TEACHING DEVOPS AND SOFTWARE ENGINEERING PRACTICES USING AN AUTOMATED PROGRAMMING ASSESSMENT SYSTEM

Bao-An Nguyen¹, Thuy-Vi Ha Thi²∗

Abstract – The software industry has seen a shift in recent years, with the adoption of agile development practices and the deployment of cloud-based, service-oriented applications using DevOps. This has increased pressure on educators in higher education institutions to provide students with the best possible preparation for their careers, based on industry best practices. This paper proposed a new approach to bridge the gap between in-class exercises and industrial practices by introducing the automated programming assessment system, ProgEdu. ProgEdu simulates the continuous integration pipeline of DevOps, incorporating unit testing and code quality assessment. It is deployed as a service-oriented application on Docker, allowing students to gain practical experience in a real-world setting. The approach not only involves assessing the code but also incorporates the service-oriented architecture, continuous integration pipeline, and Docker of the system to illustrate various DevOps practices in our software engineering classes. By providing hands-on exercises that utilize industry-standard tools and techniques, the approach can bridge the gap between theoretical knowledge and practical application, helping students to better understand key concepts and prepare for the demands of the modern workforce.

Keywords: automated programming assessment system (APAS), continuous integration, DevOps, software engineering education.

I. INTRODUCTION

Software engineering educators always aim to bring theories and real-world practices to students, so that they can gain competencies and skills that meet the requirements of the industry [1]. However, many SE curriculums have been derived from computer science (CS) programs and, hence, focus on theoretical topics of CS and computational instead of carrying practical software engineering (SE) experiences to students [2]. Consequently, the problem stated by Ford et al. [3] three decades before is still partially true today: ‘Somewhat oversimplified, industry needs software engineers, but universities are supplying computer scientists. Thus, it’s time to promote the widespread development of software engineering degree programs’ [4].

It is important to identify SE topics that should be emphasized in schools and the gap between institutions and industrial needs regarding SE education. The systematic literature review conducted in 2019 with over 4,000 data samples from 13 countries drew a figure that software requirements, design, and testing are vital skills [4]. The greatest knowledge gaps appear in configuration management, SE process, design, testing, management, quality, and SE professional practices [2]. These results emphasize the need for better preparation for students’ SE future careers to increase their employability.

In line with the aforementioned facts, Sommerville [5], the author of software engineering textbook, has shifted his perspective on SE education by writing a new book titled Engineering software products: An introduction to modern software engineering [6] instead of publishing the eleventh edition of the original book. The new book focuses on important activities in the production of SE products. The shift reflects the need for a more modern approach to SE education and the recognition of changes in the industry. Teaching software engineering can be
a challenging task for educators, as they face difficulties in attracting students’ engagement and designing practical activities [1]. In particular, the lack of tools or software deployment environments can pose a significant obstacle. For example, creating a complete DevOps process, as seen in large companies, may be beyond the scope of academic institutions.

To address these challenges, educators can explore various strategies to engage students, such as incorporating iterative and collaborative activities into their teaching methods [7]. They can also leverage online resources and tools to enhance students’ hands-on experience, even if the resources may not be available within the academic institution. Additionally, partnerships with industry and other educational institutions can provide opportunities for access to necessary tools and expertise [8]. By adopting a flexible and creative approach, educators can better equip students with the skills and knowledge needed to succeed in software engineering. With the acknowledgment that appropriate competence in using tools and familiarity with industrial practices plays an indispensable role in student success, the research has redesigned software engineering courses to align with the topics covered in the book [6]. The syllabus covers a broad range of tools to support hands-on student practices and keep up-to-date with trendy tools and practices within the software engineering community. According to the 2022 Developer Survey by StackOverflow [9], fundamental tools like Git and Docker are widely used, while CI/CD, DevOps, and automated testing are prevalent in the working environments of most professional developers. With this in mind, teaching these tools and practices to our students to prepare them for real-world SE jobs is prioritized.

Initially, the study encountered the challenge of providing students with practical exercises, including a broad range of topics in SE. Fortunately, a few semesters ago, our former team developed an automated programming assessment system (APAS) called ProgEdu based on the DevOps process to enhance the quality of students’ code [10, 11]. ProgEdu is a service-oriented system deployed on Docker, which integrates almost the entire process of CI, automated code review, unit testing, and static code quality assessment. Additionally, ProgEdu enables collaborative and iterative learning, allowing students to work on group projects and learn from their mistakes [12]. As ProgEdu is an open-source product, it has been used as a case study in the new SE syllabus starting this semester, aiming to provide our students with hands-on experience in using a real-world tool and enhance their understanding of SE concepts and practices. The service-oriented architecture, Docker deployment, and automated code review process make ProgEdu an excellent tool for teaching SE and DevOps practices. ProgEdu is considered a case study that will help the students develop practical skills that will be valuable in their future careers.

This article showcases how ProgEdu can be creatively used to teach SE and DevOps practices. The approach demonstrates that a faculty-developed tool can be utilized in teaching various subjects and topics effectively, helping educators overcome learning barriers, enhancing the learning experience, and improving learner’s SE skills.

II. BACKGROUND

DevOps has gained prominence in the industry and has been standardized by IEEE [13] as an ‘interdisciplinary approach and means to enable the realization of successful software systems’, there is a growing need to introduce DevOps concepts and practices in university courses to prepare students for the job market. It has become among the first competencies required by the industry, belonging to the Agile method [14]. However, teaching DevOps in universities presents several challenges, including the lack of a standardized curriculum, the need for hands-on experience, environment setup, tool/technology, and assessment [15].

The syllabus introduced by Schilling tried to integrate modern development practices into a SE curriculum [16]. The course covers DevOps concepts and practices with the whole process
from start to finish of a software product, which is designing and building a development pipeline, development, testing, and deployment. Moreover, the process should be conducted securely, hence, DevSecOps is the right term.

To facilitate students with multi-discipline practices like DevOps, say SE, software project management, and cloud computing, a hands-on approach with real-world projects and tools is the key [17]. Jennings et al. [18] promote using hands-on experience and collaborative learning in teaching DevOps. They recommended using Git for source code management, Jenkin for CI, Ansible, Docker, and AWS for automated virtual machine provisioning, and Capybara for automated testing. By providing students with practical experience, they can develop skills and knowledge that are not easily acquired through theory independently. Collaboration among students and industry professionals can also provide valuable insights and promote critical thinking.

The teaching approach in SE is aligned with the recommendations discussed in previous works, with the belief that hands-on experience is vital for students to gain a better understanding of DevOps practices. To facilitate this approach, ProgEdu was utilized. ProgEdu has a pre-configured system that is specifically designed to follow DevOps practices, making it easier for us to expose our students to the practical aspects of DevOps.

III. METHODOLOGY

Motivated by the advantages of using Git in classrooms [19–21], the research developed ProgEdu, a Git-driven APAS designed to monitor and evaluate students’ programming assignments and team projects. The primary goal is to alleviate instructors' workload in assessing student homework while keeping track of their progress in learning programming. The system was designed based on the following principles:

- R1. Promoting iterative learning: By recognizing the significance of iterative learning, which allows students to learn from their mistakes and improve their work iteratively until all requirements are met, the study aims to enable students to submit student code and receive feedback as many times as necessary, which can be achieved through a CI pipeline where students' code can be continuously analyzed, built, and tested.

- R2. Applying DevOps practices: Inspired by the loop of the iterative learning process, the DevOps loop has been employed (Figure 2), where students participate in the development (Dev) phase, while the operation (Ops) phase is automated by tools in the CI/CD pipeline.

- R3. Applying service-oriented architecture: To facilitate Agile software development and make it easy to add or remove features and services, ProgEdu has been developed on a service-oriented architecture and deployed in containers using Docker.

- R4. Promoting reflective learning: In addition to code assessment feedback, ProgEdu provides students with comprehensive information about
their learning progress. Students’ engagement with the system is recorded, analyzed, and presented to them through learning analytic features, promoting reflective learning. In summary, ProgEdu is a comprehensive APAS that leverages Git, DevOps, service-oriented architecture, and learning analytics to promote iterative and reflective learning while reducing instructors’ workload in assessing student homework. The architecture of ProgEdu is depicted in Figure 3.

The ProgEdu portal: To facilitate team administration and project monitoring, the portal was developed as a learning management system. The student management module and group management module allow instructors to manage enrolled students and perform team formulation tasks. The visualization management module provides visualized reports letting the instructors and students track the project progress. While instructors can track the progress of all groups, students can only view the information of their group.

The version control server – GitLab: The study employed GitLab as a source code repository that receives students’ source code committed by Git commands. The GitLab repository is linked with the CI server to perform immediate code analysis. The CI and assessment server: The CI server Jenkins is triggered by code commits to GitLab and fetches the committed code to the assessment modules. For Java programming assignments, the open-source build tool Maven is leveraged to build the code analyzer. The instructor has to provide test cases to enable the unit test. Checkstyle was adopted for static code analysis in which all violations in the coding convention are detected and presented to students. For web programming assessment, HTMLHint and stylinst were employed as static code analysis tools for HTML and CSS code assessment, respectively. The code quality status of the project and feedback information regarding coding style failures are displayed in the interface of each team after submission. Besides, to make students familiar with industrial metrics of code quality, such as bugs, vulnerabilities, and code smell, SonarQube was leveraged to analyze both code quality and code security. As not all the issues reported by SonarQube are easy to resolve for sophomore students, this research adopted the status of a project determined by the checking results of HTMLHint and stylinst as representative data for contribution analysis.

The log extractor: Since all activities of students in committing their project artifacts are recorded in the logs of GitLab and Jenkins, this information is continuously collected by the log extractor to enable feedback to the portal. The contribution analyzer: Using data given by the log extractor, the contribution analyzer computes the contributions of each team member based on their effort in source code committing and improving project code quality.

By embedding the CI practice, the system enables a real-time feedback mechanism for any change in the status of the projects. After each committing, students can access the system portal to see whether their submitted code conforms required coding styles. The status of projects and member contributions are visualized to facilitate students in monitoring their team progress.

IV. RESULTS AND DISCUSSION

The revised syllabus for the SE course is based on the recommendations outlined in the textbook [6]. The focus of the course is on essential practices for developing modern software products. The course is delivered over a single semester (15 weeks) through a combination of lectures,
By aligning ProgEdu with chapter topics, the study aims to cultivate critical hands-on skills essential for DevOps, including code quality assurance, automated testing, virtualization, Docker, and CI/CD. Moreover, the practical approach of ProgEdu ensures that students gain experience in applying theoretical concepts in a real-world setting, providing valuable experience that is highly sought after by employers.

As an illustrative instance, the study examines a hands-on experience. Students are required to complete exercise E3, which involves developing a Docker Compose file to manage and orchestrate the deployment of ProgEdu’s services. This exercise will require configuring a series of services, including GitLab for code repository, Jenkins CI server, a front-end website for the learning management system, MySQL database, MongoDB for log data, and various code quality assessment tools. Through this exercise, students will gain experience with a wide range of tools and services used in industry, providing them with valuable skills for their future careers. A sample of the Docker Compose file is shown in Figure 5.

For another example, Topic which covers concepts and techniques of testing. In exercise E6, students are assigned to write test cases for a set of programs and submit them to ProgEdu. They are then required to write code that satisfies the test cases, implementing a form of test-driven development (TDD) where tests are written before the actual code. By writing test cases before the code, students learn to think about the intended behavior of their code before...
actually implementing it, resulting in more robust and reliable software. This exercise provides students with hands-on experience in a development practice that is widely used in industry.

```
package selab.myapplication;
import static org.junit.Assert.assertEquals;

public class AppTest {
    public static void main(String[] args) {
        App app = new App();
        Account account = new Account(100.0);
        account.withdraw(50.0);
        assertEquals(50.0, account.getBalance());
    }
}
```

![Fig. 6: A sample test case fetched to ProgEdu in hands-on E6](image)

The aforementioned examples show the applicability of ProgEdu in teaching SE and DevOps practices. ProgEdu’s hands-on exercises provide students with valuable practical experience in DevOps practices, such as automated testing, continuous integration, and docker deployment. By working with ProgEdu, students can learn how to use industry-standard tools and techniques in a real-world environment, preparing them for the challenges of modern software development. Besides, ProgEdu’s focus on code quality assurance and virtualization aligns well with the primary principles of DevOps, making it an effective tool for teaching both SE and DevOps practices.

V. CONCLUSION

This paper has proposed a new approach to incorporating our APAS, ProgEdu, into a SE course to provide hands-on learning experiences and bring industrial tools and techniques to students, promoting a situated learning environment. ProgEdu’s hands-on exercises enable students to gain practical experience with industry-standard tools, such as GitLab, Jenkins CI server, and Docker Compose. This experience is essential for preparing students to meet the demands of modern software development practices. ProgEdu’s focus on code quality assurance, automated testing, virtualization, and CI/CD also helps students develop critical skills for success in DevOps. By using ProgEdu, students can learn how to use these tools in a real-world environment, enabling them to understand primary concepts and techniques. The situated learning approach of ProgEdu’s hands-on exercises effectively bridges the gap between theoretical knowledge and practical application, preparing students for successful careers in software development. The incorporation of ProgEdu into SE and DevOps courses is just one example of how technology can be used to enhance the learning experience and provide students with practical, hands-on experience with industry-standard tools and techniques. Nevertheless, it is important to note that the effectiveness of this approach has not been evaluated yet, as the course is still ongoing. Future work will include an evaluation of the impact of ProgEdu on student learning outcomes and their perspectives on this approach.

Last but not least, this paper wants to make a call to educators to explore innovative approaches to teaching and creatively utilize the available tools and resources to benefit students. As educators, it is important to constantly seek new ways to engage students and provide them with the knowledge and skills they need to succeed in their chosen field. By embracing new technologies and teaching approaches, educators can contribute to students’ well-preparation for the challenges of the modern workforce.

REFERENCES


