7-4-2022

Software Engineering Education with Industry-Mentored Projects

Raine Kauppinen
Haaga-Helia University of Applied Sciences, raine.kauppinen@haaga-helia.fi

Altti Lagstedt
Haaga-Helia University of Applied Sciences, aaelag@utu.fi

Juha P. Lindstedt
Haaga-Helia University of Applied Sciences, juha.lindstedt@haaga-helia.fi

Ohto Rainio
Haaga-Helia University of Applied Sciences, ohto.rainio@haaga-helia.fi

Recommended Citation

Kauppinen, Raine; Lagstedt, Altti; Lindstedt, Juha P.; and Rainio, Ohto, "Software Engineering Education with Industry-Mentored Projects" (2022). *PACIS 2022 Proceedings*. 134.
https://aisel.aisnet.org/pacis2022/134

Software Engineering Education with Industry-Mentored Projects

Completed Research Paper

Raine Kauppinen
Haaga-Helia University of Applied Sciences
Ratapihantie 13, 00520 Helsinki, Finland
raine.kauppinen@haaga-helia.fi

Altti Lagstedt
Haaga-Helia University of Applied Sciences and University of Turku
Ratapihantie 13, 00520 Helsinki, Finland
altti.lagstedt@haaga-helia.fi

Juha P. Lindstedt
Haaga-Helia University of Applied Sciences
Ratapihantie 13, 00520 Helsinki, Finland
juha.lindstedt@haaga-helia.fi

Ohto Rainio
Haaga-Helia University of Applied Sciences
Ratapihantie 13, 00520 Helsinki, Finland
ohto.rainio@haaga-helia.fi

Abstract

Many educational software engineering (SE) projects are done with little or no real outside involvement. Also, with an outside customer, the objective is typically to produce a demo, proof of concept, or minimum viable product. These lack real pressure from outside stakeholders because technological solutions are often agreed on based on a curriculum, and implemented features can be negotiated based on the students' skills and the course schedule. Therefore, typical industry–education cooperation SE projects hardly resemble the real SE projects carried out in software companies. To overcome these limitations, we developed and tested a university–industry model where real-life development pressure coming from real customers and a software provider was generated to suit educational purposes. The model was beneficial for all parties and produced a more realistic learning experience compared with other project-based models. However, the model also required additional support, such as facilitation and mentoring for the students.

Keywords: Software development techniques, collaboration in software development, computing education

Introduction

Whereas the basics of software engineering (SE) programming are rather general and straightforward to teach, information system (IS) development and digitalization-related education at advanced levels face many challenges. Up to a certain point, the theories and practices of digitization and IS development can be taught via lectures and group exercises. However, when there is a need to practice the complexity of a real project, theoretical examination or simplified exercises are no longer enough. In reality, IS development projects are complex group projects with multiple stakeholders and critical connections between software developers and business. It is not enough for developers to master programming; rather, understanding and considering these connections in real-life situations are also necessary elements.

In the literature, these connections are visible in systems-lifecycle-related development models or standards, such as the systems development lifecycle (SDLC) model (Bojic et al. 2019), business technology model (BTM) (Business Technology Forum 2021), and expert-oriented digitalization (EXOD) (Raine Kauppinen et al. 2020), but to achieve practical skills, it is important that these connections are visible in exercises and student projects as well. If the related education is done without incorporating these viewpoints, there is a high risk that educational SE projects will see an emphasis placed on individual-level learning objectives and schedule based on the curriculum, leaving out a lot of actual system(s) engineering and digitalization-related content.

It is important for students studying to be developers to understand that there are parts in the projects they can control and parts they must adapt to. In real-life projects, there are limitations and restrictions stemming from the environmental context, that is, outside of the development team (Avison and Fitzgerald 2003). These restrictions and limitations can relate to resources, schedules, objectives, development methods, practices, tools, and technologies, as well as to acceptance of results and to final quality assurance. These restrictions dictate what is possible to do, in which timeframe, and whether the results are acceptable.

In educational projects, real-life dynamics and pressure are missing if the projects are done without stakeholders from outside the school. To bring real life to teaching, a wide range of business cooperation projects (so-called capstone projects) have been implemented in teaching, and it is rather typical for advanced SE courses to have projects with a real company as a customer (see, e.g., Burge and Gannod 2009; Fornaro et al. 2007). However, in projects with a customer from outside the school, the project purpose is often to produce some kind of demo, proof of concept (POC), or minimum viable product (MVP) on the course schedule. These kinds of projects often lack typical real-life restrictions, such as required technological solutions or critical features. Instead, they are based on the curriculum and can be negotiated flexibly. Typically, the customers are not software development organizations and are not competent to guide and control the work of students. Thus, the role of clients is to be a source or requirement, and other guidance is given most often by the teachers of the course (see, e.g., Burge and Gannod 2009; Chase et al. 2007; Christensen and Rundus 2003; Fornaro et al. 2007; Taran et al. 2008). In some cases, software companies are used as customers, and they may give guidance in workshops (see, e.g., Ahmad and Liukkunen 2019); however, companies still have a loose connection to the development work. In addition, it may be that customers from outside are actually not that interested in the developed product either; instead, they are more interested in cooperating with the university or looking to find skillful students to recruit (Christensen and Rundus 2003; Fornaro et al. 2007; Taran et al. 2008).

Typical educational projects may be a good way to learn teamwork and technologies, as well as methodologies. However, if we want students to learn and overcome the challenges of digitalization projects, these projects are not effective. Although the success and challenges of capstone projects have not been studied extensively (Vanhanen et al. 2018), some main problems have been presented. Vanhanen et al. (2018) pointed out that the main problems relate to students' technical capability and competencies on the one hand and to their ability to manage (and understand) the tasks to be performed on the other hand (Vanhanen et al. 2018). It is understandable that students have gaps in their skills, and it is challenging to make technological decisions from scratch and account for maintenance as well. In addition, the process of digitalization involves making and managing process change with the help of IS, and mastering digitalization is not always an easy task. It is important to know how to cope with unexpected changes and how it feels when the business of all parties depends on the work being done. There is a clear need for a new industry-mentored approach in which students are more involved in implementing change in the middle of real discussions, not just as a requirements-implementing, code-producing group.

To answer the need outlined above, we used practice driven approach and developed a new kind of cooperation model, where students' results are vital for involved parties with a critical role in development, and where industry partners are committed to mentoring the students. In this new kind of model, we see it as essential to have both a real client outside the software development course and a real software company as a mentor for the course, and in that way to have some kind of development–operations (DevOps) perspective in the capstone course as well. Building on the results from our earlier work on EXOD (Kauppinen et al. 2020), we developed the model using a case where we had a real university digitalization project with clear objectives and a tight schedule, as well as a software provider committed to the further development and maintenance of the developed IS. Our research questions are as follows:

RQ1: How can SE education be combined with the real-life challenges and practices of a digitalization project with real customers and real software development companies?

RQ2: What are the effects on learning when this kind of approach is applied?

In sections 2 and 3, we go through general principles of SE education and digital transformation education, which we see as the context for IS development. In section 4, we describe the target, the case, and the methodology used. We then present the main results in section 5; in section 6, which comprises the discussion and conclusion, we answer the research questions and give our concluding remarks.

Software Engineering Education

SE education focuses on predefined levels in qualifications-based frameworks, such as the European qualification framework (EQF) (European Parliament and Council of the European Union 2008), and learning taxonomies, such as Bloom's revised taxonomy (Anderson et al. 2001). In the higher education setting in Finland, for example, studies at universities of applied sciences should reach EQF level 6, meaning high-level vocational expertise, at the curriculum and degree level. In Bloom's revised taxonomy, advanced-level education should also reach the higher levels (applying, analyzing, evaluating, and creating) instead of the lower levels (remembering, understanding), which should already be covered, at least mostly, in earlier stages of education.

In advanced-level SE education, after such basic-level courses as programming, the target is mostly on the higher levels of learning taxonomies. This part of SE education can be seen as a capstone part of the studies, considering stakeholder needs (Tafliovich et al. 2019; Todd and Magleby 2005). In addition, advanced-level SE education is crucial in reaching the curriculum and degree-level targets related to qualifications. From this viewpoint, the risk related to the challenges in advanced-level SE education discussed in section 1 is that because of the lack of practical real-life-related group projects with actual organizational information technology (IT) and business connections, the learning gained does not reach the levels targeted in a course or curriculum.

Qualification-based frameworks and learning taxonomies do not specify the specific content of higher education even though they have focus on advanced-level education as well. This is because they are independent from the specific field of education. For the content, there are both general and specific frameworks applicable in the field of SE. A general European-level framework is European Skills, Competences and Occupations (ESCO) (European Commission 2020); examples of more specific ones are Software Engineering Body of Knowledge (SWEBOK) (Bourque and Fairley 2014) and Computing Curricula (Clear et al. 2019). For defining the content in the SE curriculum and courses, documented role-based competency requirements found, for example, in the BTM (Business Technology Forum 2021) can also be utilized. The curriculum recommendations issued by the Association for Computing Machinery (ACM) (ACM 2021) have also been of great importance when SE education is developed. In their curriculum recommendations, the need for real-life experiences is recognized and highlighted as one of the things to be considered in education. For example, the ACM's Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (Ardis et al. 2015) places significant emphasis on capstone projects, whereas the ACM's Competency Model for Undergraduate Programs in Information Systems (Leidig and Salmela 2020) points to the IS practicum as one of the main competence areas, and capstone projects are seen as one way to implement IS practicums in education.

When planning implementation of the courses, there are several examples of feasible pedagogical approaches for higher education in general, such as problem-based learning (PBL) and project-based learning (Akınoğlu and Tandoğan 2007; Perrenet et al. 2000). PBL is a learning method centered on developing the student's active learning and problem-solving skills, as well as knowledge of the subject matter (Akınoğlu and Tandoğan 2007). In contrast, project-based learning is an individual or group activity that goes on over a length of time, resulting in a product, presentation, or performance. It typically has a timeline, milestones, and other aspects of formative evaluation as the project proceeds (Donnelly and Fitzmaurice 2005). The main differences between PBL and project-based learning are as follows: First, project-based learning typically has a longer time span compared with PBL because the projects typically last for several months. Second, it is typically carried out at the end of the studies, building on the skills acquired in earlier studies—including soft skills—because it resembles real professional projects, whereas PBL has more focus on knowledge acquisition (Perrenet et al. 2000).

Based on the target level of learning and the contents of advanced-level SE education, both PBL and project-based learning are feasible (Richardson and Delaney 2009; Savery 2006; Woodward et al. 2009). In addition, several other approaches have been applied, including emphasis on educational aspects and agile software development methods (Hsu 2019; Kauppinen and Lagstedt 2011; Tafliovich et al. 2019). Some approaches also incorporate industry or stakeholder perspectives to some extent (Bruegge et al. 2015; Dagnino 2014; Penzenstadler et al. 2013).

Since real-life industrial SE projects have non-trivial problems to solve (Bojic et al. 2019), higher level SE education should have characteristics of both PBL and project-based learning. In addition, as discussed in section 1, real-life SE projects often have multiple other stakeholders involved and connections to other projects in addition to the group of developers in the project group (Business Technology Forum 2021).

However, PBL, project-based learning, or incorporation of industry or stakeholder perspectives as such does not guarantee the targeted levels of qualification or learning in SE education settings, nor does it guarantee a focus on intensive industry cooperation with complex real-life challenges and practices for development projects with the element of real-life pressure. Thus, even if these approaches are used, without more intensive industry involvement, it will be difficult to ensure students understand, for example, the real business (process) development aspects that are an essential part of digitalization (Lagstedt et al. 2020) or face the contradictory needs of stakeholders and change resistance, which are also typical in real-life digitalization projects. In this kind of setting, it is also important to provide additional support for the students, such as in the form of mentoring (Alred and Garvey 2019), which moves SE education toward cooperative effort between education and industry (Maguire et al. 2019).

Digital Transformation Education

Digital change is one of the major reasons for IS development, as well as an influential environmental element developers face during IS development. Digital changes in organizations can be divided into the three following levels: digitization, digitalization, and digital transformation (Table 1).

Table 1: Different levels of digital change

Evolution	Digitization	Work automatized with information system (IS), no big process changes.
	Digitalization	Business process development / re-engineering with help of a new IS.
Revolution	Digital transformation	Whole business area/unit and related ISs are redesigned. Processes can be developed, dropped, or created.

These levels are in line with the categories of substitution, extension, and transformation presented by Pihir et al. (2018). If applying Venkatraman (1994) classification, it can be stated that digitization represents an evolutionary development approach, whereas digitalization and especially digital transformation represent a revolutionary approach. In education, with traditional project learning practices, it is challenging to achieve the revolutionary level where real process and organizational changes are implemented. However, since digitalization and digital transformation are changing the processes and practices in all business areas, and since they are important tools to enhance the efficiency of organizations (Borg et al. 2018, 2020; Pihir et al. 2018), it is important that their essential aspects are taught as part of higher education studies. Since digital transformation is a large concept that is challenging to fully implement in courses, we concentrate here on digitalization, which has elements similar to digital transformation.

In digitalization, the three following interrelated requirements can be stressed (Raine Kauppinen et al. 2020):

1. Digitalization must provide usable and helpful tools for end users.
2. Developed tools and practices should provide data for decision makers.

3. Digitalization is not just automatizing existing processes; there should be process improvements as well.

All of these points are important, and the targets of digitalization should be considered in this order. Kauppinen et al. (2020) pointed out that it is not possible to reach reliable and usable data unless systems are considered useful tools to help daily work. Processes can be improved when reliable data are available. Digitalization consists of two parallel, interdependent activities—business process development and IS development (Dahlberg and Lagstedt 2021)—which are further discussed in the following subsections.

Business Process Development

Typically, in educational projects not based on a real business case, business process development has little significance. If discussed, it is something that happens in the background. However, in real-life projects, developers are a part of business development.

In business process development, the importance of not only automating existing processes but also open-mindedly considering new solutions (process re-engineering) has been emphasized for a long time (Davenport and Short 1990; Hammer 1990). Different approaches for business process development and selection models have been presented (vom Brocke et al. 2021; Gross et al. 2021). Business processes can be developed via episodic plan-driven methods or more ongoing development supporting change-driven (agile) methods (Davenport 2010; Hammer 2010). A common element for all approaches is to define the starting situation (as-is) and plan the desired situation (to-be) (Rosemann and vom Brocke 2010), but often, the guidance on to-be process creation is limited (Gross et al. 2021), making it difficult to define concrete steps for process change.

Since most process development practices lack guidance for to-be process creation (Gross et al. 2021), politics, motivation, and communication play essential roles in organizational transformation (Harmon 2010). Thus, in process development, it is not sufficient to consider only the organizational level; people and culture must be considered as well (vom Brocke and Sinnl 2011). People are often more complex than expected and sometimes difficult to predict; they are neither fully rational (Simon 1997) nor always reliable (Argyris 1977). To cope with this, it is important to have change management practices in use as well.

Information System Development

From a control point of view, IS development methods (ISDMs) can be classified into two categories—plan-driven and change-driven methods (Moe et al. 2012). Plan-driven ISDMs were dominant at the end of the 20th century, whereas the popularity of change-driven ISDMs has grown over the last two decades such that they appear to be mainstream today (Theocharis et al. 2015). It is also possible to use a hybrid approach combining parts of plan-driven and change-driven development (Theocharis et al. 2015). Because no method fits all cases, it is important to select the method on a case-by-case basis (Lagstedt and Dahlberg 2018).

In plan-driven IS development, planning and development are divided into separate phases. The assumption is that every aspect of development work (objectives and required metrics, tasks, time, money, and resources) can be planned thoroughly and in advance. Development begins after the planning phase is completed. Plan-driven methods, such as the waterfall method, are a straightforward way to develop software, but there are many known problems; for example, early mistakes that are found late become difficult and costly to resolve. The assumption is that no changes occur during software development. Even if all the requirements are formulated correctly, success in IS development is not guaranteed because circumstances might have changed along the way (Hansen and Lyytinen 2010).

In change-driven development, such as in agile methods, the idea is that the whole IS is not planned at once; rather, planning, and development are undertaken in small incremental steps. After each step, the situation is re-evaluated, and any necessary changes are made to the objectives. Each development step results in a new release of the IS.

The change-driven approach is not problem-free. Because of the nature of the development, it is highly likely for radical changes in the code to occur during the development. These unplanned changes cause incoherencies in the software architecture; if these incoherencies are not resolved during the development step, they become technical debt (Cunningham 1992), causing more development and maintenance challenges later. Moreover, if the client has no clear vision and priorities change constantly, or if there is no shared understanding of what is to be delivered, the scope of development becomes unclear and quality assurance becomes challenging (Dahlberg and Lagstedt 2018; Moe et al. 2012). Despite the rather high success rate of projects undertaken using agile methods, 61% of them are still not considered successful (Hastie and Wojewoda 2015), meaning that agile ISDMs do not guarantee the success of IS development projects either (Dahlberg and Lagstedt 2018).

Methodology and Case

In this study, we applied action design research (ADR) as an apparent choice, hence three of the researchers were actors in the project, and an external view was not possible. ADR consists of the following four stages (Sein et al. 2011): 1) problem formulation; 2) building, intervention, and evaluation; 3) reflection and learning; and 4) formalization of learning (Figure 1). The context is advanced-level SE education, and the environment is Softala, the part of Haaga-Helia University of Applied Science (Haaga-Helia UAS) responsible for advanced-level SE education.

Here, the trigger for stage 1 comprises the challenges perceived in the SE education context discussed in section 1. Thus, for stages 2–4, the target of the intervention is capstone course Softala Project II, and the artifact developed and evaluated is the implementation of that course.

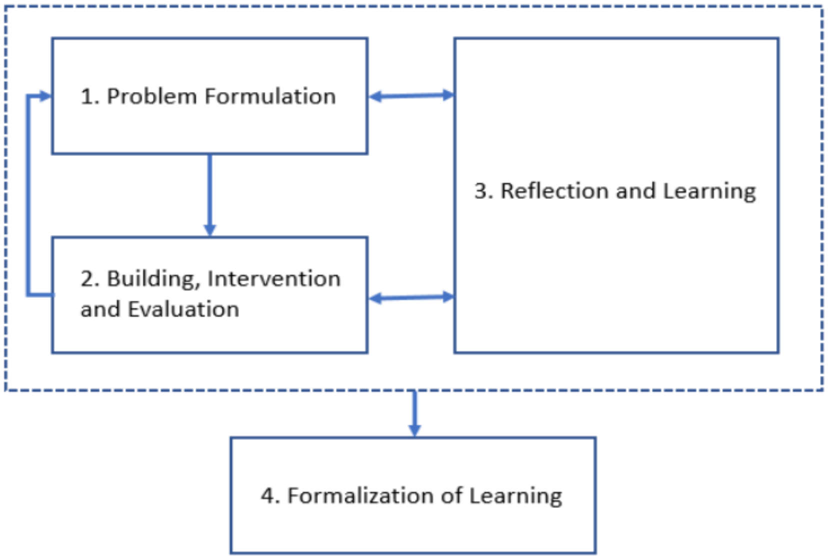
1. Problem Formulation
2. Building, Intervention and Evaluation
3. Reflection and Learning
4. Formalization of Learning

Figure 1: Four stages of action design research (ADR) (Sein et al. 2011)

Development Target and Timeline

At Haaga-Helia UAS, the Softala Project II is a project-based capstone course for final-semester students in the business IT curriculum doing their bachelor degree. In the course, they build software for real client commissions using current technologies such as Node.js, ReactJS and Docker. The course enables the students to apply their SE skills from problem definition to software deployment and handover. This kind of course at the end of bachelor studies is relevant, since the students are expected to be ready to work in the SE field after graduation before possibly applying master level studies some years later. The students work one semester in groups of 5–10 as developers and each developer group is facilitated by a SE teacher. The teacher guides the students in applying a systematic software development methodology and helps them define the most difficult problems in such a way that the students can solve them. The client representatives are welcomed to give feedback on both the content and technical implementation. The student group is encouraged to follow the Scrum framework and iteratively and incrementally produce a solution to the client's problem (ScrumGuides.org 2020).

In the fall 2020 and spring 2021 semesters, an IS for managing the student internship process was developed in two consecutive Softala II projects as a part of larger internship process development activity.

Although the commissioner of the Softala II projects was our university, the project was similar to the projects with commissioner from outside, since the development project came out of the business IT department and the customer representative (product owner) had no pedagogical responsibility. Instead, there the customer representative has a strong software engineering background and experience in development projects as well as the need to get the resulting software in schedule that came from outside of the course setup. In addition to its own schedule, the project had a budged unrelated to the course, so the based on the setting, the commissioning party could have been outside the university as well. Also, due to this setting, the goal of the commissioner was to have result exceeding MVP (minimum viable product) or POC (proof of concept) which are typical goals in these kind of learning projects even if they are commissioned from outside of the university.

What made this implementation special was that we had an agreement with a third-party company to take over the project maintenance and development after a two-semester student effort (see Figure 2).

The experts involved in the project were as follows:

- Product owner, who was also an expert in Scrum. He had led a similar IT project previously.
- Technical architect from the third-party company. His role was to give students feedback on the technical implementation.
- User interface (UI) programmer from the third-party company.
- Teacher in the role of a facilitator, who also gave feedback on the project framework the students were following.

The setup for this implementation has some advantages compared with our usual company commissions. Typically, companies use the project to develop experimental technology prototypes and to see which students they might recruit after the project. The project content is seldom business critical, and based on our observations, the companies let the students explore and build the project freely without much guidance or difficult requirements.

In this implementation, the IS would really be put to use, and the third-party company would be responsible for its maintenance and further development. This meant that the product owner had a true interest in meeting the requirements of the three most important user groups for the system. In addition, the technical architect of the third-party company had an equal interest in making sure that the product would be properly built and was encouraged to have a mentoring approach toward students. This meant that both the product owner and the architect wanted to be involved with the student group at least once a week. Normally, in Softala II courses, these meetings occur around once a month.

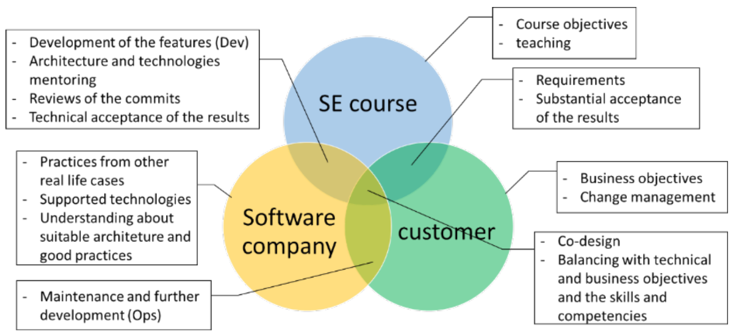
- Development of the features (Dev)
- Architecture and technologies mentoring
- Reviews of the commits
- Technical acceptance of the results
- Course objectives
- teaching
SE course
- Requirements
- Substantial acceptance of the results
- Practices from other real life cases
- Supported technologies
- Understanding about suitable architeture and good practices
- Business objectives
- Change management
Software company
customer
- Co-design
- Balancing with technical and business objectives and the skills and competencies
- Maintenance and further development (Ops)

Figure 2: Capstone course design

The product was defined in the product backlog with use cases. In every sprint planning, the use cases to be built in the next iteration were selected, and the use cases that had been built in the previous iteration were demonstrated. The student group, product owner, third-party technical architect, and teacher undertook a two-week process (sprint) to iteratively build the product.

The sprint steps are shown in Figure 3 and can be grouped into planning and implementation (phases 1–3), technical review (phases 4–6), integration (phases 7–10), and sprint review (phase 11). In planning and implementation, an iteration is planned by the student team, product owner, and technical architect. After planning, the student team defines technical tasks and does a two-week implementation iteration with weekly Q&A sessions with the product owner and technical architect.

After implementation is done, the student team starts the technical review by doing a pull request. The technical architect reviews the request and makes comments. The student team makes fixes according to the comments. After the technical review, the student team starts the integration by building automated integration tests. The technical architect merges results into the codebase, and the student team builds a docker container of the product. The technical architect then deploys a new version to the test environment. After integration, students demonstrate the results of the iteration to the product owner. After the review (phase 11), the next iteration is started (phase 1), and the steps are repeated until the end of the project.

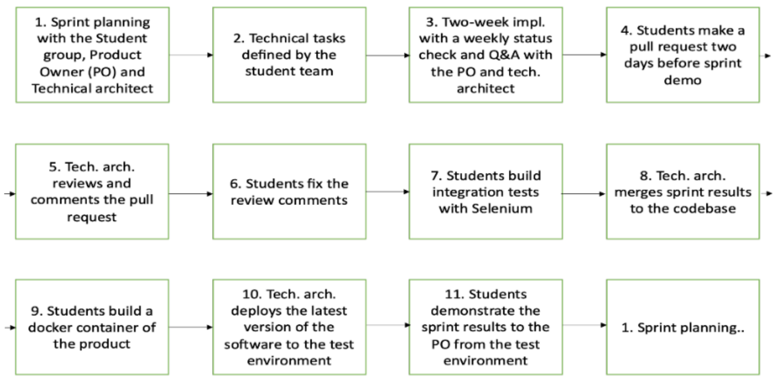
1. Sprint planning with the Student group, Product Owner (PO) and Technical architect
2. Technical tasks defined by the student team
3. Two-week impl. with a weekly status check and Q&A with the PO and tech. architect
4. Students make a pull request two days before sprint demo
5. Tech. arch. reviews and comments the pull request
6. Students fix the review comments
7. Students build integration tests with Selenium
8. Tech. arch. merges sprint results to the codebase
9. Students build a docker container of the product
10. Tech. arch. deploys the latest version of the software to the test environment
11. Students demonstrate the sprint results to the PO from the test environment
1. Sprint planning..

Figure 3: Sprint steps

The students were involved in at least the following SE activities, which are aligned with recommendations for SE education content (Bourque and Fairley 2014; Clear et al. 2019):

- Defining problems as use cases.
- Technical implementation, programming, and writing tests.
- Deployment.
- Project documentation.
- Project handover.

The project was run with the same framework with two separate student groups over two semesters. Both student groups produced results that met the goals and standards set by the product owner and the technical architect.

Data Collection Method

Since evaluation of an intervention is an essential part of ADR (Sein et al. 2011), we used several data sources to obtain a holistic picture of the situation, and the case study approach was found to be feasible here (see, e.g., (Yin 2014)). To achieve triangulation, we selected documentation (Jira comments, project chat discussions, and version control comments), interviews (semi-structured survey with open-ended questions), participant observations, and physical artifacts (the developed software) as main data sources from Yin's (Yin 2014) list of sources of evidence.

We saw the pursuit of data triangulation (Patton 1999; Yin 2014) as especially important here because the small group sizes prevented any kind of statistical analysis; moreover, because of the student–teacher setting, some interview answers could become skewed. To minimize the effect of the student–teacher setting and obtain authentic answers from students, we used an anonymized survey instead of interviews.

In a setting such as ours, participatory observations are essential because it is challenging to obtain a comprehensive inside view in other ways (Yin 2014). We were aware of the risks of being an active participant in the project and observer at the same time (Yin 2014). We minimized the risks by having three out of four of the researchers in different roles (and perspectives) in the projects, with the fourth researcher questioning the observations (investigator triangulation (Patton 1999)). All observations were discussed and in line with each other.

In evaluating the physical artifact, we used end user groups from two different universities of applied sciences (UAS). The first group (four participants) was from Haaga-Helia UAS where the software was developed, and the other one (four participants) was a reference group from another Finnish UAS.

Implementation

All the participant observations were done during the development from October 26, 2020, to May 14, 2021. Three researchers were involved in the development project, one in an SE teacher role, one as a product owner, and one as a project manager from the customer side representing the business point of view. The product owner and project manager managed requirements and communicated with stakeholders.

The survey was conducted as a Webropol questionnaire with 14 open-ended and 2 structured questions used to collect information from the Softala students. The questionnaire was open May 14–20, 2021, for all 15 students who had completed the course. Six students answered the survey (response ratio: 40%). Three students were in the course in fall 2020 and another three in spring 2021. A separate query was sent to two representatives of the third-party company (Eduix) at the end of May 2021.

The documentation was analyzed after the projects by one of the researchers (the teacher). Two of the researchers (product owner and project manager) organized the end user groups to evaluate the physical artifact (the developed software) at the end of May 2021.

Results

The findings from the four data sources are grouped into two sets. The first set consists of survey and participant observations by the product owner, the teacher, and the project manager. The idea is to have an inside view, and the set aggregates the observations of all parties involved in development. The second set has an external view and consists of documentation and artifacts to represent what is possible to do from outside the development.

Survey and Participant Observations

Students were asked what added value they were able to identify when the course was based on a real-life case. Three students mentioned increased motivation. They took the project more seriously, and it revealed real-life working methods. One student wrote that there was no room for solo performance; the students had to follow the plan strictly. Both the presence of customer representatives and the awareness that the product would be truly used by its stakeholders increased motivation. Eduix representatives (later EdxRs 1 & 2), as well as participating researchers (the teacher, product owner, and project manager), had similar opinions on the increased motivation and understanding of real-life working methods.

At the end of the questionnaire, students were also asked what kind of learning they had acquired in the course. Students mentioned several concrete details, such as precise requirement definitions from the real customer or industry (via Eduix). One student found it interesting to continue the project that some other students had already started; another found the Scrum style of working interesting.

Table 2 summarizes students' (n=6) views on what they learned and what they would have wanted to learn more about. The methods, namely overall industry method of development as well as concrete Node.js and ReactJS, were mentioned most often both as what was learned and what the students would learn more about. High number of mentions on what was learned were code reviews and project documentation. There were always less mentions for an individual item in what was learned, and there were several items that did not get any mentions in what the students would like to learn more about, namely code reviews, project documentation, systematic project handover and communicating with the stakeholders.

Based on Table 2, it could be said that while the students felt that they learned the methods and concrete technologies, they would have wanted to learn more about them. Also, it could be said that the students felt that they learned enough about the code reviews and the project documentation, since they acknowledged these areas as covered and did not wish to learn more about them.

Table 2: What students learned and what they wished to learn more about (f=frequency)

Feature/application/ method	**I learned (f)**	**Wish I learned more (f)**
Modern industry method of JavaScript development	6	5
Node.js	6	4
ReactJS	5	2
Writing modular code	5	2
Writing integration tests (Selenium)	1	1
Integration to an external system by a third party (Peppi)	6	2
Use case–based requirements engineering	3	1
Code reviews	6	0
Implementing authentication including single sign-on	5	3
Database modeling	4	1
Stabilizing software for deployment	2	1
Product containerization (Docker)	2	2
Project documentation	6	0
Systematic project handover	5	0
Communicating with stakeholders	4	0

The second question investigated was whether there were negative effects of the real-life case. Two students did not find anything negative, but one student replied that there were too many discussions, which delayed the programming, whereas another student found the amount of work substantial, and one mentioned that the project sometimes stalled in the refinement phases. EdxR 1 did not see any negative effects. EdxR 2 thought that more time was used refining details than in normal learning situations where time can be used on experiments that may not support the product but may be interesting from a learning point of view. Participating researchers did see that the discussion and refinement phases were an important part of making the project realistic and that the frustration and stress they cause are important learning outcomes.

Students considered the simultaneous development of IS and the process it supports to be a realistic situation. However, the situation was considered to produce extra work, and sometimes, there were doubts as to whether everything would end up in the production version. To assuage such doubt, one student mentioned that some work was really deleted afterwards. The EdxRs pointed out that this happens in normal situations as well, but with a more limited scope for financial reasons—a customer does not want to incur additional charges for alterations or increased business negotiations. Participating researchers noticed that the students coped with uncertainties stemming from vague business processes and changing requirements rather well; moreover, they were eager to understand the big picture.

Since students had to organize their work, there was a risk of some having to do extra work and some having a limited number of duties or less interesting tasks to perform. All students agreed that team communication was at a good level, and five students stated that the workload was distributed evenly. Only one student disagreed about the evenness of the workload. All students estimated that the pace in the course was mostly suitable. EdxR 2 did not conduct observation because of his role in the project, but EdxR 1 gave a positive estimation of the students' efficient work, and the students' responses were quick. EdxR 1 clarified that the fall term group used more time and effort to sketch the front-end planning, and the spring term group could have followed this practice.

When the whole group changed for the spring term, the new group worked in a more reactive way. The spring term group had a good leader who worked as a spokesperson in weekly meetings between the group and the facilitating teacher and Eduix. Otherwise, the students' work took place in a black box, and it was not possible to give any evaluation of the groups' internal affairs. Based on the discussions in weekly meetings, participating researchers developed the impression that communication and sharing the work within the group worked well.

The EdxRs supported the students during the project. Students found this support valuable, but they sometimes had the feeling that they were being treated like professionals and the information they received did not match their level. The general interpretation of students' answers is that they wished for more information and concrete solutions; at the same time, they admitted that they could have asked more from the EdxRs. The company received thanks for the rapid response to the questions students sent. EdxR 1 wished that he had had more time with the students because the students now remained somewhat distant. He also characterized his role as more reactive than guiding. From the participating researcher's point of view, the cooperation between students and the external company worked efficiently. In some cases, it seemed that students waited with their pressing questions until the weekly meetings instead of asking the EdxRs right away, but this problem improved over time.

All the students found the product owner's active participation useful. According to the students, the product owner clearly communicated business needs and gave valuable feedback as can be seen in the last three rows in Table 2. EdxR 2 also added to this that the product owner was the most important actor when requirements were defined. Product owner had a remarkable role to emphasize the different objectives of digitalization: the target was not only the developed software, but also improved process, better tools for users and better data for decision makers.

The students found the weekly meetings between all stakeholders a good idea, which also supported students' comprehension when the teacher elaborated on the ideas the students had received from Eduix or the product owner. EdxR 2 estimated that the teacher's role as a facilitator is vital in this kind of arrangement. EdxR 1 saw the meetings led by the teacher as important to gaining a holistic view of the project. For the future, it would be more beneficial if there could be more time and attention reserved for the planning stage.

In a real-life case, it is sometimes difficult to ensure that students learn everything that was initially planned because the case may require more weight being placed on aspects that are vital for the product and ignore some that are not needed but would be useful in general. The students responded that they were able to learn what was needed quite well, but they were not able to test some details or applications—mainly because of a lack of time.

The participating researchers emphasized that the project setup simulated a real-world software project exceptionally well and enabled the students to take significant responsibility for all the relevant aspects of professional SE. The teacher estimated the students' commitment and learning results to be better than among the typical student groups he has instructed in this course for many semesters.

Documentation and Artifacts

By analyzing the data generated throughout the project in the Jira project management system and Bitbucket version control, some conclusions about the projects can also be made. The student groups submitted 190 issues to Jira. Of them, 36 were inserted by the first group which was lower than expected while the number of issues by the second group was around the number that was expected. By looking at the activity log generated by Jira, it can be seen that EdxRs had a combined 153 activity entries (53 with status updated, 38 with status created, 37 with status commented, and the rest with other statuses). Notably, all the EdxR activity in Jira had taken place with the second group.

Apparently, the first group preferred to invite the EdxRs to a live session for clarifications. The first team created six and the second team seven pull requests in Bitbucket at the end of the sprints that were merged to the main branch by the EdxRs. In addition, the EdxRs declined eight and one pull requests, respectively, made by the teams. The declined pull requests were made in the starting phase of each team's project. Most pull requests contained comments by the EdxRs or the teacher. The EdxRs' comments in the Jira tickets and in the pull requests were related to single sign-ons, integrating to background systems, good coding conventions (e.g., using a proper logging library), exception handling, environment variables, containerization, and parametrizing the code to serve different delivery configurations. In addition, one of the EdxRs commented on the user interface.

The comments can be categorized as being important when building industry-standard production-level solutions. In addition, it can be concluded that the experts truly analyzed the project from the point of view of actually having to maintain and deploy it in the future. Thus, the feedback helped the students develop their skills related to production-level software development.

The developed artifact (new IS) was evaluated in two Finnish UASs—namely, the UAS where the development was done and one outside UAS. In both UASs, there were four end users testing the IS. As a result, the artifact was considered a good baseline version answering the basic needs of the users. Based on the results, it can be concluded that the developed solution addresses real-life needs, which also shows that the right actions have been taken in development.

Discussion and Conclusions

We implemented and studied a new kind of capstone course where we had three different parties involved—a development group from the course, a real customer, and a real software development company. We see that this kind of trilateral approach is new in capstone courses; typically, there have been only one outside party involved, which normally a client organization outside of software industry, or in some cases, the software company (see, e.g., Ahmad and Liukkunen 2019). With this kind of trilateral approach, we tried to solve the observed problems of previous capstone course implementations—namely, task-related problems and problems related to technologies (Vanhanen et al. 2018). The idea is that the software company will be responsible for maintenance of the developed software after the development project is over, and the company will have high motivation to ensure the quality and maintainability of the developed code. In contrast, the customer organization has a real process change going on, and this organization is highly motivated to prioritize requirements and answer and comment on students' questions to obtain a supporting tool for their new process. The teacher of the course acted as a facilitator, giving guidance related to the technologies used and the teamwork.

As an answer to the first research question (RQ1), the Softala project confirmed that the selected trilateral approach was a good solution. The course being integrated with a real-life case and truly aiming for a commercialized product makes a difference; even more so, the developed IS was not only a prototype or a technological experiment but also an application that will be further developed and should be robust, that is, vital for a customer's business.

As the students understood the high bar set for the real-world case, there was a fear that it could cause extra stress or anxiety. The findings revealed that the stress factor appeared, but from a positive perspective. The students had a more responsible attitude, and they were ambitious to show what they could do.

The project also showed that there are substantial benefits to the project owner having time to join the weekly meetings and communicate with the students directly. As the project owner in this case was a professional in the field, he was able to deliver a structured presentation of requirements and priorities. Aligned with the agile development method, some requirements and priorities were changed during the process. Since this was transparent to the students, it helped them understand how the entirety of development works and how the smaller tasks were related to the bigger picture.

Since it was planned that Eduix would take over the application in the future, it was natural that their representatives wanted to take an active role in the course and mentor the students who supported them in their work. However, a question remained as to whether it would have been beneficial for the Eduix representatives to have had more time with the students or whether this would have led to a situation where students shifted the duties to the representatives instead of trying to solve the problems independently, as they did in reality. The shift could have decreased the level of motivation, and the risk was considered from the beginning of the project by the guiding teacher of the course.

An important result was understanding the role of a teacher in this type of setting. As a facilitator, the teacher also operated as a mediator between the students and the product owner and Eduix. The teacher understood students' level of comprehension, and sometimes it was necessary to translate the needs and requirements to concrete actions the students should complete. There is a significant danger when an expert outside of the educational world comes to advise students that the level of abstraction in their instruction will be too high. The students' answers confirmed that this tendency was somewhat visible in the course but did not do much harm thanks to the teacher's mediator role.

We can also conclude some reasons why this kind of setting works better for students (and lecturers): it seems that making difference from other courses, having a realistic setting with different professionals involved, and having real life objectives affects positively to motivation of students (and lecturers). At best, they see the development project more like on opportunity than an assignment. However, we see it important that the why-question is further studied in future studies.

As an answer to the second research question (RQ2) it can be stated that the approach helped students to understand not only the application they were programming but also the business process they were digitalizing. This is a crucial benefit of this approach because it takes learning to the digitalization education level. Because of the strong role of outside product owner, in this kind of setting the other aspects of digitalization, such as process development, building a tool for users and creating a data collection practices for decision makers are forced to take into account as well.

More detailed results of the learning outcomes (see Table 2) can be interpreted so that students were able to learn a good variety of skills and competencies needed for their future. As expected, the results also revealed that some topics were not properly covered (the second column in Table 2)—or possibly that the students were motivated to learn even more. When students divide the work independently, there is a chance that not all the students will be able to familiarize themselves with all aspects of the task. For example, Selenium testing was done by only one student; the others did not have a chance to practice this.

From another perspective, it can be said that the autonomy in arranging the work within the group forced students to communicate efficiently and learn real teamwork. As Vanhanen et al. (2018) stated, in real-life connected capstone courses, it is unrealistic to have common learning objectives for all students. However, working in considerably small intensive groups helps all students to understand the whole development and different roles, even if students have not done all the different tasks. An additional benefit was that the students needed to communicate with stakeholders, which does not always occur with developers, who just receive orders from their managers. Presumably, this increased the students' motivation and responsibility.

Overall, one observation was the differences between the two student groups. Since the fall group started from scratch, it was obvious that the pace was a slower and the number of issues significantly lower than that of the spring term group. The spring term group also benefited from the motivated and active student leader, and the group as a whole was skillful in communicating the students' concerns to stakeholders.

The results achieved via ADR are prone to biases because researchers have two roles: developers and observers (researchers). To minimize bias, triangulation of different sources and analysis were used to cross-validate results. Reliability is fair since the documentation was precise and up to date during the whole project. The theoretical and content validity were at a good level since all participants were familiar with the concepts and they were used, e.g., in the questionnaire. However, the ecological validity is limited because of the uniqueness of the project.

In conclusion, it seems that the model used in this case gave substantial benefits compared with typical courses or even compared with other project-based implementations while being aligned as a whole with the learning framework (such as EFQ and SWEBOK) contents. We see that this kind of trilateral approach, where both a software company and a customer outside the capstone course have clear roles and responsibilities, helps to solve both task management–related problems and problems related to technologies observed in previous research (Vanhanen et al. 2018). In addition, a hybrid development model in which the waterfall and Scrum approaches were mixed seemed to be an optimal choice.

The unfinished but operational version of the resulting IS has been tested by Haaga-Helia and another UAS, and the reviewers' comments have been positive. Therefore, we are confident that the model introduced in this article would bring added value to both SE education and digitalization education wherever it can be put into practice. In particular, the trilateral approach emphasizes having several key stakeholders that need and will benefit from the results of an IS development project. These findings give a good basis for future planning of similar courses and SE education using the trilateral approach contributing to the educational setting by adding important real-life project characteristics to the learning experience.

This study opens new paths for researchers as well. It is important to have more studies how different settings affect to student motivation, and how to make students understand digitalization as a whole, and not just from a software development perspective.

References

ACM. 2021. “Curricula Recommendations.”

Ahmad, M. O., and Liukkunen, K. 2019. “Software Factory Project for Enhancement of Student Experiential Learning,” *16th International Conference on Cognition and Exploratory Learning in Digital Age, CELDA 2019* (Celda), pp. 297–306. (https://doi.org/10.33965/celda2019_201911l037).

Akınoğlu, O., and Tandoğan, R. Ö. 2007. “The Effects of Problem-Based Active Learning in Science Education on Students’ Academic Achievement, Attitude and Concept Learning,” *EURASIA Journal of Mathematics, Science and Technology Education* (3:1). (https://doi.org/10.12973/ejmste/75375).

Alred, G., and Garvey, B. 2019. *Mentoring Pocket Book*, (4th ed.).

Anderson, L., Peter, K., Airasian, D., Cruikshank, K., Mayer, R., Pintrich, P., Raths, J., and Wittrock, M. 2001. *Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives*, Addison Wesley Longman.

Ardis, M., Budgen, D., Hislop, G. W., Offutt, J., Sebern, M., and Visser, W. 2015. “SE 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering,” *A Volume of the Computing Curricula Series.*

Argyris, C. 1977. “Organizational Learning and Management Information Systems,” *Accounting, Organizations and Society* (2:2), pp. 113–123.

Avison, D. B. E., and Fitzgerald, G. 2003. “Where Now for Development Methodologies?,” *Communications of the ACM* (46:1), pp. 78–82.

Bojic, P., Greasley, A., and Hickie, S. 2019. *Business Information Systems : Technology, Development and Management for the Modern Business*, (Sixth edit.), Pearson.

Borg, M., Olsson, T., Franke, U., and Assar, S. 2018. “Digitalization of Swedish Government Agencies — A Perspective Through the Lens of a Software Development Census,” in *International Conference on Software Engineering*.

Borg, M., Wernberg, J., Olsson, T., Franke, U., and Andersson, M. 2020. “Illuminating a Blind Spot in Digitalization - Software Development in Sweden’s Private and Public Sector,” in *42nd International Conference on Software Engineering Workshops (ICSEW’20).*

Bourque, P., and Fairley, R. E. 2014. *Guide to the Software Engineering Body of Knowledge, Version 3.0*, IEEE Computer Society. (www.swebok.org).

vom Brocke, J., Baier, M. S., Schmiedel, T., Stelzl, K., Röglinger, M., and Wehking, C. 2021. “Context-Aware Business Process Management: Method Assessment and Selection,” *Business and Information Systems Engineering*. (https://doi.org/10.1007/s12599-021-00685-0).

vom Brocke, J., and Sinnl, T. 2011. “Culture in Business Process Management : A Literature Review,” *Business Process Management Journal* (17:2), pp. 357–377. (https://doi.org/10.1108/14637151111122383).

Bruegge, B., Krusche, S., and Alperowitz, L. 2015. “Software Engineering Project Courses with Industrial Clients,” *ACM Transactions on Computinig Education* (15:4), pp. 1–32. (https://doi.org/10.1145/2732155).

Burge, J. E., and Gannod, G. C. 2009. “Dimensions for Categorizing Capstone Projects,” in *22nd Conference on Software Engineering Education and Training, CSEET 2009*, pp. 166–173. (https://doi.org/10.1109/CSEET.2009.37).

Business Technology Forum. 2021. “Business Technology Standard.” (https://www.managebt.org/).

Chase, J. D., Oakes, E., and Ramsey, S. 2007. “Using Live Projects without Pain: The Development of the Small Project Support Center at Radford University,” in *38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2007,* pp. 469–473. (https://doi.org/10.1145/1227310.1227468).

Christensen, K., and Rundus, D. 2003. “The Capstone Senior Design Course: An Initiative in Partnering with Industry,” *Proceedings - Frontiers in Education Conference, FIE* (3), S2B12-S2B17. (https://doi.org/10.1109/FIE.2003.1265941).

Clear, A., Parrish, A. S., Impagliazzo, J., and Zhang, M. 2019. *Computing Curricula 2020*. (https://doi.org/10.1145/3287324.3287517).

Cunningham, W. 1992. “Experience Report- The WyCash Portfolio Management System,” *ACM SIGPLAN OOPS Messenger* (4:2), pp. 29–30.

Dagnino, A. 2014. “Increasing the Effectiveness of Teaching Software Engineering: A University and Industry Partnership,” in *27th Conference on Software Engineering Education and Training, CSEE* *and T 2014*, pp. 49–54. (https://doi.org/10.1109/CSEET.2014.6816781).

Dahlberg, T., and Lagstedt, A. 2018. "There Is Still No ' Fit for All ' IS Development Method : Business Development Context and IS Development Characteristics Need to Match," in *51st Hawaii International Conference on System Sciences* (Vol. 9).

Dahlberg, T., and Lagstedt, A. 2021. "Fit to Context Matters – Selecting and Using Information Systems Development Methods to Develop Business in Digitalization Contexts," in *54th Hawaii International Conference on System Sciences 2021*, pp. 6902–6911. (https://doi.org/10.24251/HICSS.2021.829).

Davenport, T. H. 2010. "Process Management for Knowledge Work," in *Handbook on Business Process Management 1* (2nd ed.), J. vom Brocke and M. Rosemann (eds.), Springer Berlin Heidelberg, pp. 17–35.

Davenport, T. H., and Short, J. E. 1990. "The New Industrial Engineering : Information Technology And Business Process Redesign," *Sloan Management Review* (31:4), pp. 11–27.

Donnelly, R., and Fitzmaurice, M. 2005. "Collaborative Project–Based Learning And Problem–Based Learning In Higher Education: A Consideration Of Tutor And Student Roles In Learner-Focused Strategies," *Aishe Readings* (1), pp. 87–98.

European Commission. 2020. "European Skills, Competences and Occupations (ESCO)." (https://ec.europa.eu/esco/portal/home).

European Parliament and Council of the European Union. 2008. "Recommendations of the European Parliament and of the Council of the 23 April 2008 on the Establishment of the European Qualifications Framework for Lifelong Learning (2008/C 111/01)," *Off. J. Eur. Union* (C111:1), pp. 1–7. (https://doi.org/10.2766/14352).

Fornaro, R. J., Heil, M. R., and Tharp, A. L. 2007. "Reflections on 10 Years of Sponsored Senior Design Projects: Students Win-Clients Win!," *Journal of Systems and Software* (80:8), pp. 1209–1216. (https://doi.org/10.1016/j.jss.2006.09.052).

Gross, S., Stelzl, K., Grisold, T., Mendling, J., Röglinger, M., and vom Brocke, J. 2021. "The Business Process Design Space for Exploring Process Redesign Alternatives," *Business Process Management Journal*. (https://doi.org/10.1108/BPMJ-03-2020-0116).

Hammer, M. 1990. "Reengineering Work: Don't Autmate, Obliterate," *Harvard Business Review* (July-Augus), pp. 104–112.

Hammer, M. 2010. "What Is Business Process Management?," in *Handbook on Business Process Management 1*, J. vom Brocke and M. Rosemann (eds.), Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 3–16. (https://doi.org/10.1007/978-3-642-00416-2).

Hansen, S., and Lyytinen, K. 2010. "Challenges in Contemporary Requirements Practice," in *42nd Hawaii International Conference on System Sciences*, pp. 1–11. (https://doi.org/10.1109/HICSS.2010.98).

Harmon, P. 2010. "The Scope and Evolution of Business Process Management," in *Handbook on Business Process Management 1*, J. vom Brocke and M. Rosemann (eds.), Springer Berlin Heidelberg, pp. 37–82.

Hastie, S., and Wojewoda, S. 2015. "Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch," *InfoQ, Blog*. (https://www.infoq.com/articles/standish-chaos-2015, accessed February 22, 2018).

Hsu, H.-J. 2019. "Practicing Scrum in Institute Course," in *52nd Hawaii International Conference on System Sciences* (6), pp. 7770–7778. (https://doi.org/10.24251/hicss.2019.935).

Kauppinen, R., and Lagstedt, A. 2011. "Progressive Inquiry in Agile Software Development Education," *Interdisciplinary Studies Journal* (1:3), pp. 38–46.

Kauppinen, Raine, Lagstedt, A., and Lindstedt, J. 2020. "Digitalizing Teaching Processes – How to Create Usable Data with Minimal Effort," *European Journal of Higher Education IT* (1).

Kauppinen, R., Lagstedt, A., and Lindstedt, J. P. 2020. "Expert-Oriented Digitalization of University Processes," *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 11984 LNCS). (https://doi.org/10.1007/978-3-030-38778-5_8).

Lagstedt, A., and Dahlberg, T. 2018. "A Contingency Theory Motivated Framework to Select Information System Development Methods," in *Pacific Asia Conference on Information Systems*, , June 26, pp. 1–14. (https://aisel.aisnet.org/pacis2018/46).

Lagstedt, A., Lindstedt, J. P., and Kauppinen, R. 2020. "An Outcome of Expert-Oriented Digitalization of University Processes," *Education and Information Technologies*, Education and Information Technologies. (https://doi.org/10.1007/s10639-020-10252-x).

Leidig, P., and Salmela, H. 2020. *A Competency Model for Undergraduate Programs in Information Systems*, ACM.

Maguire, J., Sheridan, N., Draper, S., and Cutts, Q. 2019. “Mentoring Mentors in Cooperative Software Engineering Education Programmes,” in *International Computing Education Research*, New York, NY, USA: ACM, July, pp. 307–307. (https://doi.org/10.1145/3291279.3341205).

Moe, N. B., Aurum, A., and Dybå, T. 2012. “Challenges of Shared Decision-Making: A Multiple Case Study of Agile Software Development,” *Information and Software Technology* (54:8), Elsevier B.V., pp. 853–865. (https://doi.org/10.1016/j.infsof.2011.11.006).

Patton, M. Q. 1999. “Enhancing the Quality and Credibility of Qualitative Analysis.,” *Health Services Research* (34:5 Pt 2), pp. 1189–208. (http://www.ncbi.nlm.nih.gov/pubmed/10591279%0Ahttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1089059).

Penzenstadler, B., Mahaux, M., and Heymans, P. 2013. “University Meets Industry: Calling in Real Stakeholders,” in *Software Engineering Education Conference,* pp. 1–10. (https://doi.org/10.1109/CSEET.2013.6595231).

Perrenet, J. C., Bouhuijs, P. A. J., and Smits, J. G. M. M. 2000. “The Suitability of Problem-Based Learning for Engineering Education: Theory and Practice,” *Teaching in Higher Education* (5:3), pp. 345–358. (https://doi.org/10.1080/713699144).

Pihir, I., Tomičić-Pupek, K., and Furjan, M. T. 2018. “Digital Transformation Insights and Trends,” in *Central European Conference on Information and Intelligent Systems*, pp. 141–150.

Richardson, I., and Delaney, Y. 2009. “Problem Based Learning in the Software Engineering Classroom,” in *22nd Conference on Software Engineering Education and Training, CSEET 2009*, pp. 174–181. (https://doi.org/10.1109/CSEET.2009.34).

Rosemann, M., and vom Brocke, J. 2010. “The Six Core Elements of Business Process Management,” in *Handbook on Business Process Management 1* (2nd ed.), J. vom Brocke and M. Rosemann (eds.), Springer Berlin Heidelberg, pp. 107–122. (https://doi.org/10.1007/978-3-642-45100-3_5).

Savery. 2006. “Overview of PBL: Definitions and Distinctions,” *Interdisciplinary Journal of Problem-Based Learning* (1:1), pp. 9–20.

ScrumGuides.org. 2020. “Scrum Guide.” (https://scrumguides.org/scrum-guide.html).

Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., and Lindgren, R. 2011. “Action Design Research,” *MIS Quarterly* (35:1), pp. 37–56.

Simon, H. A. 1997. *Administrative Behavior*, (Fourth edi.), New York, USA: The Free Press.

Tafliovich, A., Estrada, F., and Caswell, T. 2019. “Teaching Software Engineering with Free Open Source Software Development: An Experience Report,” in *52nd Hawaii International Conference on System Sciences* (6), pp. 7731–7741. (https://doi.org/10.24251/hicss.2019.931).

Taran, G., Root, D., and Rosso-Llopart, M. 2008. “Continuing Challenges in Selecting Industry Projects for Academic Credit: Points to Consider and Pitfalls to Avoid,” in *Software Engineering Education Conference,* pp. 163–170. (https://doi.org/10.1109/CSEET.2008.31).

Theocharis, G., Kuhrmann, M., Münch, J., and Diebold, P. 2015. “Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices,” in *Product-Focused Software Process Improvement* (Vol. 9459), Springer, pp. 149–166. (https://doi.org/10.1007/978-3-319-26844-6_11).

Todd, R. H., and Magleby, S. P. 2005. “Elements of a Successful Capstone Course Considering the Needs of Stakeholders,” *European Journal of Engineering Education* (30:2), pp. 203–214. (https://doi.org/10.1080/03043790500087332).

Vanhanen, J., Lehtinen, T. O. A., and Lassenius, C. 2018. “Software Engineering Problems and Their Relationship to Perceived Learning and Customer Satisfaction on a Software Capstone Project,” *Journal of Systems and Software* (137), Elsevier Inc., pp. 50–66. (https://doi.org/10.1016/j.jss.2017.11.021).

Venkatraman, N. 1994. “IT-Enabled Business Transformation: From Automation to Business Scope Redefinition,” *Sloan Management Review* (35:2), pp. 73–87.

Woodward, B., Sendall, P., and Ceccucci, W. 2009. “Integrating Soft Skill Competencies through Project-Based Learning across the Information Systems Curriculum,” *Proceedings of the Information Systems Education Conference, ISECON* (26:8).

Yin, R. K. 2014. *Case Study Research: Design and Methods*, (5th ed.), Sage Publications.