The Adoption of Open Source Projects in Engineering Education: A Real Software Development Experience

Debora M. C. Nascimento
UFS – Federal University of Sergipe
São Cristovão, Sergipe
Brazil 49100–000
dmcnascimento@ufs.br

Christina F. G. Chavez
UFBA – Federal University of Bahia
Salvador, Bahia
Brazil 40170–110
flach@ufba.br

Roberto A. Bittencourt
UEFS – State University of Feira de Santana
Feira de Santana, Bahia
Brazil 44036–900
roberto@uefs.br

***Abstract*—This research to practice full paper investigates software engineering students' perceptions of their contact with open source projects as a real-world experience. Working with open source projects (OSPs) has been shown as an interesting option in software engineering courses to bringing students closer to more realistic environments. However, when instructors use this approach, it is not clear whether students perceive the OSP as a real industrial software project, or whether they perceive the tasks they perform over OSPs as typical or close to industrial software project activities. The goal of this work was to identify students' perceptions of their interaction with an open source project as a real world experience. To do so, we performed three mixed-methods case studies with three different undergraduate classes. Each class had a different focus: i) software maintenance and evolution, ii) software testing, and iii) reverse engineering of software requirements. Results show that students perceived features that make OSPs close to industrial projects, realized that their OSP tasks are close to the ones in industrial projects, and also faced difficulties typical of working with real world software. Moreover, students forged a view of the skills needed for their future professional success. We conclude that students realized that performing tasks in OSPs was a real world experience they took part, contributing to their background both for the competencies they acquired and the difficulties they had to overcome.**

I. Introduction

For about two decades, the literature has reported on software industry dissatisfaction with the competencies of recent graduates [1], [2]. A more recent systematic review has identified several studies that point to a gap between the skills and knowledge presented by graduates and industry expectations as well as some areas where recent graduates are weak or perform poorly [3].

In the various courses of undergraduate computing programs, students are used to writing relatively small programs, working individually, and not following formal development processes to accomplish the tasks proposed by instructors [4]. It is rather common for students to believe that an apparently working program is enough to reach an acceptable solution [4]. Traditionally, the typical software engineering (SE) course is based on extensive content exposure and the realization of a practical project. Usually, projects are small, cover a small group of students, have their requirements defined by instructors and students involved, follow the waterfall development model and are developed during only one academic term. Such projects do not reflect the actual situation of industrial projects, and have been often called *'toy projects'*.

Universities are not software industry companies, neither should they be, and it is very hard to fully reproduce all real-world circumstances in academia [2]. However, to improve student competencies, the literature describes several approaches to bring them closer to more realistic environments [2], [5]–[9]. Among the possibilities, the approach of using open source projects (OSPs) has proven an interesting option. Real source code and development environments are accessible over the Internet, with no restrictions of deployment location [10] or code ownership. Students can download and study the code, generate contributions and interact with the community of project users and developers.

In recent years, several studies have been published reporting the use of OSPs for teaching SE [11], [12]. One of their main goals is to make students' experiences as close as possible to future situations in work environments. However, when instructors apply this approach, questions arise about students' perceptions: do they see the OSP as a real project? Do they agree that the requested activities are close to the ones they will perform in the marketplace? Students' perceptions of the experience with an OSP as a real-world experience are capital to their understanding of the relevance of the proposed challenges. Otherwise, they will lack motivation to work with a larger and often more complex project than the ones they are used to.

In this context, the goal of this study was to investigate students' perceptions of their interactions with an open source project as a real world experience. To do so, we performed three mixed-methods case studies with three distinct classes, where students worked with an OSP to respectively learn about i) software maintenance and evolution, ii) software testing and iii) reverse engineering software requirements.

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

As a result of the analysis, students highlighted some features that approximate the OSP used with industrial software. Among them, we mention: 'size', 'complexity', 'code structure issues', 'multi-developer involvement', 'scarce documentation', 'user involvement' and 'software implemented with no simplification'. As for the activities they carried out, the students pointed out the 'real need to apply the studied practices', 'having to deal with existing software' and 'undocumented software', as well as particular issues of each case study, as close to situations they will face on the marketplace. On the other hand, closeness to reality led students to face a set of difficulties. The main difficulties encountered were: 'initial impact', 'difficulty to carry out tasks', the challenge of 'understanding code developed by third parties' and 'lack of documentation'. Finally, with the experience, students acquired a view of the needed skills for their professional future, whether particular such as 'determining what is to be tested and how it should be tested', or general such as 'knowing how to identify the problem and that apply to solve it'.

This paper is organized as follows. Section II presents the importance of real experiences in context of learning theories. Section III presents an brief description for the three case studies performed. Section IV presents the main results and a discussion of the main findings and limitations. Finally, Section V presents conclusions and areas for further research.

II. Background

According to several authors, learning results from experience [13]–[15]. This experience can be the interaction with something, participation in something, or exposure to some internal or external event [15].

The learner's cognitive structure, established by previous experiences, determines what he or she will learn. However, a teacher can influence learning by specifying new experiences the learner should face [16]. It is relevant to mention that not every experience brings good results to learning [13], [16], [17]. Experiences must be vivid, lively, interesting and must be connected with future experiences, especially with out-of-school situations [13].

According to the phenomenon known as situated learning, learning is strongly related to the context and can not be easily separated from it [18]. Consequently, learning must take place in situations close to the real world, so that students are able to transfer what they have learned to the real world. Likewise, students' required activities and skills should reflect those needed in the future. As a solution to these issues, McKeachie and Svinicki [18] propose the use of '*experiential learning*'.

Experiential learning represents situations in which the pedagogical project seeks direct engagement of students in experiences close to real world problems. According to McKeachie and Svinicki [18], reflection on experience is a key element for experiential learning, since learning from real-world experiences is not an easy task and requires considerable mental effort. For the transfer of what has been learned to happen, it is essential that there be reflection on what has been learned. Apprentices need to be aware that they are learning and what they are learning. That is, they must be aware that some learning goal was behind what was being done.

Thus, the use of OSPs in software engineering education appears as an experiential learning strategy and an opportunity to provide learners with a vivid experience, connected with possible future experiences. However, it is also important to capture learners' reflections on this strategy.

A. Related Work

Nascimento *et al.* [12] carried out a systematic mapping study, that pointed out the existence of several initiatives whose aim was to learn SE through the use of OSPs. Most studies correspond to either solution proposals or experience reports, so they report the particular issues of how they applied the approach and sometimes the lessons learned from using it. Other studies present a more general view of how the approach can be incorporated into the curriculum, or even how to fill the vocational training gaps pointed out by the software industry.

Previous work has, among other issues, analyzed whether an open-source development project delivers the appropriate kind of "real-world" experience that CS students need [19]. This survey has strongly influenced the definition of our research question, since although the OSP provides a real-world experience, it is also important to understand students' perspective, i.e., how they see this experience. Two studies investigated this perception in a descriptive way. In the first, the OSP was used to learn aspects related to software design [20], while the second adopted the OSP approach for learning software engineering in general [21]. The reported quantitative results provide a student view on this issue, yet they do not explore the reasons why students consider the use of OSP as a real world experience lived by them and its consequences. In this work, we sought to obtain both quantitative and qualitative insights of this perception, not only in one study, but in three case studies, where the approach was used to learn maintenance and evolution of software, software testing and reverse requirements engineering. It should be noted that we report the results of the cited studies throughout the presentation and discussion of our results (Section IV).

III. Methodology

To explore students' understanding of their interactions with an OSP as a real-world experience, we conducted three case studies. Table I provides an overview of the performed case studies. We selected the focus areas of SE knowledge to apply the approach according to the syllabus of each course. As participation in the research was voluntary, we had a reduced participation in data collection in the first study (approximately one third of the students) and a reasonable value in the other two (approximately half the students). For studies EC1 and EC3, the use of the OSP to perform activities was mandatory, whereas in the EC2 study, students could choose to either perform or not the requested activities. Finally, we observed some heterogeneity regarding academic maturity shown by the

Table I
GENERAL OVERVIEW OF THE PERFORMED CASE STUDIES

ID	Focus Knowledge Area	Place	Period	# Participant students	OSP Activity Choice	Academic Maturity
EC1	Software Evolution	Blind Review	2013, 2nd	9	Compulsory	Varied
EC2	Software Testing	Blind Review	2015, 2nd	15	Optional	Seniors
EC3	Software Requirements	Blind Review	2015, 2nd	15	Compulsory	Juniors

Table II
INTERVIEWED STUDENTS FOR CASE STUDY EC1

Student	Previous Experience
SE1	Projects only in undergraduate courses.
SE2	Projects also in undergraduate research and internship.
SE3	Projects also in undergraduate research, junior enterprise and internship.
SE4	Projects only in undergraduate courses.
SE5	Projects also in internship.
SE6	Large project experience; worked for three companies.
SE7	Projects also in internship.
SE8	Projects also in undergraduate research.

Table III
INTERVIEWED STUDENTS FOR CASE STUDY EC2

Student	Previous Experience
ST1	No experience with real projects
ST3	No experience with real projects
ST7	No experience with real projects
ST8	No experience with real projects
ST4	Some experience (up to two projects)
ST2	More experienced
ST5	More experienced
ST6	More experienced

Table IV
INTERVIEWED STUDENTS FOR CASE STUDY EC3

Student	Previous Experience
SR2	No experience with real projects
SR3	No experience with real projects
SR4	No experience with real projects
SR5	No experience with real projects
SR6	No experience with real projects
SR7	No experience with real projects
SR1	Some experience (up to two projects)
SR8	Some experience (up to two projects)

participating students. Because of the double blind review, we hide the locations of the case studies.

JabRef[1], a system for managing bibliographic references in the Bibtex format, was the OSP we chose to work the intended content. As the research team was responsible to define the activities to be carried out and to support students on the software system issues, we chose to use the same OSP in all the case studies. Thus, we eliminated the need to seek understanding code and documentation from other OSPs and decreased the number of research variables involved between studies.

The activities to be performed by students in the selected OSP depended on the focus of each study. For instance, student teams had to change several JabRef features in EC1, implement automated tests in EC2, and recover functional and nonfunctional software requirements in EC3. Each team was responsible for a given software module.

We used interviews and questionnaires as instruments of data collection. When we carried out the EC1 study, we were still narrowing our research objectives, thus, this study was mainly exploratory. In subsequent studies (EC2 and EC3), with a defined research goal, we adopted a more investigative strategy, but not discarding exploration, important for achieving the research goal. Hence, the questionnaire applied in EC1, unlike the other studies, did not directly seek to meet the goal of this research. In the EC2 and EC3 studies, the questionnaires were answered in three different moments: in the first contact with the students, before and after the intervention using the OSP. In the first moment, the intention was to identify students' previous experience with real software projects. In the second, the goal was to gather students' perceptions about the knowledge and skills regarding the studied subject, before applying the OSP approach. Finally, in the end, in addition to capturing their perceptions on the knowledge acquired and the skills developed during the intervention, we sought to identify whether students perceived the interaction with the OSP as a real world experience.

We used semi-structured interviews and performed them after the intervention. While in EC1, students were interviewed according to availability, in EC2 and EC3, we invited at least one member from each team. We also sought to cover students with different previous experience with real projects. (i) inexperienced, for students who had so far had no experience with real projects; (ii) some experience, for those who had participated in up to two projects previously and (iii) more experienced, for those who had participated in at least three real projects. For the EC1 study, as the experience was not quantified, we only describe the reported experience. All interviews were recorded, transcribed and reviewed. Tables II, III and IV present a characterization of the interviewed students, for each case study, with respect to their previous experience. We highlight that only students SE2, SR3 and SR6 were female students. Questions from the interviews related to the topic of this paper are available in

[1]http://www.jabref.org/

https://sites.google.com/site/fie2018osp/.

For data analysis, we used descriptive statistics for the quantitative data, and the inductive-deductive process described by Merriam for the qualitative data [22]. In the beginning, we used open coding, seeking to discover information that might be relevant to the research. After we reached a certain number of codes, we performed axial coding simultaneously with open coding, grouping the interrelated codes and creating a hierarchy of themes. As coding proceeded, we verified existing codes that could be used or we created new codes whenever needed. After completing coding of all data, we reviewed the resulting codes, refined the hierarchy of themes, and generate memos for the key themes. Throughout the process, we sought to identify relationships between codes and themes created. In the following, the term "categories" also refers to the created codes.

After data analysis and interpretation were individually performed for each study, we triangulated the results in order to identify intersections between studies and the particularities of each study. In the following section, we present these results and discuss them in the light of our research goal and of the existing literature.

IV. Results

Based on the analysis of the qualitative data, we identified four themes related to the students' views of their interactions with an OSP as a real world experience: (i) closeness of open source projects to existing industrial projects; (ii) closeness to activities performed in software industry; (iii) difficulties arising from the use of real projects; and (iv) view about needed skills. We present below the findings organized according to these themes.

Tables V, VI, VII, and VIII present results for the themes; each column shows codes resulting from a particular case study; codes in the same row correspond to intersections between studies, and a single code in a row represents a particular result of the respective case study.

A. Closeness of open source projects to existing industrial projects

In the studies, 93% (EC2 and EC3) and 89% (EC1) of the students considered that the project used (JabRef) is similar, with respect to size and complexity, to those in the software industry, while 100% (EC2 and EC3) believed that JabRef allowed them to have a real world experience (this question was not quantitatively investigated in the EC1 study). Related work shows similar results. In the study conducted by Carrington [20], 67% of the students agreed that the OSP allowed them to have a real world experience, whereas in Xing's [21] study, students strongly agreed that the use of OSP provided a real-world experience (mean value of 4.9 in a scale from 1 to 5). Excerpts from the interviews in the case studies corroborate those findings:

> *"(...) a real example as the instructor took JabRef (...)"* (SE6)

> *"I think it's cool because you can do it ... bring things ... that are used in the real world, and this makes a, makes a cool difference, I think this is ... brings you closer to reality (...)"* (SE1).

Likewise, studies qualitatively report that students appreciated the real-world experience provided [23] or recognized it as a benefit [24]. Table V summarizes the features present in the interviews which bring, in each study, JabRef closer to industrial software.

The following excerpts illustrate students' perceptions about the features reported:

> *"In this project, we got the notions of a big system that was developed by several people, and you really see what a system would be like when you get into a business environment, get into a company, you realize that you're going to face a problem similar to that of JabRef"* (SE6).

> *"It's very similar because it's very common for us to have poorly organized projects in industry nowadays ... If it has ... I've worked in some places that had projects from 1999, and projects that were growing without documentation, with new technologies coming up, and you're putting one thing on top of the other, and you have multiple customers, one customer wants one way, the other wants another way, and it goes messing around and becoming a big ... bomb (laughs), and the developer is the one who suffers, mainly because it greatly increases the expense ... the turnover is always high in these companies ... So I would say it is very similar (laughs)"* (ST5).

> *"In industry, we're going to work with a team, this team can be modified and it is more or less what happens inside ... in the open source code, it is not always the same people who changes, who starts changing the code, who will end it up (...)"* (SE2).

> *"(...) it's not gonna be very different from what will be found... in industry... not every project will be clear, nor everything documented in a clear way, that not only helps the user but also... the developer"* (SR5).

Reflecting on the categories presented in Table V, we noticed a homogeneous perception of those involved about features that bring OSP closer to industrial projects. Some features appeared in the three studies, others in only two. Only the features of 'requires environment configuration' and 'hard to understand' arose in a single study.

It should be noted that such features can not be generalized to other contexts. For instance, there may be well-structured projects in industry as well as well-structured open source projects. There may even be open source projects and industrial projects that are simple. In fact, they represent features that are usually found but greatly vary between projects.

We highlight a particularity in the EC3 study, in which one of the students pointed out that the complexity of the

Table V
FEATURES OF JABREF MAKING IT CLOSE TO EXISTING INDUSTRIAL PROJECTS

EC1	EC2	EC3
Size	Size	-
Complexity	Complexity	Complexity
Code structure issues – Architecture erosion	Code structure issues	Confusing structure
Several participating developers	Several participating developers	-
Documentation issues	Scarce documentation	Scarce documentation
-	Requires environment configuration	-
Used by various people	There is used feedback	-
-	Software implemented with no simplifications	Project is not simplified
Hard to understand	-	-

project was not suitable for beginning students. Although this opinion has not been shared by other students, it is relevant, since it may occur whenever a real project is used as a learning resource. In this sense, the balance between the project complexity and learning goals should guide the choice of the open source project (or projects) to be used.

On the other hand, students in the EC2 and EC3 studies highlighted the academic experience of dealing with projects close to industrial projects, projects that are not simplifications created for learning purposes, i.e., not the typical *'toy projects'* usually employed in software engineering courses.

> *"What we usually see in the classroom, right, are small projects, nothing commercial, not so robust, with several modules, or nothing"* (ST4).

> *"The projects I developed in the courses (...) are simple projects, like building a game, or then it's a tool to talk to the database"* (SR3). (...) *"And it is quite different, the complexity of the tool is much higher, much higher. There are many details that need to be addressed, so it's... well, my little project it's a lot simpler than the tool, of course. (...) And other types of tools that do things that are used a lot, that bring more features, ... they will certainly be more, more complicated"* (SR3).

> *"The big issue is always this, academia and industry. Within academia, projects tend not to be as large, except for capstone projects, masters and so on. Outside academia, the projects tend to be much bigger, much more complex (...) In academia, you see small projects; in industry, they are much bigger projects"* (ST7).

> *"It is good to interact with reality too, because not everything is flowers, right? Not every software is all cute, easy to maintain"* (ST1).

For one of EC2 students, working with real software is a "paradigm break" (ST7). According to a student in EC3, the current paradigm of using simplified projects raises the concern of not being able to deal with real projects when they go to industry, since they do not practice with this type of project when they are undergraduates (SR1).

In two studies (EC1 and EC2), similarly to what happens in industry, students have confirmed user involvement with the software. In the EC2 study, one student reminds of user feedback (ST3), whereas in EC1, students remind that the software is already used by several people. According to the EC1 instructor, finding the system's usefulness is important to increase students' interest. This fact is confirmed in the study by Santore et al. [25], where students were enthusiastic about implementing a real and useful project.

Other studies also point to those features as close to the real world. Ellis *et al.* [19] highlight the diversity and distribution of 'several participating developers', interaction with 'users involved', constant change of requirements, having to deal with 'poor documentation' and 'written by third parties' and, finally, having to meet quality professional standards. On the other hand, Buchta *et al.* [26] emphasize 'software size' and individual contribution in the context of a team.

B. Closeness to activities performed in software industry

In the first case study (EC1), 56% of the interviewed students agreed upon the closeness between OSP activities and activities performed in industry. Throughout interviews, students also recognized such closeness. However, they pointed out that the modifications themselves requested by the instructor were quite simple, although they required executing all the activities of the studied incremental change process.

> *"I think they're pretty close, I guess we've worked with examples... softer in terms of complexity, but... it's... I think the load you have to get to handle these simpler problems, it's the same kind of thinking, it's the same kind of thing you're going to do at work, maybe change implementation is a bit more complicated, especially in industry. But I think the reasoning, all the logic, all the work done to, say, it will be done here, it will impact here, I think it's the same (...)"* (SE1).

> *"I found the same... It's the same. What changes there... it's only the project, because the issue of impact, that will be different ... at least the changes we made, I think the instructor did not... he was careful not to get a request so critical that it would involve a... that would involve several classes... On the other hand, in our case, the impact was not so big... the change... but in a softhouse we get... it usually has a much bigger impact than the ones in the project, because... the harder it was for you to*

Table VI
CLOSENESS TO ACTIVITIES PERFORMED IN SOFTWARE INDUSTRY

EC1	EC2	EC3
-	Having to deal with undocumented software	Having to deal with undocumented software
-	Having to deal with OSP	-
-	Demanding more autonomy	-
-	-	Reverse engineering requirements
Application of studied content	Real application of studied context	Applicability of practices
-	Testing software developed by third parties	-
Giving continuity to an existing project	Dealing with existing software	Working with existing software

find the... to find where you were going to change inside the code, not making the change itself" (SE6).

The qualitative data in EC2 and EC3 also asserted the recognition of closeness of the requested activities to industrial activities: 'testing software developed by third parties' and 'reverse engineering requirements', respectively. In the literature, the study by Buchta *et al.* [26] emphasizes that the use of medium-sized OSPs made it possible for students to be involved with real software change requests.

Table VI lists activities highlighted by students in each study, which are close to those in software industry. Comparing the results of each study, the most common student perceptions are having to 'dealing with existing software', 'applicability of practices' and 'having to deal with undocumented software'. The remaining features happened in only one study.

For Carrington [20], the use of OSPs provides the possibility to work with reverse engineering and software maintenance, which is a more realistic experience for students than just focusing on developing a new project. As shown in Table VI, students from the three case studies highlighted the need of having to deal with existing software, unlike what usually happens in college, showing their perception that professionals are not always doing projects from scratch.

> *"(...) Because that's what's going to happen. You will not ... usually you will not catch a problem from scratch"* (SE8).
>
> *"(...) our program is very ... you develop, reinvent the wheel, you do something that already exists, instead of learning practices of how to improve, how to optimize ... this, for me, I think it is not so feasible when go to industry"* (SE6).
>
> *"(...) 'get in a company and maintain this system', I never did it. And after having that experience I saw that the things are going to be serious. It is what industry expects, then, you need to be up-to-date, not keeping this traditional method of elaborating from the beginning"* (SR1).
>
> *"And, in itself, this reverse engineering that we've seen is what happens in industry, where we're going to get products... you know... ready in the market, and we're going to be subject to maintaining what other programmers had already done, full of patches, and not well organized (...) who follow the career of developer in industry, will probably going to see this stuff a lot"* (SR1).

Another feature that confirms the closeness of students' OSP activities to those performed in industry were students reporting they were already applying the same practices in their internships (EC1 and EC3) or even as professionals (according to an ex-student – ST24 – who participated in EC2).

> *"A lot of stuff, I already applied there in the company where I work, some practices of you improving the organization of your code, use a versioning system, these practices that have ... some techniques... like it was in class... like it was in the labs ... the techniques for you to find where you will make a change in your project (...)"* (SE6).
>
> *"(...) I'm using this stuff of concept location, impact analysis, each piece of code like this (...)"* (SE2).
>
> *"(...) and this issue of documentation, so ... more in-depth ... So, I think I had a chance to actually get a tool which is what I'm really using in the internship, that's what I'm going to use when I get out of university, and the methods are methods I'm going to have to use in the... at the office"* (SR8).
>
> *"Another important issue is that after the course I started to worry about doing automated tests..."* (ST24).

Although detected in only one study, the fact that the approach required more student autonomy is quite relevant. As recalled by one student in EC2, he had to develop this behavior throughout his professional experiences. His classmates, on the other hand, even though they were not used to, had to overcome this challenge of performing activities in the project. According to Budd [27], unlike a traditional course, where all that is needed has been explained earlier or there is an explanation in some textbook, because of the diversity of tools used or skills required, students will always need to find out what they need to know and where they can get that information, regardless of the OSP adopted. Therefore, the approach requires that students develop some autonomy.

C. Difficulties arising from the use of real projects

On the other hand, closeness to reality leads to a set of difficulties that students have to face. 100% of the students in EC2 and EC3 agreed that using an OSP allowed them to visualize the complexity and difficulties of a real software project.

Table VII
DIFFICULTIES TO DEAL WITH AN OSP

EC1	EC2	EC3
Intimidating at start	Difficulty with large projects	Initial impact
	Not having experience with complex projects	
	Not knowing how to start in a large project	
Difficulty to perform activities	Difficulties to run tests	Difficulty to perform activities
-	-	Understanding some features
-	Understanding code developed by third parties	Understanding code developed by third parties
-	Lack of documentation	Lack of documentation
-	-	Novelty of different approach

Table VIII
NEEDED SKILLS PERCEIVED BY STUDENTS AFTER PARTICIPATION IN THE OSP

EC2	EC3
Reading code and documentation produced by third parties	-
Writing code and documentation understandable by third parties	Knowing how to document software
Adapting to standards used by the team	-
Knowing how to work in teams	Knowing how to work in teams
Having a theoretical knowledge base	-
Knowing the basics about existing tools	-
Knowing how to search information and to filter resulting information	-
Determining what should be tested and how it should be tested	-
Experiencing the most common errors	-
Thinking of all possibilities when testing software	-
-	Knowing how to identify the problem and what to apply to solve it
-	Knowing how to seek requirements

Table VII describes the difficulties identified by students when performing activities in EC1, EC2 and EC3. Among the commonly found difficulties, we highlight: 'initial impact', 'difficulty to carry out activities', the challenge of 'understanding code developed by third parties' and 'lack of documentation'. Particularities were only found in EC3 because of the 'novelty of the approach' (dealing with requirements from an existing project) and, consequently, the need to 'understand all the features' of the software system, which, particularly in the case of JabRef, has posed some difficulty for the features are user-configured.

The novelty of dealing with unfamiliar software of reasonable size and complexity impacts students initially, the same way as when they face similar situations in industry. A student in EC1 stated the feeling of being intimidated when faced with this type of situation. Having this experience still in college serves as learning on how to proceed and provides confidence that one will be able to accomplish what is requested. However, one needs to measure the size of the challenge to be given to students so that they do not feel discouraged and drop out, as happened in EC2 and EC3.

As for the difficulty to perform activities, this is an expected difficulty given the absence of simplifications when real software is used. Size, complexity, code structure problems, incomplete documentation, among others, pose difficulties in locating and performing change (EC1), in defining what to test and how to test (EC2), and in building diagrams that depict the application code (EC3). However, they also reflect the possible difficulties they may face in software industry.

Related work also highlights difficulties such as 'incomplete documentation' [24], [28]; 'project understanding' [24], [28]–[30] and 'difficulty to perform the requested activity' [31]. We may assert that the difficulty in 'understanding code developed by third parties' (Table VII) is one explanation for the difficulty of 'understanding the design', as well as the 'lack of documentation'. Finally, that the difficulty of understanding directly influences the 'difficulty to perform the activities'.

If, on the one hand, the evidence found shows that such difficulties appear with frequently, we point out that they are not exclusive to OSPs. They can arise regardless of the real project used, precisely because they are a consequence of the features present in this type of project (see Table V). On the other hand, Carrington [20] highlights that the difficulties vary depending on the OSP used.

D. View about needed skills

Despite the difficulties, participation in the project allowed 93% of students in EC2 and 100% of students in EC3 to better understand the skills needed by the software development professional. Table VIII details the required skills indicated by students in EC2 and EC3. It is worth noting that we did not directly investigate this issue in EC1 and no category emerged in this respect from those data.

Table VIII shows that students have realized the need for particular skills related to the area of knowledge under study, such as: 'determining what should be tested and how it should be tested', 'experiencing the most common errors' and 'think of all the possibilities when testing the software', for the EC2 study, which dealt with software testing, and 'knowing how to seek requirements', for the EC3 study, which dealt with software requirements. Students also recognized the need for general skills such as 'writing code and documentation

understandable by third parties' and 'knowing how to work in teams', recognized in both studies (EC2 and EC3) and other skills recognized in only one of the studies: 'adapting to the standards used by the team' and 'knowing how to identify the problem and what to apply to solve it', respectively, in EC2 and EC3. One observation is that students in EC2 identified more features than students in EC3. We assume this happened because of the greater maturity of the former, with students close to graduation.

E. Limitations

This work essentially presents the results of a qualitative analysis of three case studies. Therefore, we had no intention to generalize its results to other scenarios or contexts. Given the methodological rigor suggested for qualitative research and followed in this work, its results can only be extrapolated or transferred to similar conditions. We believe, however, that even particular results may be useful to researchers and practitioners. Finally, because of space constraints, we chose to present only a few excerpts from interviews to give consistency to our findings.

V. Conclusions

This study investigated students' perceptions of their interactions with an open source project as a real world experience. By performing three mixed-methods case studies in the context of three distinct classes, where students worked with an OSP to respectively learn about software maintenance and evolution, software testing and reverse engineering software requirements, we could gather students' perceptions of features of OSPs, skills developed by interacting with an OSP, and difficulties faced along the process that led us to conclude they perceived it as a real world experience.

According to Dewey [13], for education to achieve its goal, both for the individual and for society, it must be based on experiences close to real life. Experiential learning seeks to apply this strategy by immersing students in real life activities. However, to go beyond informal learning that occurs in the context of everyday life events, design of an experiential learning approach requires both selection of experiences within a pedagogical framework and reflection, critical analysis and synthesis [32].

The use of OSPs is proving itself as a relevant option to bring software project reality into the academic environment. In this research we sought to accurately capture students' reflections on this strategy throughout the performed case studies.

From our results and discussion, taking into account both the context of our case studies and the results of previous work, we verified that students perceived the activities they've performed in a OSP as a real world experience. As a consequence, they were able to overcome difficulties of working with a real project and to understand the needed skills to face future challenges in their professional practice. We believe that this is relevant for it may raise awareness and, therefore, attitude changes regarding the pursuit of those skills even during the course of their undergraduate programs.

Future work should address other possible consequences of the use of open source projects as a didactic resource for the professional training of software engineering students, as well as more in-depth studies on the difficulties faced by students to successfully participate in OSPs and the challenges faced by instructors to plan, execute and evaluate such experiences.

Acknowledgment

The authors would like to thank the both students and faculty members from the State University of Feira de Santana (UEFS) and the Federal University of Sergipe (UFS) that voluntarily took part in the OSP software engineering courses described in this paper.

References

[1] D. Parnas, "Software engineering programs are not computer science programs," *IEEE Software*, vol. 16, no. 6, pp. 19–30, 1999.

[2] B. Meyer, "Software engineering in the academy," *Computer*, vol. 34, no. 5, pp. 28–35, may 2001.

[3] A. Radermacher and G. Walia, "Gaps between industry expectations and the abilities of graduates," in *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13*. Denver, Colorado, USA: ACM Press, mar 2013, p. 525.

[4] H. Zhang and H. Su, "A Collaborative System for Software Engineering Education," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*. IEEE, jul 2007, pp. 313–318.

[5] C. Szabo, "Student Projects Are Not Throwaways: Teaching Practical Software Maintenance in a Software Engineering Course," in *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 55–60.

[6] M. Nauman and M. Uzair, "SE and CS Collaboration: Training Students for Engineering Large, Complex Systems," in *20th Conference on Software Engineering Education & Training (CSEET'07)*. IEEE, jul 2007, pp. 167–174.

[7] J. D. Tvedt, R. Tesoriero, and K. A. Gary, "The Software Factory: An Undergraduate Computer Science Curriculum," *Computer Science Education*, vol. 2, no. 1-2, pp. 91–117, 2002.

[8] D. Knudson and A. Radermacher, "Software Engineering and Project Management in CS Projects vs. "Real-world" Projects: A Case Study," in *IASTED Internation Conference on Software Engineering and Applications, SEA '09*, 2009.

[9] T. J. Reichlmayr, "Collaborating with Industry: Strategies for an Undergraduate Software Engineering Program," in *International Workshop on Summit on Software Engineering Education (SSEE'06)*. New York, NY, USA: ACM, 2006, pp. 13–16.

[10] T. Yamakami, "Re-engineering Software Education: OSS-aware Software Education in the Era of Utilizing External Resources," *International Conference on Advanced Communication Technology, ICACT*, 2012.

[11] D. M. Nascimento, C. Chavez, R. A. Bittencourt, K. Cox, T. Almeida, and W. Sampaio, "Using Open Source Projects in Software Engineering Education: A Systematic Mapping Study," in *Proceedings of the 43rd Annual Frontiers In Education Conference*, Oklahoma City, 2013.

[12] D. M. C. Nascimento, R. A. Bittencourt, and C. V. F. G. Chavez, "Open Source Projects in Software Engineering Education: A Mapping Study," *Computer Science Education*, vol. 25, pp. 67–114, 2015.

[13] J. Dewey, *Experience and Education*. Macmillan, 1938. [Online]. Available: http://ruby.fgcu.edu/courses/ndemers/colloquium/experienceducationdewey.pdf

[14] L. Uden and C. Beaumont, *Technology and Problem-Based Learning*. London: Information Science Publishing, 2006.

[15] G. R. Lefrançois, *Teorias da Aprendizagem*, 5th ed., J. F. B. Lomônaco, Ed. São Paulo: Cengage Learning, 2012.

[16] J. Moon, *A Handbook of Reflective and Experiential Learning: Theory and Practice*. London: RoutledgeFalmer, 2004.

[17] J. W. Gentry, "WHAT IS EXPERIENTIAL LEARNING?" in *Guide to Business Gaming and Experiential Learning*. Nichols Pub Co, 1990, ch. 2, p. 370.

[18] M. Svinicki and W. J. McKeachie, *McKeachie's Teaching Tips: Strategies, Research, and Theory for College and University Teachers*, 13th ed. Wadsworth Cengage Learning, 2011.

[19] H. J. C. Ellis, R. A. Morelli, T. R. de Lanerolle, J. Damon, and J. Raye, "Can Humanitarian Open-Source Software Development Draw New Students to CS?" *ACM SIGCSE Bulletin*, vol. 39, no. 1, pp. 551–555, mar 2007.

[20] D. Carrington, "Teaching Software Design with Open Source Software," in *33rd Annual Frontiers in Education Conference (FIE)*, vol. 3. Westminster, CO, USA: IEEE, 2003, pp. S1C–9–S1C–14.

[21] G. Xing, "Teaching Software Engineering Using Open Source Software," in *Proceedings of the 48th Annual Southeast Regional Conference on (ACM SE '10)*. New York, New York, USA: ACM Press, apr 2010.

[22] S. B. Merriam, *Qualitative Research: A Guide to Design and Implementation*. San Francisco: Jossey-Bass, 2009.

[23] R. Raj and F. Kazemian, "Using Open Source Software in Computer Science Courses," in *36th Annual Frontiers in Education Conference (FIE)*. IEEE, 2006, pp. 21–26.

[24] H. J. Ellis, R. A. Morelli, T. R. de Lanerolle, and G. W. Hislop, "Holistic Software Engineering Education Based on a Humanitarian Open Source Project," in *20th Conference on Software Engineering Education & Training (CSEET'07)*. Dublin: IEEE, jul 2007, pp. 327–335.

[25] J. Santore, T. Lorenzen, R. Creed, D. Murphy, and R. Orcutt, "The Software Engineering Class Builds a GUI for Subversion," *ACM SIGCSE Bulletin*, vol. 41, no. 4, pp. 82—-84, jan 2010.

[26] J. Buchta, M. Petrenko, D. Poshyvanyk, and V. Rajlich, "Teaching Evolution of Open-Source Projects in Software Engineering Courses," in *22nd IEEE International Conference on Software Maintenance*. Philadelphia: IEEE, sep 2006, pp. 136–144.

[27] T. A. Budd, "A Course in Open Source Development," in *Integrating FOSS into the Undergraduate Computing Curriculum, Free and Open Source Software (FOSS) Symposium*, Chattanooga, 2009. [Online]. Available: http://www.cs.trincoll.edu/{∼}ram/hfoss/Budd-FOSS-Course.pdf

[28] R. Marmorstein, "Open Source Contribution as an Effective Software Engineering Class Project," in *Proceedings of the 16th Annual joint Conference on Innovation and Technology in Computer Science Education (ITiCSE'11)*. New York, New York, USA: ACM Press, jun 2011, pp. 268–272.

[29] V. Smrithi Rekha, V. Adinarayanan, A. Maherchandani, and S. Aswani, "Bridging the Computer Science Skill Gap with Free and Open Source Software," in *International Conference on Engineering Education (ICEED)*, 2009, pp. 77–82.

[30] P. M. Papadopoulos, I. G. Stamelos, and A. Meiszner, "Students' Perspectives on Learning Software Engineering with Open Source Projects: Lessons Learnt After Three Years of Program Operation," in *4th International Conference on Computer Supported Education – CSEDU*, 2012, pp. 313–322.

[31] S. K. Sowe and I. Stamelos, "Involving Software Engineering Students in Open Source Software Projects: Experiences from a Pilot Study," *Journal of Information Systems Education (JISE)*, vol. 18, no. 4, pp. 425–435, 2007.

[32] "What is Experiential Education?" AEE. Association for Experiential Education, 2011. [Online]. Available: http://www.aee.org/what-is-ee