Design and Evaluation of an Agile Framework for Continuous Education in Software Engineering

Patrick Wolfschwenger
Johannes Kepler University
Linz, Austria
0000-0001-5325-0511

Barbara Sabitzer
Johannes Kepler University
Linz, Austria
0000-0002-1304-6863

Zsolt Lavicza
Johannes Kepler University
Linz, Austria
0000-0002-3701-5068

***Abstract*—This Research-To-Practice Full Paper presents an agile framework that was created and evaluated in the context of a continuous professional training and development program in the IT industry. It was designed with the values and principles of Agile Software Development in mind and provides a learning concept to become acquainted with DevOps and Cloud Computing practices. While DevOps promises to build, test and release software faster and more reliably through methods like continuous integration, continuous deployment/delivery, automated testing and infrastructure-as-code, the integration of Cloud Computing services empowers each step of the agile life cycle through ubiquitous access to computing resources that can be provisioned with minimal effort. The paper describes the experiences of creating and applying the didactic concept as well as observations of conducted work, learning progress, motivation and achievements.**

***Index Terms*—Lifelong Learning, Continuing Professional Training and Development, Agile Software Development, Cloud Computing, DevOps, Project-Based Learning**

I. Introduction

Agile Software Development (ASD) is becoming the standard for development projects. It is an iterative approach to project management and software development that is as unbureaucratic, simple and flexible as possible. According to the 15th Annual State of Agile Report [1], 94% of the surveyed organizations have worked with agile frameworks, mainly Scrum or Scrum-based methods like Scrumban and Scrum and XP. Over half of respondents (52%) say either a majority or all of their company's teams have adopted agile practices.

Agile approaches are no longer as controversial in terms of their feasibility and potential. However, implementing and maintaining agile process models continues to be a difficulty for many companies. The most significant adoption barriers include inconsistencies in internal processes, cultural clashes, organizational resistance to change, lack of knowledge and experience, absence of leadership participation, inadequate management support and sponsorship [1], [2]. Bringing the idea of agility closer to the groups and organizational units is an important prerequisite to implement and maintain agile process models [2].

DevOps combines cultural philosophies, practices, and tools to increases an organization's ability to deliver applications and services at high velocity [3]. It leverages the use of the right tools and automation strategies that streamline and simplify the software creation and deployment process. In combination with scalable IT resources from the cloud, a flexible working environment is created within which teams can collaborate in an agile manner [4]. IT resources can grow or shrink as needed, configured and scaled with ease. Cloud Computing (CC) facilitates the provisioning of development, test and production environments and makes it easier for agile development teams to combine them with other services from the cloud.

This nexus gives rise to the idea of deriving an educational group project that combines the concepts of ASD, DevOps and CC. Having a group or organizational unit that supports and promotes agile and cloud approaches is an important prerequisite to implement and maintain modern concepts of working. The ability to work effectively in a team is an important skill for any software developer working in an agile environment [5]. Through working with educational projects in software engineering, teams can tackle authentic problems, share diverse perspectives and develop their own knowledge and skills in relation to peers. It can be an effective method to motivate, encourage active learning and develop critical thinking, communication, leadership as well as decision-making skills [6].

II. Theoretical framework

A. Agile Software Development

The Manifesto for Agile Software Development [7] defines the foundation of values and principles for ASD which underpin a broad range of today's software development methodologies. The Agile Manifesto raised awareness of agile development, but also of the ideas behind agility in general. The seventeen signatories to the manifesto proclaim four foundational values and twelve supporting principles which lead the agile approach to software development:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more." [7]

With the Agile Manifesto, they want to express that software development is about people and their relationships. The people involved have different individual requirements, needs, and goals. Their collaboration should be supported by and not be subordinated to the agile processes and tools. A working software is more meaningful and important than a detailed documentation down to the last detail. The collaboration should be based on partnership, on eye level, looking together constructively forward into the future. Trying to contractually regulate everything in advance is futile, as software development is a complex and challenging process where hurdles and obstacles reveal themselves along the way. In the case of problems, looking at how to solve problems together instead of demanding predefined contracts is key for useful results. It is not wrong to make a plan, but it needs to be adjustable and the team should always question this plan.

B. DevOps

DevOps is a software engineering culture and philosophy that aims at building, testing and releasing software faster and more reliably through automation [8]. It leverages cross-functional teams to facilitate feedback cycles and collaboration between development and IT operations and uses methods, including but not limited to continuous integration, continuous delivery and deployment, automated testing and infrastructure-as-code, to improve quality, speed and security in software delivery [8].

CI/CD is the combined practice of continuous integration (CI) and continuous delivery/deployment (CD). CI/CD promises to help software development become more agile by delivering software faster and more reliably than traditional approaches. The goal of CI/CD is to create a cohesive release process driven by automation that reduces periods between deployments from months or weeks down to days or even hours [9].

The Infrastructure as Code (IaC) principle denotes that infrastructure is managed and provisioned by code, not by manual processes. Configuration files contain all infrastructure specifications, ensuring that the same environment is provisioned every time. It aims at minimizing human interaction in the build-deploy cycle, which thereby becomes faster, less error-prone and better auditable [10].

Automated testing is the application of tools to automate the process of software testing. There are different views of testing, which can be addressed based on testing levels, testing models, testing types and testing techniques [11].

C. Cloud Computing

A definition that is mostly used in professional circles is provided by the National Institute of Standards and Technology (NIST). It describes CC as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [12] NIST also set out the five essential characteristics, three service models and four deployment models nowadays widely used in literature.

It is characteristic that a consumer can unilaterally undertake the provisioning of resources running automatically without the need to interact with the service provider in person (*Self Service*). The services are available via the network, accessed through standard mechanisms and are not tied to a specific client (*Broad Network Access*). The provider's resources are pooled to serve multiple users based on a multi-tenant model (*Resource Pooling*). Resources can be rapidly scaled outward and inward in line with demand (*Rapid Elasticity*). Transparency is provided to users and suppliers by performing ongoing, traceable measurements (*Measured Service*).

The service models are distinguished by the level of abstraction they present to consumers. The lowest level is *Infrastructure as a Service (IaaS)* where users have control over the operating system, storage as well as deployed applications and can partially configure networking components, while the underlying infrastructure is managed by the service provider. *Platform as a Service (PaaS)* is the next level up. User control is limited to the deployed applications and a range of configuration options for the hosting environment, the network, servers, operating systems, and storage are managed by the service provider. In *Software as a Service (SaaS)*, almost all components of an application are run and managed by the service provider.

Deployment models determine where the infrastructure resides and who has control over it. Datacenter infrastructure providing services for the general public is referred to as *Public Cloud*. Computing services offered only to select consumers is called *Private Cloud*. A *Community Cloud* limits its services to multiple organizations with similar interests. A *Hybrid Cloud* is a combination of two or more different cloud infrastructures.

D. Project-Based Learning

Project-Based Learning (PBL) is an instructional methodology that encourages learners to solve real-life problems while fostering engagement and critical thinking. Markula and Aksela [13] analyze six key characteristics of PBL that were originally proposed by Krajcik and Shin [14]:

- *Driving question* - Projects are centered around solving a problem.
- *Learning goals* – Practical goals set by instructors and learners determine the intended purposes and desired achievements of a project, such as knowledge, skills, and capacities individuals should achieve.
- *Scientific practices* – Participants use scientific practices to carry out research, such as asking questions, exploration, interpretation, presentation and reflection.
- *Collaboration* – Any interaction that takes place in the course of the project, either between the learners, between instructors or with outside partners.
- *Using technological tools* – Technologies are used to facilitate project realization.

- *Artefact* – Artefacts are concrete creations that are a result of the project.

A software project typically has a start and end date, even if the end date is not always fixed from the start. Projects arise from identified problems and operate within the framework of limited resources. A project is based on methods, procedures and process models, usually specified in a project plan, to provide transparency, controllability and repeatability. Project risks include unclear or unrealistic project plans, inaccurate estimates of effort, and overly demanding requirements [15, pp. 53–92].

III. State of Research

PBL has become a widely used method of teaching in software engineering. There are many benefits of using PBL in Informatics courses, but also challenges and drawbacks that have been identified. A range of PBL approaches for ASD have been documented and evaluated by research. [16] describes an undergraduate capstone course on ASD using Scrum following core software engineering courses that taught the traditional approach. Anslow and Maurer [17] provide an experience report of teaching an ASD project course that involved teams developing web applications. Pérez and Rubio [18] have shown that academic performance in software engineering courses has improved compared to the period when PBL was not used, and the students themselves value the experience positively both for learning the subject and for improving their training as engineers. Teaching of theoretical concepts in software engineering with no link to their practical applications or no examples in the student's context is likely to discourage learning, thus it is fundamental that the teaching of software engineering provides solid links between theory and practice, which can be achieved through integrating group projects [6].

CC education is a young research area which deals with the application of CC in the education context [19]. It aims to provide methods for supporting instruction with cloud-based tools but also teaching about CC in computer science courses [20], [21]. Up to now, there has been little research on using and teaching CC in PBL. In a literature review, Wolfschwenger and Sabitzer [22] analyzed widespread challenges of CC adoption in the education sector. Çakiroğlu and Erdemir [23] analyzed the role of cloud-based tools in an online PBL setting, showing that CC technologies have adequate potential for supporting PBL by facilitating planning and distributing tasks, implementation, reporting, presentation and evaluation of stages of PBL. CC supports the agile process by increasing simplicity, reducing costs, minimizing the need to make far-reaching predictions, encouraging experimentation and closing capability gaps [4], [24]. The significant proportion of research type contribution in terms of ASD with CC is solution proposal with existing tools, the second research type contribution is case study, dealing with team collaboration, provision of development environments on the cloud and the benefits and impact of CC on ASD [4].

Since DevOps is a cultural, procedural and technological movement, DevOps education is different from other kind of technical training. It is not a common practice yet to provide such strongly technology-oriented and technology-dependent courses that could catch the fast development of both theory and practice of the software development and the continuously evolving platforms facilitated by cloud technologies [25]. Demchenko et al. [25] present a concept and recommendations on the design and pilot implementation of DevOps and cloud-based software engineering in computer science and software engineering curricula. Pang et al. [26] used Grounded Theory to study DevOps education from academic and industrial perspectives, proposing future studies on whether embedding different DevOps processes into programming assignments will improve learning outcomes, whether DevOps skills are transferable, whether DevOps adds value to academic publications and can initiate joint DevOps research with computer science academics in non-software engineering specializations, whether knowing DevOps improves employment opportunities and speeds up job integration and how DevOps is adopted and implemented in industry.

IV. Methodology

The framework was developed and applied in the context of an in-company professional training and development program, which is based on collaborative learning principles and promotes the integration of constructive community engagement with instruction and reflection to create active learning environments and mutually beneficial partnerships. The first researcher is part of this community and, as one of the course organizers, responsible for providing and managing the course design and delivery.

Action research was applied, with the goal of developing both the practical situation and the knowledge about the practice [27]. Action research is intended to support teachers in coping with the challenges and problems of practice and carrying through innovations in a reflective way [28]. The study is situated around a group of experienced software engineers in a monthly community-based upskilling course. The researcher has an active research role and gains experience through collecting and analyzing the data.

V. Design

The agile framework is intended to serve as an overarching guideline based on which the optimal procedure during the course can be derived. It determines the course structure, success criteria, and key deliverables to be provided by instructors and learners. We are not building on any pre-defined agile methodology, but try to cover the main concepts of agile development based on the values and principles of the Agile Manifesto [7]. The PBL design follows an agile approach, meaning that the team re-evaluates the effectiveness of the methods at the end of each cycle, and continuously adjusts the specific learning objectives and the ways to achieve those goals. The purpose is not to build a predefined product, but to create the path to get there and to assess what was learned

along the way. The project plan embraces integral concepts of agile methodology, including estimation techniques, iterative planning, incremental design, reviews and retrospective analysis. Our conception of the agile framework along with driving questions for each stage is illustrated by Figure 1.

Engaging participants and encouraging learning through modeling has priority over implementation activities. Student-centered modeling has shown to be an effective instructional strategy in science education. Building models of one's own mental representations and discussing them publicly can result in better understanding of the targeted phenomena and processes [29]. It would be exceedingly laborious, if learners had to rely solely on the effects of their full implementations to inform them what to do. Nevertheless, experimentation to grasp particular behavior of software, systems and services is of course desired and necessary. Learning outcome is based on the quality of collaborative planning, modeling, experimentation and reflection in heterogeneous teams.

Course instructors are responsible for the organization of the course and the management of the backlog of learning objectives. They define user stories based on the learning objectives, divide course participants in heterogeneous teams and distribute responsibilities. User stories are a suitable means to capture requirements in a structured form [30]. The composition of well-balanced, effective development teams is important for facilitating learning, fostering learners' motivation as well as obtaining a successful project outcome. Dzvonyar et al. [31] propose a process for composing development teams in software engineering project courses based on a set of criteria:

- *Practical Constraints* - Teams are similar in size unless a project requires more manpower due to a challenging problem or requirement. The technologies involved in the course determine which devices are needed for development and testing. Furthermore, a certain flexibility should be provided in terms of scheduling within the team for team meetings or work sessions.
- *Skill Distribution* – The best learning outcome for all course participants can likely be achieved if a setting is created in which less experienced students can learn from their more advanced peers. If a project requires experience in a particular field or technology, the requirements should be met by participants whenever possible.
- *Motivational Factors* – When there is a surplus of interested individuals, participants should be prioritized based on their motivation for the project course as a whole. Also, participants should have the possibility to prioritize the projects offered so that they work in projects they enjoy.
- *Personal Criteria* – To ensure cultural fit within a team, compatibility of work habits and way of thinking should be considered. Fostering collaboration across genders can break down stereotypes. The group should also be balanced in terms of language skills.

The size of software project teams is an important aspect of project outcome. Rodríguez et al. [5] found that projects with an average team size of 8 or fewer people are less productive than groups above this threshold, enhancement projects have a better productivity than new projects and the type of programming language has an impact on productivity. Therefore, we take these criteria into consideration so that groups are not too large and well balanced. Teams may be reconstituted after each cycle, and it is desirable that teams develop different models in parallel to encourage the exchange of ideas and sharing of knowledge in the collective reviews.

By mapping the complexity of the Software Development Life Cycle (SDLC), which occurs in both classic and agile projects and should therefore be familiar to every software developer, to an educational group project, we strive for the internalization of software engineering skills among participants. By originating from the well-known SDLC, instructors can build on participants' prior knowledge and make progress toward an agile methodology. The SDLC consists of standardized phases that a software development team goes through when developing a new application. The number and characteristics of the respective phases can vary depending on the software development methodology and framework. The phases occur in each approach, but may have different names or be condensed depending on the framework. The PBL design is aligned with the goals of enhancing web development skills in combination with CC and raising awareness for ASD and DevOps practices. The specific procedure within a project depends, to a large extent, on the complexity of its tasks as well as a number of internal and external factors, such as the type of technology used, the number of decision-makers involved and the size of the available budget.

A. Planning phase

At the beginning, measures need to be taken regarding the framework conditions and the learning environment. Learning goals are determined, user stories are specified, anticipated scope and resources are clarified, and feasibility is assessed. Instructors work together with experienced participants to make decisions about the scope of the upcoming lessons. They determine the requirements and reconcile them with the learning objectives derived from the learners' needs. These should be individuals who know the participants, their strengths and weaknesses and have the ability to evaluate the importance of a learned skill as well as what operational purposes its features might serve. Ideally, software is designed that arises from actual business requirements.

In almost every agile methodology, the use of estimation methods is an integral part in planning [32]. Each team should come up with an estimate of the complexity of the user stories relevant for the upcoming iteration. The estimates of the individual teams should be compared and discussed in a joint meeting. In ASD, relative estimation (A is more complex than B) is preferred over absolute estimation (e.g. two person-days of effort), because the effort is strongly dependent on who estimates and who does the work in the end, and also at what time and on what level of knowledge the estimate is being made [33, p. 161]. Kusay-Merkle [33, p. 165] describes a range

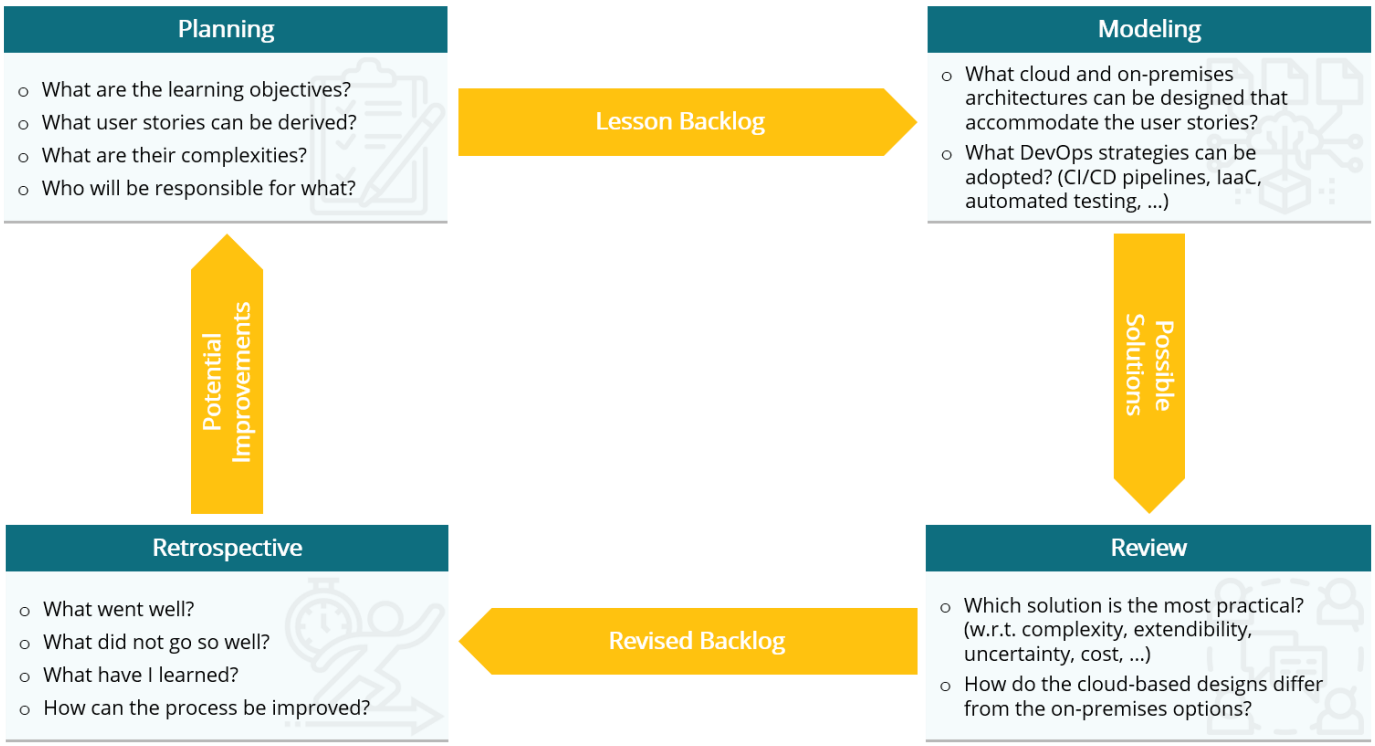

Fig. 1. The agile life cycle with driving questions for each stage.

of methods that can be used. When the estimates are far apart, it signals that the problem has been understood differently. A developer with a low estimate may know of a library or tool that can speed the implementation up; a developer with a high estimate may know of a pitfall that others have forgotten. The estimates help to talk about the plan and also act as a barometer for the scope of the increment.

B. Modeling phase

Software design is the process that defines the architecture, components, interfaces and other properties such as data structures and algorithms. A tech stack is selected including programming languages, frameworks and libraries in the front and back end and decisions are made regarding database and third-party services. By choosing a suitable software design, development work can be simplified so that, for instance, frontend and backend can be programmed separately. Agile, iteration-based development supports using the microservice approach, in which complex software is generated from independent processes that communicate with each other through language-independent programming interfaces [34].

Decisions need to be made on the infrastructure level, such as whether the software will be deployed on-premises or in the cloud and whether it will be a public, private or hybrid cloud approach, and which service models to consider. It is not always necessary to build everything from scratch and manage every component down to the smallest detail; in many cases, a suitable cloud service based on PaaS or SaaS is readily available and can be used to streamline development. To be able to make comparisons between cloud and on-premises architecture models, groups should complement each other with their approaches.

The teams should also consider how to use DevOps practices effectively. The design needs to include strategies how the underlying infrastructure can be deployed via code and how the application can be steadily integrated and checked for interoperability, potential errors and bugs with the help of CI/CD pipelines. Identifying appropriate testing strategies is an important aspect.

C. Review phase

The teams come together to inspect, compare and discuss their individual architecture models. The results of the past iteration are presented. Feedback from peers is solicited. The meeting does not require a great deal of preparation, since no formal presentation needs to be made. Only completed models may be presented, which should increase the motivation to finish the assignment.

The meeting should highlight the variations and analogies between the on-premises and cloud architecture models and build and expand on the knowledge of each individual. The complexity of the individual architecture models as well as the uncertainties, efforts and costs involved with the possible implementation should be assessed and estimated.

D. Retrospective phase

The retrospective meeting allows the team to discuss what went well or needs to be improved regarding people, relationships, processes and tools and to make commitments to address the most pressing issues. All participants must truly believe that everybody involved is equally committed to success. The process will probably fail if participants feel insecure or do not trust others on the team. Matthies et al. [35] review common retrospective activities and potential problems associated with them.

VI. Evaluation

We report our last year's experiences of applying the didactic concept as well as our observations of conducted work, learning progress and achievements. Additionally, reflections of the learners that were collected in a research diary are recapped and analyzed with regard to intrinsic motivation, perceived competence, learning success, social engagement and other factors.

The agile framework was applied in an in-company upskilling course in the IT service industry, which comprised monthly two-hour gatherings carried out mainly online due to Covid-19 restrictions in Austria [36]. Around ten people were regularly involved. The group was heterogeneously distributed in terms of development experience, ranging from six months to multiple years of professional experience. Each team member had at least a solid foundation of programming basics as taught at technical high schools, colleges and universities. We did not set an end time for the project because of its perpetual nature. We decided on continuation in the retrospective meetings under consideration of motivational factors, available learning opportunities and technical feasibility. After twelve months, the agile structure was well established, topics for ongoing training did not run out and the program has therefore been continued.

Continual alignment of the topics with the needs of the learners was an important part in the execution of the project. A list of target areas was created and kept up-to-date based on learning objectives derived from regular surveys of learner's skills and needs for development. Learning objectives were discussed and user stories were drafted with experienced colleagues in the planning phases. Also, online questionnaires pertaining to key topics were sent out to get more quantitative responses. This allowed us to gain insight into the extent to which skills were developed and which topics were perceived as important for the job and thus necessary for the learners. Becoming acquainted with a variety of cloud services, improving web development skills and familiarizing oneself with the agile way of working were the main objectives of the course.

Based on the learning objectives, project ideas were collected that should fulfill a practical purpose. But also completely detached examples were covered, which served purely educational purposes. In this context, it must be mentioned that it was sometimes difficult to find ideas that fulfilled the learning objectives on the one hand and provided a solution for current issues on the other hand. Aligning both sides requires a deep knowledge of the operational requirements as well as the required skills of the learners and was therefore driven by experienced individuals in the first place.

Multiple short cycles took place, which continuously brought about small but functional progress. In the proposed way of working, it is critical to come to presentable results in a timely manner, without getting lost in the details, in order to be able to provide an opportunity for change, to discuss further courses of action regularly and to reshape the requirements, even fundamentally, if they cannot be fulfilled or the output does not meet the expectations. The microservice perspective has proven to be supportive when working in agile frameworks and short cycles. No large monoliths should emerge, but small components that are functional, deployable, have an independent interface and single responsibility.

Placing emphasis on the use of the microservice approach constitutes a good basis for structuring applications as a collection of loosely connected services, promoting separation of concern and thus collaboration. JAMstack, which is an acronym for JavaScript, APIs, and Markup-HTML, was valued for web application design. It is a modern architectural pattern for building websites and applications [37]. The rapidly adopted architectural style in web development goes in line with the microservice approach. Compared to a traditional (monolithic) web server model, a JAMstack architecture does not build on a central server running the entire system, but rather relies on decoupled and independent systems that are integrated through APIs [37], [38]. The frontend and the backend are considered separately and can be served and developed independently of each other. It builds on the static site generator model, which offers a simple method of managing content by serving a full static website [39]. If a modular structure is chosen, software components will be able to work relatively independently. Separate development of services that are to a large extent decoupled and complete only small tasks allows better compatibility with other systems and testing.

Member dropout can be disruptive because each individual has unique skills and stake in the success of the project. It can be frustrating to pick up where someone else left off. Consequently, the possibility of a team member leaving needs to be expected. Short iteration cycles with broad participation and comprehensive reviews can compensate for those limitations.

It is a common mistake in project management to underestimate the duration time of tasks. To avoid this, the project scope should be kept as flexible as possible. The focus should be placed on meeting the minimum requirements as early as possible. Especially in vocational training courses, it is likely that participants are not always available.

Different estimation techniques were used to estimate complexity of user stories in agile projects. Planning Poker was preferred over T-Shirt Sizes and Magic Estimation methods, because it yielded good results and was an amusing way to estimate requirements as a group, although the structure of the technique itself is not entirely intuitive. The result is not the estimate of a single expert, but that of a team that has jointly arrived at this result. This increases the quality of

the estimate and the acceptance in the team. Furthermore, estimation sessions yielded lively discussions, and learners progressed per iteration. User story-based estimation supports the bottom-up approach, where requirements are broken down into smaller manageable parts depending on the necessary accuracy.

Cloud platforms have been used not only for hosting and building the applications, but also for creating CI/CD pipelines, source code management and discussion of the possible architectures. Licenses and subscriptions with a public cloud service provider were available and provided by the organization so that experimentation with DevOps practices such as IaC and automated tests could be carried out easily. The initial development effort for CI/CD is usually higher, because the pipelines for the respective software project must first be developed. In the long term, however, the process is usually more sustainable and resilient against errors.

A modern development platform providing DevOps support was the building block for coordination, sharing, and collaboration across the teams and the management of the backlog of learning goals and user stories. Most of the available cloud platforms, either public or private, support DevOps systematically on the platform, including tools for automation [40]. This creates a central basis for source code management and experimentation with CI/CD pipelines, IaC deployments and automated testing.

PBL has been used as an active learning method. Overall, the learners participated actively, constructively and contributed to a productive meeting. The general tenor in the course was that gathering, understanding, analyzing and specifying requirements was supported by the structured form of user stories. Modeling the architectures, estimating user stories and reviewing the outcomes encouraged interaction and lively discussions. It was motivating to work on an application that actually served a real-life purpose. Sometimes, experience reports from other projects were included. We used the Developer's Journey framework, which is a storytelling concept based on narrative learning principles for software engineering [41].

It was strongly agreed that programming and coding, problem-solving, logical thinking, communication, teamwork, critical thinking as well as project management can be considered as relevant software engineering skills. The course was found useful to develop the skills required for improving at the workplace. In addition, it was well received overall to try things out and experiment together on new topics.

VII. Discussion

Agile practices with DevOps and CC provide numerous benefits for software delivery including automation of the software release process, improvement of productivity and quality by freeing developers from manual tasks and possibility of additional types of code tests. Cloud platforms are common destinations of delivery automation, because their different service and deployment models provide abstraction as well as a multitude of components-off-the-shelf to achieve fast and easy application building, hosting and deployment [42].

The more complex and monolithic programs get, the more interwoven and difficult they become to troubleshoot, maintain, extend and reuse. Microservices serve as a remedy to these shortcomings, insofar as this approach aims at simplifying the architecture and make the systems more modular and maintainable. Analyzing the outputs of the iterations in terms of the IDEALS properties of microservices could give an estimate of whether the sprint has produced functional results that can be tested and assessed by users [10]:

- *Interface Segregation* - There is often a multitude of client programs (frontends) to the same service logic, which is the main motivation to apply interface segregation to microservices. Each type of frontend should see the service contract that best suits its needs instead of imposing the same service contract on all types of service clients.
- *Deployability* – Microservice developers should ensure that the software and its new versions are readily available to its users by configuring the runtime infrastructure, scaling microservices in and out based on the needs, conducting the CI/CD process, minimizing downtime and monitoring the health of the services.
- *Event-Driven* – Backend services are typically activated using HTTP call, RPC-like call using a platform-specific component technology or an asynchronous message that goes through a queue in a message broker.
- *Availability over Consistency* – Accepting the risk of occasional inconsistencies should have priority over a system that goes out of service because of trying to enforce strong consistency for a nice to have feature.
- *Loose-Coupling* – If the service contract is tightly coupled to the service logic or technology, then it is more prone to change when the logic or technology needs to evolve.
- *Single Responsibility* – The notion of single responsibility refers to the cohesiveness of services within a microservice. The microservice architecture style dictates that the deployment unit should contain only one service or just a few cohesive services.

Web engineering is concerned with the methodologies, techniques and tools that are the foundation of web application development and support the design, development and evolution process [43]. Web application development has certain characteristics that make it a specialization of traditional software engineering. It incorporates new approaches to meet the unique requirements of web-based applications. Particularly important cornerstones in web applications are interfaces, not only to the users, but also to other applications that communicate by means of standardized web protocols. The use of JAMstack that arose in the group project enabled an independent development of software components to a great extent, which can therefore be seen as a well-suited software design framework for a group-based learning approach.

It is difficult to accommodate all learning needs in a professional training and development course. With this method,

we think that we have covered different levels of learning requirements. More experienced developers are responsible for the software architecture, less experienced team members are immersed into an agile development, code review and design process with experts.

VIII. Conclusion

A PBL approach based on the values and principles of ASD and DevOps has been presented. Its purpose is to structure a continuous education program for experienced software engineers in web development with cloud focus. Over time, it has become apparent that the agile approach can also be useful for other disciplines of software engineering and can be viewed as a general framework for continuous, lifelong learning in a cooperative learning environment.

In software engineering, it is generally important to stay up to date, be well versed and specialized with a technology stack and comprehend different frameworks and templates. Software engineers are responsible for designing, building, testing, maintaining and operating software systems [44]. Thus, it was vital to address the corresponding requirements for skills in a learning design.

The more companies move away from the sequential waterfall model towards ASD, the more important it becomes to train employees and prospects in this direction. The use of ASD, DevOps and CC practices in the school and academic sector and their integration in curricula has been demanded by many, and there have already been some practical approaches. An evaluation of the presented agile framework for software engineering education in other contexts would provide opportunity for a comparative study.

References

[1] "15th state of agile report," 2021. [Online]. Available: https://info.digital.ai/rs/981-LQX-968/images/RE-SA-15th-Annual-State-Of-Agile-Report.pdf

[2] A. Komus, "Status quo (scaled) agile: 4th study on benefits and success factors of agile methods," 2020. [Online]. Available: https://www.hs-koblenz.de/en/bpm-labor/status-quo-scaled-agile-2020

[3] E. Dornenburg, "The path to devops," *IEEE Software*, vol. 35, no. 5, pp. 71–75, 2018.

[4] M. Younas, D. N. Jawawi, I. Ghani, T. Fries, and R. Kazmi, "Agile development in the cloud computing environment: A systematic review," *Information and Software Technology*, vol. 103, pp. 142–158, 2018.

[5] D. Rodríguez, M. A. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012.

[6] M. L. Fioravanti, B. Sena, L. N. Paschoal, L. R. Silva, A. P. Allian, E. Y. Nakagawa, S. R. Souza, S. Isotani, and E. F. Barbosa, "Integrating project based learning and project management for software engineering teaching," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. ACM Conferences, T. Barnes, Ed. New York, NY: ACM, 2018, pp. 806–811.

[7] K. Beck, J. Grenning, R. C. Martin, M. Beedle, J. Highsmith, S. Mellor, A. van Bennekum, A. Hunt, K. Schwaber, A. Cockburn, R. Jeffries, J. Sutherland, W. Cunningham, J. Kern, D. Thomas, M. Fowler, and B. Marick, "Manifesto for agile software development," 2001. [Online]. Available: http://agilemanifesto.org/

[8] R. W. Macarthy and J. M. Bass, "An empirical taxonomy of devops in practice," in *46th Euromicro Conference on Software Engineering and Advanced Applications*, A. Skavhaug, Ed. Piscataway, NJ: IEEE, 2020, pp. 221–228.

[9] S. Arachchi and I. Perera, "Continuous integration and continuous delivery pipeline automation for agile software project management," in *MERCon 2018*. Piscataway, NJ: IEEE, 2018, pp. 156–161.

[10] P. Merson, "Principles for microservice design: Think ideals, rather than solid," *The InfoQ eMag*, no. 91, pp. 6–13, 2021.

[11] A. J. Abdulwareth and A. A. Al-Shargabi, "Toward a multi-criteria framework for selecting software testing tools," *IEEE Access*, vol. 9, pp. 158 872–158 891, 2021.

[12] P. Mell and T. Grance, "The nist definition of cloud computing," 2011. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-145/final

[13] A. Markula and M. Aksela, "The key characteristics of project-based learning: how teachers implement projects in k-12 science education," *Disciplinary and Interdisciplinary Science Education Research*, vol. 4, no. 1, 2022.

[14] J. S. Krajcik and N. Shin, "Project-based learning," in *The Cambridge handbook of the learning sciences*, ser. Cambridge handbooks in psychology, R. K. Sawyer, Ed. Cambridge: Cambridge University Press, 2014, pp. 275–297.

[15] T. Grechenig, M. Bernhart, R. Breiteneder, and K. Kappel, *Softwaretechnik: Mit Fallbeispielen aus realen Entwicklungsprojekten*. München: Pearson Studium, 2010.

[16] V. Mahnic, "A capstone course on agile software development using scrum," *IEEE Transactions on Education*, vol. 55, no. 1, pp. 99–106, 2012.

[17] C. Anslow and F. Maurer, "An experience report at teaching a group based agile software development project course," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. ACM Digital Library, A. Decker, Ed. New York, NY: ACM, 2015, pp. 500–505.

[18] B. Pérez and Á. L. Rubio, "A project-based learning approach for enhancing learning skills and motivation in software engineering," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. ACM Digital Library, J. Zhang, Ed. New York,NY,United States: Association for Computing Machinery, 2020, pp. 309–315.

[19] H. P. Breivold and I. Crnkovic, "Cloud computing education strategies," in *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*, A. Bollin, Ed. Piscataway, NJ: IEEE, 2014, pp. 29–38.

[20] J. A. González-Martínez, M. L. Bote-Lorenzo, E. Gómez-Sánchez, and R. Cano-Parra, "Cloud computing and education: A state-of-the-art survey," *Computers & Education*, vol. 80, pp. 132–151, 2015.

[21] P. Wolfschwenger, B. Albaner, O. Kastner-Hauler, and B. Sabitzer, "The value of cloud-based learning environments for digital education," in *2020 IEEE Frontiers in Education Conference (FIE)*. Piscataway, NJ: IEEE, 2020, pp. 1–5.

[22] P. Wolfschwenger and B. Sabitzer, "Effective integration of cloud services into educational organizations," in *Proceedings of EdMedia + Innovate Learning 2020*. Waynesville, NC: Association for the Advancement of Computing in Education (AACE), June 2020, pp. 132–137.

[23] Ü. Çakiroğlu and T. Erdemir, "Online project based learning via cloud computing: exploring roles of instructor and students," *Interactive Learning Environments*, vol. 27, no. 4, pp. 547–566, 2019.

[24] D. Bermbach, A. Chandra, C. Krintz, A. Gokhale, A. Slominski, L. Thamsen, E. Cavalcante, T. Guo, I. Brandic, and R. Wolski, "On the future of cloud engineering," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2021, pp. 264–275.

[25] Y. Demchenko, Z. Zhao, J. Surbiryala, S. Koulouzis, Z. Shi, X. Liao, and J. Gordiyenko, "Teaching devops and cloud based software engineering in university curricula," in *IEEE 15th International Conference on eScience*. Piscataway, NJ: IEEE, 2019, pp. 548–552.

[26] C. Pang, A. Hindle, and D. Barbosa, "Understanding devops education with grounded theory," in *2020 ACM/IEEE 42nd International Conference on Software Engineering: Software engineering education and training*, G. Rothermel and D.-H. Bae, Eds. Piscataway, NJ: IEEE, 2020, pp. 107–118.

[27] B. M. Capobianco and A. Feldman, "Repositioning teacher action research in science teacher education," *Journal of Science Teacher Education*, vol. 21, no. 8, pp. 909–915, 2010.

[28] A. Feldman, H. Altrichter, P. Posch, and B. Somekh, *Teachers investigate their work: An introduction to action research across the professions*,

third edition ed. Abingdon, Oxon and New York, N.Y.: Routledge, 2018.

[29] P. S. Oh and S. J. Oh, "What teachers of science need to know about models: An overview," *International Journal of Science Education*, vol. 33, no. 8, pp. 1109–1130, 2011.

[30] M. R. M. Chopade and N. S. Dhavase, "Agile software development: Positive and negative user stories," in *2017 2nd International Conference for Convergence in Technology (I2CT)*, G. K. Kharate, Ed. Piscataway, NJ: IEEE, 2017, pp. 297–299.

[31] D. Dzvonyar, L. Alperowitz, D. Henze, and B. Bruegge, "Team composition in software engineering project courses," in *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials*, ser. ACM Conferences, P&eacute, C. raire, and cile, Eds. New York, NY: ACM, 2018, pp. 16–23.

[32] N. e. Farih, K. Nafil, and R. El Messousi, "Effort estimation in agile software development: A systematic mapping study," in *New Trends in Intelligent Software Methodologies, Tools and Techniques*, ser. Frontiers in Artificial Intelligence and Applications, H. Fujita and H. Perez-Meana, Eds. IOS Press, 2021.

[33] U. Kusay-Merkle, *Agiles Projektmanagement im Berufsalltag: Für mittlere und kleine Projekte*, 2nd ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021.

[34] S. Li, H. Zhang, Z. Jia, C. Zhong, C. Zhang, Z. Shan, J. Shen, and M. A. Babar, "Understanding and addressing quality attributes of microservices architecture: A systematic literature review," *Information and Software Technology*, vol. 131, p. 106449, 2021.

[35] C. Matthies, F. Dobrigkeit, and A. Ernst, "Counteracting agile retrospective problems with retrospective activities," in *Systems, Software and Services Process Improvement*, ser. Communications in Computer and Information Science Ser, A. Walker, Ed. Cham: Springer International Publishing AG, 2019, vol. 1060, pp. 532–545.

[36] P. Wolfschwenger, S. Hinterplattner, H. Demarle-Meusel, B. Albaner, and B. Sabitzer, "Learning under lockdown: The conditions in austria in a global context," in *Proceedings of the 13th International Conference on Computer Supported Education*, B. Csapó and J. Uhomoibhi, Eds. Setúbal, Portugal: SCITEPRESS - Science and Technology Publications, 2021, pp. 648–656.

[37] Bilmann Mathias & Contributors, "Jamstack: For fast and secure sites," 2016. [Online]. Available: https://jamstack.org/

[38] F. Zammetti, Ed., *Practical JAMstack: Blazing Fast, Simple, and Secure Web Development, the Modern Way*. Berkeley, CA: Apress L. P, 2020.

[39] A. Visconti, P. Morgan, and J. Howe, "Building a static website with jekyll and github pages," *Programming Historian*, no. 5, 2016.

[40] P. Agrawal and N. Rawat, "Devops, a new approach to cloud development & testing," in *IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT-2019)*. Piscataway, NJ: IEEE, 2019, pp. 1–4.

[41] P. Wolfschwenger, M. Emara, W. Lumetsberger, T. Hatter, B. Sabitzer, and Z. Lavicza, "The developer's journey: A storytelling framework for cooperative learning in software engineering," in *Proceedings of the 14th International Conference on Computer Supported Education*, M. Cukurova, N. Rummel, D. Gillet, B. McLaren, and J. Uhomoibhi, Eds. Setúbal, Portugal: SCITEPRESS - Science and Technology Publications, 2022, pp. 525–533.

[42] Y. Rouf, J. Mukherjee, M. Litoiu, J. Wigglesworth, and R. Mateescu, "A framework for developing devops operation automation in clouds using components-off-the-shelf," in *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, ser. ACM Digital Library, J. Bourcier, Ed. New York,NY,United States: Association for Computing Machinery, 2021, pp. 265–276.

[43] M. Nußbaumer, *Entwicklung und Evolution dienstorientierter Anwendungen im Web Engineering*. Erscheinungsort nicht ermittelbar: KIT Scientific Publishing, 2008.

[44] S. Baltes and S. Diehl, "Towards a theory of software development expertise," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ACM Conferences, G. T. Leavens, Ed. New York, NY: ACM, 2018, pp. 187–200.