to familiarize with an unknown domain and work in a realistic development environment.
- S3 Students should learn to work in an agile development process.

In addition, our approach should meet the following two organizational requirements:

- R1 Our effort to collaborate with the industry partner should be low.
- R2 The effort for the industry partner should be defined in advance so that appropriate resources can be planned.
- R3 A topic and environment should be provided that motivates the students to push their limits.

3. Organization of our software project lab

The SPL is divided in the phases *preparation*, *execution*, and *transition* as depicted in figure 1.

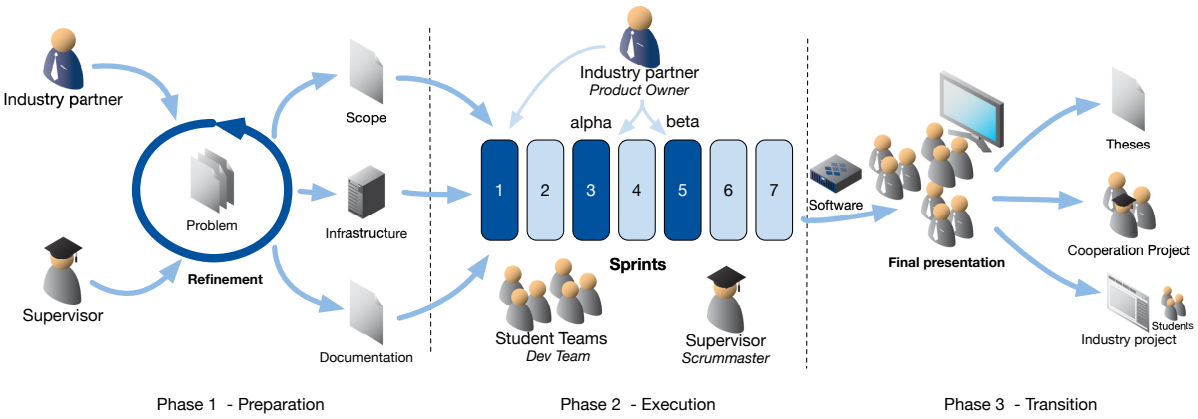

Figure 1: Phases of our SPL

3.1. Phase 1 - Preparation

Here, the most important goal is to find a suitable topic and derive a task description. The industry partner provides a general problem description, which is refined in an iterative process with the supervisors. Finally, a scope is defined including the boundaries of the problem, general architectural decisions and requirements in terms of data, infrastructure, software, and knowledge. Based on the task description, the industry partner designates at least one representative, who is accountable for the decisions to be made as a stakeholder in the SPL. In addition, the industry partner provides all necessary input, e.g. data documentation, infrastructure, and credentials for APIs. The last task is to schedule the mandatory milestones and meetings during the execution of the SPL.

3.2. Phase 2 - Execution

The SPL execution is based on the well-known Scrum process framework, as it is lightweight and defines only a small set of roles, artifacts, and rules. Research shows that introducing agile methods like Extreme Programming or Scrum leads to positive effects, e.g. increased awareness for software quality or higher motivation due to more freedom and responsibility [10, 11, 12].

Scrum defines three roles: (1) the product owner, who provides the information to help the development team to understand and work on the needed stories; (2) the development team, which implements the software; and (3) the Scrum master who enforces the rules of Scrum and continuously supports the other roles. We mapped these roles to the stakeholders of the SPL. The product owner role is played by the industry partner, the organizing supervisors act as Scrum masters, and the students represent the development team. We do not introduce more roles as we want to encourage a high degree of self-organization inside the development team. In order to integrate the industry partner as loose as possible, the roles have to be slightly adapted. The responsibility of maintaining the backlog is transferred to the development team, which is a valid adaption according to the Scrum guide [13]. The second adaption is the proxy function of the Scrum masters. While the industry partner cannot fulfill the product owner role full-time, the Scrum masters can organize and streamline the communication with the industry partner and answer upcoming questions by themselves.

The execution phase is divided in seven sprints, each lasting 14 days. The following five major milestones and events are defined:

1. **Meeting with product owner** - Before the first sprint, the industry partner introduces the

task description and relevant aspects of the domain to the students. Arising questions can be answered and discussed immediately.
2. **Alpha milestone** - After the third sprint the teams present their preliminary results. Based on the insights gained during the presentations, the industry partner gives feedback and may even suggest requirement changes.
3. **Beta milestone** - After the fifth sprint the teams present their first working prototype to the industry partner and receive more feedback.
4. **Feature freeze** - After the sixth sprint the teams stop developing new features in order to focus on the stability and quality of the prototype in the last sprint.

The daily Scrum meetings are omitted, as they are impractical for students attending other courses in parallel. Instead, all teams meet with the supervisors at two mandatory in person appointments per week. There, the students organize their work, discuss problems and questions, or just continue with the work. This is an efficient way to communicate and ensures that the teams effectively organize their work with minimal overhead. If necessary, these meetings can also be used for talks given by the supervisors, the industry partner, or external speakers in order to convey knowledge about technologies, the domain, or soft skills. During the sprints, the supervisors are constantly available for support and answering questions. At the end of each sprint, each team presents its latest results to the other teams.

3.3. Phase 3 - Transition

After the last sprint, the results are presented and demonstrated in front of a bigger audience organized at a site of the industry partner. This final event enforces the students to provide a deliverable prototype, which can actually be handed over to the industry partner. As usually stakeholders from management or business development, prior unfamiliar with the actual scope of the SPL, attend this event, the teams have to consider this when preparing their presentation.

During this event, the students will also reflect on the complete SPL and their lessons learned and the gathered insights regarding professional software development. The industry partner will provide valuable feedback on the achieved results and the experienced process.

As the SPL is based on a problem provided by the industry partner, the SPL may lead to a variety of different activities even after its completion. First, the industry partner can decide to follow the produced results and set up an internal project on the topic, which often involves recruiting some of the SPL students as student workers. Furthermore, follow-up joint Bachelor theses may be defined. Finally, due to the highly collaborative nature of the SPL, the industry partner may start further research activities with the organizing chair.

4. Experience report

Since 2014 we have conducted 11 SPLs in collaboration with five different industry partners in which about 150 students participated. The topics varied with the industry partner and included the domains public transportation, insurance, energy market and travel[1]. Here, we report our experiences with respect to the previously stated objectives.

The preparation phase usually began six months in advance. In our experience an early start is needed to have enough time to define the topic and responsibilities and to provide the tools and data

[1]Task descriptions of all labs can be found at https://www.swc.rwth-aachen.de/software-project-labs

needed to work on the task. The latter is the responsibility of the industry partner. Furthermore, the milestone meetings were planned early to avoid conflicts in the schedule later on.

The SPL topics proposed by the industry partners were refined in cooperation to fit to the SPL objectives. In our experience, this helps to keep the topic selection process as simple as possible and to define clean responsibilities (R1, R2). However, it should not be forgotten that the task must be solvable within the SPLs time frame. To achieve this, we always provide a first sketch of the top-level architecture in terms of central components or sub-systems, as most of the students are inexperienced in designing quite complex software systems. The top-level architectures were jointly developed with the industry partners.

With first-time collaborations we encountered the problem that the students expertise is misjudged. In many cases, the students were seen similar to external service providers or contractors. During the topic selection refinement, we made sure that the students were expected to deliver a proof of concept and not a ready-to-use product. Relevant and exploratory problems from the industry that can be solved applying modern technologies proved to be very good candidates for SPL topics and in addition led to an increased intrinsic motivation of the students, in our experience.

Once the lab has started, students needed some time to familiarize themselves with the domain and its associated data and APIs. In our experience, real data and APIs led to the best results. Nevertheless, mocks were beneficial if real APIs were to complex or contained confidential data. Using mocks in these cases made it easier to focus on the given domain problem. It is our responsibility to find a balance, such that the students get to know a relevant domain problem (R2) and keep the motivation high (R3).

The technical part was usually perceived as less challenging by the students. Nevertheless, in each lab there was a strong variation among the students with respect to their practical experience as well as technology and tool knowledge. In our opinion, to support peer teaching (S1) it is important to avoid pure expert teams as this may result in other teams consisting of inexperienced students only. In recent labs, we started to survey the students' knowledge in the first meeting in order to identify experts, i.e. students with advanced knowledge. We then distributed the experts across the teams. We experienced that in the beginning the less advanced students learned much from the experts, especially about programming languages and technologies (O2) as well as software tools and environments (O4). The regular presentations challenged the other teams, but also led to a transfer of ideas, knowledge, and experiences across the team borders, as the teams could compare their solutions and ideas. They also contributed to the objective of documenting and presenting the results (O5) and became more professional throughout the SPL and milestone meetings.

Our SPL format encourages to implement a Scrum-based process, as intended by our third objective (S3). Our decision to prescribe Scrum provided the students with a framework that helped them getting started working agile. In the first two sprints, the students were always focused on learning what to build and how to implement it by familiarizing themselves with necessary technologies. The first artifacts delivered by the teams ranged from simple sketches to first prototypical implementations. The organization of the work is the responsibility of the team. The produced artifacts were used in the alpha milestone meeting to discuss with the industry partner whether they head in the right direction. In our experience, the students often realized any misconceptions in the alpha milestone meeting and then continued with a more clear understanding of the goals and how to achieve them. Throughout the SPL, the students started to specialize with regards to certain technologies for frontend, backend, or database development. As a result, the students started to use interface definitions on their own, to communicate across specialized groups within the team. Consequentially, the teams improved their ability to define tasks for the backlog, choose the next sprint's tasks, and assign the work and also improved their velocity. Thereby, they learned about the application of software

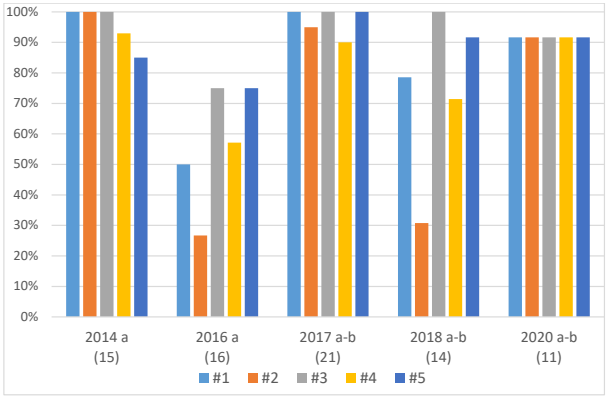

Figure 2: Approval to learning objectives

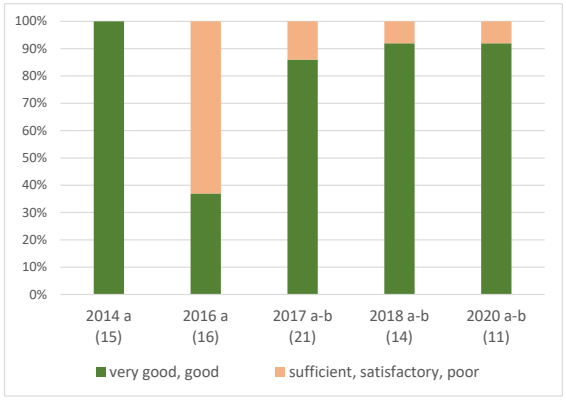

Figure 3: SPL concept rating

engineering methods and techniques (O1) and team-based planning and coordination of tasks (O6).

For our SPL we decided, that all teams work on the same task. We argue that working in parallel on the same task facilitates the transfer of knowledge and ideas inside and also between the different teams. This competitive element of the SPL increases student motivation and animates the teams to produce some kind of a unique solution to stand out. The final presentation at the industry partner's place emphasized this assumption. On the one hand, the students appreciated the feedback of non-academic staff. On the other hand, the employees of the industry partner were intrigued by the different and sometimes unconventional solutions the students presented. In many cases, the contacts made were further intensified in the form of bachelor theses, student jobs, or further development of the results.

5. Student evaluation

Each of our SPLs was evaluated by the students. Below, we discuss some evaluation results obtained. We refer only to the data of eight SPLs conducted in 2014/16/17/18/20, since the evaluation data of 2015 and 2019 could not be collected completely. Among other things, students were asked to rate the following important learning objectives: (#1) In the lab I can apply what I have learned; (#2) The given tasks motivate me to work on them; (#3) I consider what I learned in the lab to be important; (#4) I have improved my programming skills; (#5) I can apply the knowledge I have acquired in the lab to new tasks. These objectives could be rated from 1 (strongly agree) to 5 (strongly disagree).

Figure 2 visualizes the percentage of the students rating them with "agree" or "strongly agree"; the number of students who participated in the respective evaluation is shown in brackets. It can be observed for most SPLs that the students highly (< 80%) approved most of these learning objectives. It is also visible that the right choice of an SPL topic is crucial in order to motivate and challenge the students; we did not succeed in doing this well enough for the SPLs in 2016 and 2018. Overall, however, our SPL concept is evaluated positively by the majority of the students (see figure 3)

This is also supported by the following quotations taken from the evaluation sheets: *"I would have liked to have had more practical modules in this form during my studies, where you work on a project with a team and set milestones and goals more independently"* and *"It was a lot of fun to work on a project in a team, just as it would have been in the real working life"*.

6. Online lab during the Covid-19 pandemic

In 2020, in an effort to fight the Covid-19 pandemic the RWTH Aachen University decided to teach the courses online whenever possible instead of in person at the university. This also meant that we had to adapt our SPL format. Essentially, the SPL could be carried out as in the years before, with the exception of the two mandatory meetings per week, that we hosted using video conferencing software instead of conducting those in person. We recommended the students to enable their webcam for these meetings if possible; this way the communication was enriched by facial expressions and gestures. Though we learned that for an SPL conducted online instead of in person, the communication within and across the student teams was more challenging and the students' work became less transparent to us. As we require all students to show their work in the frequent presentations, we were nevertheless able to follow the progress. We also tried to make sure to talk to every student in the mandatory meetings. However, in the online format, it is harder to support individual students if they do not approach their supervisors. We also learned that online courses grant more flexibility for the student's project management. The students reported back that they were able to meet up more frequently as they didn't had to attend other courses at a specific time. The biggest disadvantage in our opinion was the lack of direct contact between students and the industry partner. This especially affected the final presentation that could not take place on site. Therefore, there was no networking after the presentation, where the employees of the industry partner and the students could get in contact. This should be made up for as soon as possible.

7. Conclusion

In this paper we presented the design of our lightweight collaborative approach for teaching software project labs (SPLs) with industry partners. This approach implements the learning objectives defined by our computer science module handbook, i.e. engineering a larger software project as team applying modern methods and tools. Our approach even goes beyond that by requiring the students to learn to familiarize with an unknown domain introduced by the industry partner, to work in a realistic environment following an agile software development approach, and to learn from their peers. Besides providing a topic for the SPL, the industry partner also plays the role of the product owner, meeting them at a few occasions during the lab and providing feedback. Further, the student teams present their final software product to the industry partner.

We taught 11 SPLs using this approach. In our experience, a topic from industry rather than academia raises the students' interest and the industry partner can benefit from surprising solutions and proof of concepts developed by the students. Further, our lightweight approach ensures that the effort for the industry partner and us is kept low. This is important as we do not reuse topics in subsequent SPLs and thus rely on the will of industry partners to collaborate; however, our history of SPLs shows that many industry partners saw the benefits of the collaboration and, according to the evaluation feedback we received, so did the students.

Based on our experience the following practices heavily contribute to the success of SPLs conducted with industry partners: (1) start the preparation early; (2) select the SPL topic carefully; (3) use real data whenever possible; (4) provide mocks if needed; (5) prescribe a simple development process which enables self-organization of teams; (6) let student teams work on the same topic in parallel.

While we are convinced that this SPL approach has its unique benefits, we have only little empirical evidence to support our claim. When writing this paper, we realized that the general evaluations done for every course at our university need to be adapted to this special format to provide meaningful

evidence. We intend to implement these changes for future SPLs.

Acknowledgments

Thanks to our industry partners *Amadeus*, *Generali Informatik Services*, *ITERGO*, *IVU Traffic Technologies*, and *Kisters* to support our SPL students.

References

[1] A. J. Kornecki, I. Hirmanpour, M. Towhidnajad, R. Boyd, T. Ghiorzi, L. Margolis, Strengthening software engineering education through academic industry collaboration, in: 10th Conference on Software Engineering Education and Training, CSEE&T, Virginia Beach, Virginia, USA, April 13-16, 1997, pp. 204–211. doi:10.1109/SEDC.1997.592455.

[2] D. Tahmoush, S. Fouché, S. McMaster, J. Stuckman, J. Purtilo, Enhancing software project management courses with industry participation, in: International Conference on Frontiers in Education: Computer Science & Computer Engineering, FECS 2009, July 13-16, Las Vegas, Nevada, USA, 2009, pp. 135–140.

[3] B. Penzenstadler, M. Mahaux, P. Heymans, University meets industry: Calling in real stakeholders, in: 26th International Conference on Software Engineering Education and Training, CSEE&T, San Francisco, CA, USA, May 19-21, 2013, pp. 1–10. doi:10.1109/CSEET.2013.6595231.

[4] K. C. Williams, C. C. Williams, Five key ingredients for improving student motivation, Higher Education Journal 12 (2011) 121–123.

[5] G. Gabrysiak, M. Guentert, R. Hebig, H. Giese, Teaching requirements engineering with authentic stakeholders: towards a scalable course setting, in: Proceedings of the First International Workshop on Software Engineering Education Based on Real-World Experiences, EduRex 2012, Zurich, Switzerland, June 9, 2012, pp. 1–4. doi:10.1109/EduRex.2012.6225708.

[6] B. Bruegge, S. Krusche, L. Alperowitz, Software engineering project courses with industrial clients, ACM Trans. Comput. Educ. 15 (2015) 17:1–17:31. doi:10.1145/2732155.

[7] G. Smith, A. Hjalmarsson, H. Burden, Catalyzing knowledge transfer in innovation ecosystems through contests, in: 22nd Americas Conference on Information Systems, AMCIS 2016, San Diego, CA, USA, August 11-14, 2016.

[8] E. P. Katz, Software engineering practicum course experience, in: 2010 23rd IEEE Conference on Software Engineering Education and Training , CSEE&T, Pittsburgh, PA, USA, March 9-12, 2010, pp. 169–172. doi:10.1109/CSEET.2010.38.

[9] A. Gogus, Peer Learning and Assessment, Springer US, Boston, MA, 2012, pp. 2572–2576.

[10] C. Bunse, R. L. Feldmann, J. Dörr, Agile methods in software engineering education, in: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 284–293.

[11] J. Schneider, R. Vasa, Agile practices in software development - experiences from student projects, in: 17th Australian Software Engineering Conference (ASWEC 2006), 18-21 April, Sydney, Australia, IEEE Computer Society, 2006, pp. 401–410. doi:10.1109/ASWEC.2006.9.

[12] Z. Masood, R. Hoda, K. Blincoe, Adapting agile practices in university contexts, Journal of Systems and Software 144 (2018) 501–510. doi:10.1016/j.jss.2018.07.011.

[13] J. Sutherland, K. Schwaber, The scrum guide - The definitive guide to scrum: The rules of the game, Scrum. org, 2017.