DevOpsUse for Rapid Training of Agile Practices Within Undergraduate and Startup Communities

Peter de Lange(✉), Petru Nicolaescu, Ralf Klamma, and István Koren

Advanced Community Information Systems (ACIS) Group,
RWTH Aachen University, Ahornstr. 55, 52056 Aachen, Germany
{lange,nicolaescu,klamma,koren}@dbis.rwth-aachen.de
http://dbis.rwth-aachen.de

Abstract. Establishing a common practice between (startup) companies and universities in applied computer science labs has been tackled by pedagogical approaches based on the communities of practice theory. However, modern agile and distributed software engineering methods and recent developments like DevOps demand focused training of undergraduate students to enable them joining practices in companies. In this paper, we present the Community Application Editor (CAE) embedded in a DevOpsUse methodology supporting this form of basic training for bachelor students of computer science. We have evaluated the methodology and the tool usage in a first-stage undergraduate lab course. The results indicate that the students had a much smoother transition when later joining the second-stage lab with real companies.

Keywords: Community of practice · MDWE · End user development · Case study · Entrepreneurship

1 Introduction

Universities with a technical focus or curriculum have an important influence on the knowledge and experience of their students, building their theoretical and practical foundation to be later used in the industry. There is a two-way benefit established from cooperations between academia and companies: real-world practice and requirements can be incorporated into university courses and teaching, preparing the students for their later employment; and companies can later make use of innovations and state-of-the-art practice that result from university research projects. Following these aspects, previous research showed that computer science students can take contact with industry within the curricula and encourage entrepreneurship following socio-cultural theories of learning [1]. Based on these foundations, a series of lab courses were held yearly at RWTH Aachen University, where groups of students were forming communities of practice together with local start-up companies for developing IT projects [1]. These

K. Verbert et al. (Eds.): EC-TEL 2016, LNCS 9891, pp. 570–574, 2016.
DOI: 10.1007/978-3-319-45153-4_65

follow the examples from universities with a tradition in developing entrepreneurship teaching, like the MIT Entrepreneurship Lab [2] and facilitate several groups of computer science students at a Master of Science level to work on a concrete project task for and together with startup companies [1].

2 Supporting DevOpsUse in CoPs: A Methodology

DevOps is an emerging paradigm in software development which minimizes the gap between development and operation during agile software engineering processes. It tries to establish a new culture by a tighter integration of software development and deployment resulting in faster release cycles. The term DevOps comprises not only the methodology, but a mindset of working towards the same goal, and a collection of software tools that support this collaboration culture. Due to the missing notion of end user involvement in DevOps, we introduce the extended DevOpsUse approach that aims to unify agile practices of developers, operators and end users. We have used the DevOpsUse methodology in our practical course for teaching agile community-oriented software development. For this purpose we used Requirements Bazaar, a browser-based platform for prospective feedback, developed at our institute [3]. The Bazaar aims at supporting all stakeholders in reaching their particular goals with a common base: end users in expressing their particular needs and negotiating realizations in an intuitive, community-aware manner; service providers in prioritizing requirements realizations for maximized impact.

Further, to support a CoP in scaffolding their own Web applications and rapidly prototype ideas and architectures, we developed the CAE [4] for modeling and generating widget-based, collaborative community Web applications. We use near real-time collaborative modeling such that developers and community users can benefit from a structured approach to redesign existing applications or develop new ones. The architecture of the resulting community applications is constructed with regard to the following three key aspects: A RESTful microservice architecture backend based on las2peer[1] Web-services, a widget based frontend composed of multiple Web widgets running in a widget space and finally near real-time communication and collaboration support via the integration of collaboration frameworks.

Figure 1 shows the integrated perspective of our methodology. Our main target is to consider the learning aspect of using such a methodology, by involving students into the agile process and enable them to pursue it with the gained knowledge in industry. The rationale is to support the CoP in its development process by providing a coherent and integrated set of resources, mainly reflecting the tasks of DevOpsUse with a focus on collaboration and community practice, for speeding up the overall workflow.

[1] https://las2peer.org.

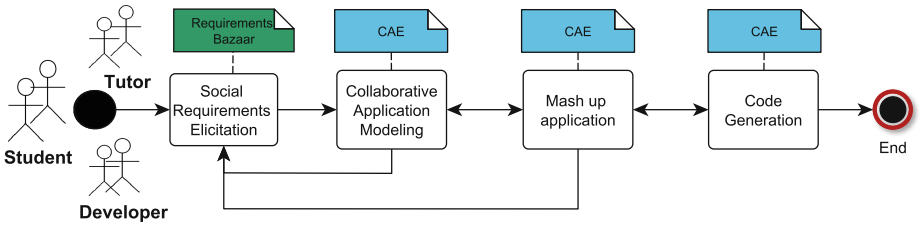

Fig. 1. Joint social requirements engineering and CAE methodology

3 Application in Lab Courses

For more than 15 years, RWTH Aachen University hosts a yearly five-months practical course on entrepreneurship for graduate students. Since 2011, we introduced a course for undergrads, that first gets students acquainted with our methodology described in the previous section, before they join the master students' projects to apply their knowledge in practice. In the following, we focus on the undergraduate course.

The lab starts with forming groups of about three students, each group working independently on a given project. This project is split up into different subtasks with an average working time of two weeks. At the end of each subtask, a review takes place where students and advisors come together to evaluate the current state of the project. The subtasks build up on each other, starting with a requirements analysis and design phase, then going over to teaching basic infrastructure setup and the basics of Web services.

About half way through the semester, the modus operandi of the course is changed from the structured tasks in the 'sandbox' environment to the real-world problems of local startups. This way, the students of the undergraduate course can apply their gained knowledge of the first half of the course in the context of a bigger project. The master students learn how to deal with the real-world situation of people coming into a project at a late stage, where much of the work is already done and the CoP has already evolved and established their working practices. In parallel, the undergrads continue working on their last subtask by refining it. At this stage, no strict requirements are enforced, giving the students the opportunity for creative problem-solving. The course finishes with a joint presentation of the produced software artifacts performed in short pitches.

4 Evaluation

We evaluated our teaching methodology, tools and the MDWE approach in the Winter semester of 2015–2016 with five bachelor students from RWTH Aachen University, split into two groups. Students were required to refine a short initial description in a requirements elicitation phase, via collaborative collection and discussion of requirements using the Requirements Bazaar (c.f. Fig. 1). Later on, we introduced CAE to each group in an one hour collaborative session. This was

designed as an example for community formation around the software artifacts and had also the goal to familiarize the students with the technology. As the session was conducted by a tutor (i.e. expert), students could ask questions. After this, students were required to work within their groups to design and realize the complete microservices and the corresponding frontends. After handing in the final application, students were required to complete a questionnaire about their experience with our methodology.

At the time we performed the evaluation using CAE, students were already familiar with Web development using RESTful services, Javascript and HTML from previous tasks. However, as the questionnaire showed, they were not familiar with collaborative modeling as a tool for requirement analysis, system architecture design and MDWE. The teaching with CAE was rated very high from the understandability of separation of concerns between components (4.4/5) and the simplicity of the modeling framework that lead to a quick understanding of the concepts which were explained or designed (4.4/5). Results above average were obtained for learning how to design widget environments, understanding how the relations between microservices and frontend code are realized and the usability of the modeling framework for application redesign and development.

The code generation aspects of CAE were also considered to be relevant in speeding the development process and understanding technical notions. Among the advantages of CAE, students mentioned the redesign of applications, for which the tool is very useful. The collaborative work on the same resource at the same time was also considered to be helpful for learning purposes. Students suggested to improve the usability by adding a Wiki and emphasizing use-cases for classic Web applications, which do not involve widgets.

5 Conclusion and Future Work

In this paper, we presented a methodology and tool support for teaching undergraduate students state-of-the-art approaches for requirements elicitation, design and development of Web applications using cutting edge practices and technologies. Our main findings are that relevant tool support, social requirements, near real-time collaboration and collaborative MDWE approaches provide a solid foundation for bridging the gap between academia and industry and is able to rapidly train students for joint work with agile startups. In the future, we plan to further evaluate our method within our practical courses and to deeper investige the role of near real-time collaboration and end user development in formal teaching scenarios.

References

1. Rohde, M., Klamma, R., Jarke, M., Wulf, V.: Reality is our laboratory: communities of practice in applied computer science. Behav. IT **26**(1), 81–94 (2007)
2. Roberts, E.B.: Entrepreneurs in High Technology: Lessons from MIT and Beyond. Oxford University Press, Oxford (1991)

3. Renzel, D., Behrendt, M., Klamma, R., Jarke, M.: Requirements Bazaar: social requirements engineering for community-driven innovation. In: 21st IEEE International Requirements Engineering Conference, RE 2013, pp. 326–327(2013)
4. de Lange, P., Nicolaescu, P., Derntl, M., Jarke, M., Klamma, R.: Community application editor: collaborative near real-time modeling and composition of microservice-based web applications. In: Modellierung 2016 (2016)