2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)

Industry-Academy Collaboration in Teaching DevOps and Continuous Delivery to Software Engineering Students: Towards Improved Industrial Relevance in Higher Education

1st Kati Kuusinen
University of Southern Denmark (former)
Technical University of Denmark (current)
Copenhagen, Denmark
kakuu@dtu.dk

2nd Sofus Albertsen
Praqma inc.
Copenhagen, Denmark
soa@praqma.net

***Abstract*—Global industrial demand for highly skilled professional software engineers is increasing. Many countries already experience shortage of developer workforce and it is predicted that the industrial need for software engineers will grow on a higher rate than educational institutes are able to train new workforce. The main reasons for this deficit are in the education system's inability to adapt to current market needs and in difficulties in matching available skills with existing jobs. Therefore, increasing the industrial and market relevance of the education can be a key solution. Another significant contributor is teaching more efficient working methods such as automating repetitive parts of developer work to help to concentrate on tasks that directly create customer and business value. This paper presents the design and execution of a Continuous Delivery and DevOps course organized in company-university collaboration. The objective is to investigate how university courses requiring multidisciplinary lecturer skills and complex execution architectures can be organized in industry-academia collaboration to improve the industrial relevance of higher education.**

***Index Terms*—Engineering education, education courses, agile software development**

I. Introduction

A core goal of higher education in software engineering is to educate professional workforce for the variety of industrial software engineering jobs. Therefore, the requirement for industrial relevance in software engineering education is high as the majority of new graduates are employed in software intensive industries. Moreover, software engineering is a professional human activity that demands soft, technical, and business skills and competences. The requirement of soft skills is increasing in the software industry with phenomena such as DevOps [1] that focuses both on automatizing the software production pipeline with technical tools but also on the collaboration between different specialists and stakeholders.

Teaching DevOps in higher education is challenging for multiple reasons. First, it is multidisciplinary and therefore crosscutting the traditional course boundaries by requiring t-shaped skills from the educators [2]. Secondly, it necessitates complex execution architectures and virtualization environments which are laborious to build up and maintain in a university environment [2]. Thirdly, as industrially relevant technologies are still emerging and developing around DevOps and Continuous Delivery, obtaining and maintaining the needed knowledge and skills is time-consuming for a university lecturer who is not using these tools in their daily work. However, as DevOps and Continuous Delivery are increasingly used methodologies in the software industry, they must be addressed in higher education for industrial relevance.

To overcome the aforementioned challenges and to improve the industrial relevance of software engineering education, this paper describes the design and execution of a DevOps and Continuous Delivery course organized in industry-university collaboration in Denmark. The course was organized by the Software Engineering section at the University of Southern Denmark and Praqma which is a Scandinavian company specialized in training and consulting in Continuous Delivery and DevOps. This paper investigates the question of how the collaboration between industry and academia can be utilized to improve the teaching of DevOps and Continuous Delivery. The goal is to increase the industrial and market relevance of the knowledge, skills and competences of software engineering students by teaching them modern software development technologies and tools to an extent that rarely is possible in a university due to the aforementioned challenges. Indirect objectives of this work are to investigate a form of academy-industry collaboration to develop the relevance of higher education in software engineering and to develop approaches to organize such courses also without company collaboration.

II. Background

A. Teaching Continuous Delivery and DevOps

Continuous Delivery automates operations from code commit to deployment and DevOps expands it with the organi-

*The University of Southern Denmark (https://www.sdu.dk/en/) and Praqma (https://www.praqma.com/) funded the course.

978-1-7281-1000-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSE-SEET.2019.00011

zational viewpoint of development and operations working together in a single team [2]. The challenge for educators is that they need to be able to teach both development and operations activities such as monitoring, networking, operations, operating systems [2], [17]. Additional management and administration tasks for maintaining the virtual environment both increase costs and educators' workload [17].

Teaching cultural enablers of DevOps or *DevOps mindset* is another challenge. Cultural enablers include shared values, goals and ways of working, definition of success, collective ownership, respect and trust, constant communication, and continuous experimentation and learning [3]. Clear [4] discusses the mindset and attitude required in DevOps roles and the dispositions employers expect from their DevOps staff. He reflects on whether and how we can teach our students, for instance, to be *passionate, motivated team players*.

Hussain et al. [5] studied what knowledge, skills and capabilities employers mention in their job advertisements for DevOps roles in New Zealand. They observed that the most often mentioned knowledge include cloud and network infrastructure, continuous integration, delivery and deployment, and operation systems. Over 90% of the advertisements mentioned experience in using delivery and deployment tools and configuration management tools. Sought after capabilities included attributes or soft skills such as communication skills, leadership, customer engagement, mentorship, and adaptability and learnability. Dispositions included being passionate, team player, motivated, enthusiastic, curious and innovative.

In summary, the research is scarce on how to teach DevOps and Continuous Delivery in higher education in an efficient, multidisciplinary and industrially relevant way.

B. Educational Collaboration and Multidisciplinary Teaching

Kunttu [6] defines educational collaboration as interactions between academic institutions and non-academic organizations involving academic educational activities. While academy industry collaboration is often research-focused [6], this paper concentrates on collaboration with focus on educational aspects and the improvement of the relevance of software engineering education in higher education institutes, and especially on jointly organized learning activities. Kunttu [6] divides educational collaboration under the following: 1. student projects for groups of undergraduate students, 2. thesis projects, 3. tailored degree courses, and 4. jointly organized courses. This paper investigates the fourth format of collaboration.

Kunttu [6] interviewed educational collaborators both from companies and universities. Company benefits of jointly organized courses included finding workforce among the participating students, being able to train own staff at the same time, and getting training on an industry-specific topic as needed. A recent systematic literature review [7] on preparing ICT graduates for real-world challenges summarizes that the higher education of future ICT professionals requires a more holistic and strategic approach. The EU has already recognized a growing deficient of educated and skilled ICT workers and it is predicted to increase into hundreds of thousands of unfilled ICT jobs [7]. Better bridges between education and work are thus needed. The OECD has proposed employer engagement in skills development in various forms such as workplace learning, apprenticeships, proprietary study programs, and cooperation with universities [7]. ICT workers in the EU are highly educated; 56.5% of them have a tertiary level education degree [8] and the demand for highly skilled professionals is rapidly growing.

A systematic review [7] found that IT programs should seek to tailor their curricula to better meet the entry-level needs of IT companies. They state that the focus should be on learning outcomes that support a higher level of competency in the areas of programming languages, systems development life cycle methodologies, and knowledge of the IT industry. Most of the included papers addressing academy-industry collaboration suggested involving industry representatives in the process of curriculum design and delivery, which was also seen as one of the most important implications for the area of collaboration between academia and industry. One often mentioned practice was constant dialog between academy and working life for the academy to stay aware of the skill-set requirements of industries and to enable innovative teaching strategies such as learning by doing, teamwork, collaborative learning, experiential learning, studio based learning, job-oriented experiment course system, problem- or project based approach, and work integrated learning that reflect industry practices and real-life examples [7] ref. ([8], [9], [10], [11], [12]).

Educational trends in organizing teaching and learning are towards multi-disciplinary teaching and multi-professional collaboration. A goal of this paper is to seek for means to improve alignment of both theoretical and practical interdisciplinary learning objectives and outcomes of software engineering with the help of automated software production pipeline that connects previously siloed tasks of software engineering into a single production line. This can be supported by multi-disciplinary learning goals combining software business, user experience, software development and operations.

III. Syllabus

The first author participated a software seminar[1] on the relation of Software Architecture and DevOps / Continuous Delivery where she discussed with other academics and industry partners what an introductory course on DevOps should cover. The authors then selected teaching topics based on these discussions and industrial relevance as seen by the consultant company partner. The first author developed learning objectives with the following question in mind: *What learning objectives promote both learning of underpinning theories and abstraction frameworks as well as learning of practical skills that improve the employability of new software engineering graduates?*

The first author defined learning objectives using the typology of knowledge, skills and competences presented by

[1] Vienna Software Seminar https://vss.swa.univie.ac.at/2017/

Winterton et al. [14] and structured the learning objectives using the Structure of Observed Learning Outcomes (SOLO) taxonomy [15]. She applied the principles of constructive alignment in designing the course and used a template from York University [16] to align learning outcomes, contents, activities and assessments. The course was an elective targeted to software engineering students starting their third (final) year of bachelor studies or studying in master programs. An introductory course to software engineering and basic programming skills were defined as prerequisites. The course was offered in the format of two-week summer school and the scope was 5 ECTS (European Credit Transfer System) equivalent to approximately 125 hours of student work. Program-wise, the goal of the course was to deepen students' competences in industrially relevant software engineering methods, improve their early-career

A. Learning Outcomes

We classified learning outcomes under cognitive competences (or knowledge), functional competences (or skills), and social and meta-competences (or competences) [14] as follows. After completing the course participants will be able to

1) compare Continuous Deployment and DevOps with other agile approaches
2) explain the benefits and barriers of automation in software engineering
3) interpret continuous delivery pipeline as business experimentation system
4) construct a continuous delivery pipeline
5) apply professional tools for build, test, and deployment automation
6) utilize continuous delivery pipeline in a small software project
7) demonstrate DevOps mindset in their work
8) recognize healthy software team behaviour and adjust their own behaviour accordingly
9) reflect on their own and team's performance and learning

The first three are cognitive competences, the following three functional competences and the last three meta-competences. On the SOLO taxonomy [15], the learning objectives map onto levels 2 - 4 (unistructural, multistructural, and relational) as follows. Learning objectives 2 and 5 are on unistructural level which means that students learn to recognise and name isolated facts or follow simple rules but not properly relate or apply the learning. Learning objectives 3, 4, 7, 8, and 9 are on the multistructural level where students can explain and make sense of things and information. Finally, learning objectives 1 and 6 are on the relational level where students can recognize relationships, connect ideas to each other and apply the learning in a familiar context.

B. Study Topics

The course consisted of ten full contact teaching days and individual learning time where the students conducted exercises and read the reading material. Contact teaching included the following half-day themes.

- Introduction to DevOps and Continuous Delivery
- Revision of Agility
- Value streams, continuous flow and feedback loops
- Lean Startup and Lean Canvas exercise
- CD pipeline as experimentation system
- Behaviour-driven development
- Cucumber exercises
- Cloud architectures and microservices
- Microservices programming and orchestration
- Introduction to Continuous Delivery Pipeline
- Containers and Docker
- Version control and Git basics
- Testing and Jenkins
- Advanced Git
- Advanced Docker and Kubernetes
- Advanced Jenkins and workflow
- Using the pipeline in software project exercise (full day)
- Revision
- Written exam

The topics were aligned with learning objectives and selected to cover the focal multidisciplinary aspects of DevOps and Continuous Delivery including business and experimentation aspects, technical aspects, and soft skills. We organized the teaching so that the first author (university teacher) organized the learning activities during the first week and the second author (industrial consultant) acted as an teaching assistant. On the second week the roles were reversed.

On the second week, students acquired practical skills with pipeline tools. Students used Bash on their own laptops to run commands in the virtual environment. The pipeline consisted of source code control (Git), operating system level virtualization (Docker), and continuous integration server (Jenkins). The students first learned to use each tool during a half-day introduction and then the students utilized their skills on a small software project using the pipeline together. Finally, the students learning was evaluated in individual assignments where they had to apply their learning in another context.

C. Assessment

The course was graded on the Danish seven-point grading scale[2] and students performance was assessed based on practical assignments (open book) and a written exam (closed book). Practical assignments were used to assess learning objectives 4 - 6 and written exam for learning objectives 1 - 3. Learning objectives 7 - 9 were evaluated formatively in group discussions and in a written portfolio that was not graded. Formative evaluation was also used throughout the course for all learning objectives. Participation to 80 % of the contact teaching was required to take the exam.

IV. Execution and Evaluation of the Course

The course ran over two calendar weeks in August 2018. We had 15 students of which 13 were our own and 2

[2]http://eng.uvm.dk/general-overview/7-point-grading-scale

were exchange students. Of the students, 7 were advanced level bachelor students and 8 master students. The majority (14) were software engineering students and 1 was computer science student. The age range was 22 to 32 years with a mean of 24 years. Two of the students were female and 13 male. 11 students provided us feedback on the course.

Overall, the students were happy with the course. They felt that the course had delivered what they expected. They were particularly happy about learning both practical and academic skills. One of them commented that they for instance use version control on their projects but it is something that lecturers usually assume them to learn on their own. Therefore, although many of the students told in the beginning that they have used Git for years, they said after the course that they learned now that their skills with Git were extremely shallow. The biggest complain was the workload of 125 study hours over two calendar weeks. The international organization at our university decided on the duration and scope of the course and we could not influence the decision. On a scale from 1 not at all to 7 completely, we got the following means (M) and standard deviations (in brackets).

- The course content met my expectations M = 6.4 (0.7)
- The course level met my expectations M = 6.4 (0.7)
- I was satisfied with the teachers M = 6.7 (0.9)

Content-wise the students wished that the practical and theoretical days would be mixed. In our implementation, the first week was theoretical and the second week was practical. The students were also quite advanced in agile and wished that the revision on agile could be shorter. This is however challenging to change as typically the extent of agile understanding on software engineering students can be variable and we believe that getting the concept of agility is crucial for learning DevOps and Continuous Delivery. Some of the students considered the reading material too wide for such a short time frame. We agree on this; two-week timeframe is too short for 125 learning hours and if we need to squeeze such a course into two weeks in the future, we would use more time to summarize the most focal parts of the reading material into a handout. However, the best option would be to allow more time for the summer school and organize it over three weeks instead which would allow the students to take the course within the regular 40-hour study weeks instead of having to work 60 hours per week.

From the examination we observed that the students had difficulties in explaining Infrastructure as Code. Given that cloud and network infrastructure was the most sought after skill in Hussain et al.'s [5] study, we need to improve our teaching regarding this for the next year (part of learning objective 2). In general, the students were able to describe the idea of the automated delivery pipeline and its benefits (learning objectives 3 and 4). As to learning objective 1, the students performance split up. About 60% of the students were able to compare DevOps to other agile methodologies to relational level whereas the rest had difficulties in it. This might indicate that although the students considered their previous knowledge on agile good, they were not able to assess their missing skills reliably. This could be alleviated by making the revision part more interactive and student-centered which could help the students in identifying their knowledge gaps.

Formative assessment was used to foster and evaluate the learning of DevOps mindset. We saw enthusiasm and team player mindset grow in the students. One student stated in their feedback that normally they do not participate in classroom discussions or ask questions but because of the easygoing and trusting atmosphere amongst the students and the teachers, they felt comfortable to participate. Thus, we agree with Clear [4] that teaching DevOps mindset is not straightforward or easy to assess. Nevertheless, we are positive that it can be fostered in the learning environment and as Clear states, it is something that educators need to be conscious about when planning and conducting their teaching and which we are going to continue in our next year's implementation.

Both authors have multidisciplinary and pedagogical background and working experience in higher education which we felt supported our collaboration and helped in designing a relatively coherent course. To modify the course in a way that would allow running it as part of the regular academic year would still necessitate a different approach. We were able to utilize teachers from a company locating in another city than the university as the course was run on consecutive days. Travelling every week for teaching would not be feasible. Also, running the servers as a service is more straightforward on consecutive days. In addition, the overhead of keeping up with the industrial state-of-the-art at universities is often too high. As Christensen [2] states, Docker version changed multiple times during their course which typically is something that educators try to avoid even between teaching years. Moreover, although Git, Jenkins and Docker might be representative tool selection this year, it could be that in a year or two, the industry has adopted another toolset. In the continuous deployment era, universities need to explore feasible ways to keep up with the technological development in a way that allows them to concentrate on fundamentals and underpinning theories that help the students in lifelong learning but at the same time also keep the employability of a newly graduated in mind.

V. Conclusion

This paper discussed teaching Continuous Delivery and DevOps in university-company collaboration. The motivation was to improve the industrial relevance and employability of newly graduated software engineers and to explore the possibilities of multidisciplinary teaching and educational collaboration of academia and industry. This paper presents one of the first investigations on teaching DevOps at universities. We conclude that in our case, the collaboration was fruitful and lead to good student satisfaction and high learning outcome. Supporting factors were found to be collaborators' multidisciplinary and practitioner's background in education. Future work includes further improvement of the course and determining how to implement it as part of the regular curriculum.

REFERENCES

[1] Debois, P. (2011). Devops: A software revolution in the making. Journal of Information Technology Management, 24(8), 3-39.

[2] Christensen, H. B. (2016). Teaching DevOps and cloud computing using a cognitive apprenticeship and story-telling approach. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (pp. 174-179). ACM.

[3] Smeds, J., Nybom, K., & Porres, I. (2015, May). DevOps: a definition and perceived adoption impediments. In International Conference on Agile Software Development (pp. 166-177). Springer, Cham.

[4] Clear, T. (2017). Meeting employers expectations of devops roles: can dispositions be taught?. ACM Inroads, 8(2), 19-21.

[5] Hussain, W., Clear, T., & MacDonell, S. (2017, May). Emerging trends for global DevOps: a New Zealand perspective. In 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE) (pp. 21-30). IEEE.

[6] Kunttu, L. (2017). Educational Involvement in Innovative University-Industry Collaboration. Technology Innovation Management Review, 7(12): 14-22.

[7] Pazur Anii, K., Divjak, B., & Arbanas, K. (2017). Preparing ICT Graduates for Real-World Challenges: Results of a Meta-Analysis. IEEE Transactions on Education, 60(3), 191-197.

[8] Eurostat. (Jan. 21, 2016). Almost 8 Million ICT Specialists Employed in the EU in 2014. Available at: http:// ec.europa.eu/eurostat/documents/ 2995521/7141198/4-21012016-AP-EN.pdf

[9] Carbone A. & Sheard, J. (2002) A studio-based teaching and learning model in IT: What do first year students think? in Proc. 7th Annu. Conf. Innov. Technol. Comput. Sci. Educ., 2002, pp. 213217.

[10] Deng, C., Mei, Y., and Xie, Q. (2009). Job position oriented experiment courses of computer science, in Proc. Int. Conf. Inf. Eng. Comput. Sci., Wuhan, China, 2009, pp. 14.

[11] Intayoad, W. PBL framework for enhancing software development skills: An empirical study for information technology students, Wireless Pers. Commun., vol. 76, no. 3, pp. 419433, 2014.

[12] Koppi, T., Edwards, S. L., Sheard, J., Naghdy, F., and Brookes, W. (2010). The case for ICT work-integrated learning from graduates in the workplace, in Proc. 12th Aust. Comput. Educ. Conf., Brisbane, QLD, Australia, pp. 107116.

[13] Simpson, M., Burmeister, J., Boykiw, A. and Zhu, J. (2003). Successful studio based real-world projects in IT education, in Proc. 5th Aust. Conf. Comput. Educ., Adelaide, SA, Australia, 2003, pp. 4151.

[14] Jeschke, S. (2015). Engineering education for Industry 4.0 Challenges, chances, opportunities. Presentation slides at World Engineering Education Forum. Available at: http://www.ima-zlw-ifu.rwth-aachen.de/fileadmin/user _upload/INSTITUTSCLUSTER/Publikation _Medien/Vortraege/download/ EngEducationInd4.0 _22Sept2015.pdf

[15] Winterton, J., Delamare-Le Deist, F., & Stringfellow, E. (2006). Typology of knowledge, skills and competences: clarification of the concept and prototype. Luxembourg: Office for Official Publications of the European Communities.

[16] Biggs, J. B., & Collis, K. F. (2014). Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome). Academic Press.

[17] York University. Course Director Handbook. Available at: http://teachingcommons.yorku.ca/for-cds/workshops-and-courses-for-cds/events-2/nfo/faculty-information-gateway/handbook-for-course-directors/

[18] Campbell, S. (2016). Teaching Cloud Computing. IEEE Computer, 49(9), 91-93.