Preprint

Does FLOSS in Software Engineering Education narrow the Theory-Practice Gap? A Study Grounded on Students' Perception

Debora Maria Coelho Nascimento[1][0000−0002−0326−5261], Christina von Flach Garcia Chavez[2][0000−0001−5172−9641], and Roberto Almeida Bittencourt[3][0000−0002−8854−8956]

[1] Federal University of Sergipe, São Cristovão, Brazil
deboramcn@dcomp.ufs.br
[2] Federal University of Bahia, Salvador, Brazil
flach@ufba.br
[3] State University of Feira de Santana, Feira de Santana, Brazil
roberto@uefs.br

Abstract. Software engineering education is challenged by the need to convey practical experience in the context of a rich and large body of theoretical knowledge. This study investigates whether the use of open source projects can reduce the gap between theory and practice in undergraduate software engineering courses. Two qualitative case studies were conducted with students performing activities in an open source project, each one in a different course: software testing and software requirements. Results point out that the use of open source projects provides a concrete experience similar to industry experience, allows high cognitive engagement when performing tasks, favors understanding and content retention, and leads to the recognition of the usefulness of software engineering principles, techniques and methods.

Keywords: Education · Theory-practice gap · Qualitative studies.

1 Introduction

Free/Libre/Open Source Software has not only been instrumental for education and research in the academia, but also provides a real world object of study (software and its development) for software engineering (SE) researchers. Their use in SE education is becoming more popular, since it provides an opportunity for learning SE principles, techniques and methods and, thus, for narrowing the theory-practice gap usually present in undergraduate courses on the subject.

Software Engineering reference curricula [23] emphasize the need for professional practice and student participation in real projects. Several countries provide guidelines for Computer Science courses that recommend that curricula must leverage the coexistence between theory and practice, so that students can adapt to new situations of their training area in the future. Nonetheless,

examples and exercises presented either in SE courses or in textbooks are usually simple and easier to understand than real world problems. Such limitations sometimes make students remain unaware whether they really need to apply theory from the SE discipline.

On the other hand, the adoption of open source projects (OSP) to perform practical activities in the formal education of SE allows students to live practical experience environments close to the ones they will find in industry [2, 19]. Nandigam *et al.* [17] argue that students learning with OSP acquire a view that allows them to understand the basic principles of SE, not as merely academic knowledge, but as something useful and necessary for practice.

This work investigates whether the adoption of OSP allows students to make connections between theoretical knowledge with practical knowledge and skills. This investigation is relevant to address the lack of motivation that arises when students do not perceive the real usefulness of theoretical knowledge, which may lead students to ignore SE principles, practices and methods. We conducted two case studies in different courses of SE, in which students used an OSP to perform activities related to software testing and software requirements.

The results show that students perceived that the use of OSP: (i) provides a concrete experience equivalent to the situations that they will experience in the labor market; (ii) allows high cognitive engagement with active participation; (iii) favors understanding and retention of content; and (iv) reduces abstraction and allows the object of study to be more concrete and meaningful. Consequently, within the case studies and according to their own perceptions, students were able to connect the theory provided in an academic environment with real-world practice, recognizing the importance of SE knowledge, and therefore, narrowing the theory-practice gap in SE education.

2 Experience, engagement and content significance

Boud *et al.* [1] emphasize that *experience* is the basis and the stimulus for learning. Watching a lecture, reading a textbook, discussing certain content, performing practical activities, visiting a museum are examples of experience. Moon [15] distinguishes between external experience (what is experienced by the learner, whether an object, a concept, an image, etc.) and internal experience (what the learner recovers from his/her cognitive structure to the current learning situation; the set of previous experiences that are important for the present situation). Therefore, learning occurs by comparing external experience with current internal experience, and is the result of variations between them [15]. Jarvis [12] emphasizes that internal/previous experience is what guides how the apprentice responds to present experience. However, the teacher/educator can have influence on learning by specifying external experiences for the learner [15]. Finally, not every experience brings good results for learning [6, 8, 15]. Experience must be vivid, lively, interesting and must be connected with future experience, especially with out-of-school situations [6].

Cognitive engagement refers to the quality of the student's psychological engagement in academic tasks [5]. It comprises the need for the learner to process the content of the lesson, so that this engagement may be superficial or deep [24]. Learning is a function of the student's cognitive engagement, that is, it increases as a result of increased quality of cognitive engagement [24]. Attending a lecture, attending a demonstration, describing certain content in their own words, giving examples, solving exercises, discussing ideas on the subject, presenting their opinion using argumentation, solving real problems are examples of gradual forms of engagement. Engagement strategies should promote the manipulation of information, rather than memorization [9]. Uden and Beaumont [24] emphasize that engaging in complex cognitive activities generates useful and authentic learning. Moreover, engagement through practical activities allows the occurrence of errors and mistakes, which are also important for learning.

Significant content is an element that participates early in the learning process, with the perception or selection of the information to be processed. Only what is significant or meaningful to the learner is captured. Information needs to be perceived or selected so that learning takes place [24]. Lefrançois [13] argues that the more meaningful the content is, the more easily it will be remembered. This would be the explanation for the vulnerability of episodic memory, whereas an event with related meaning is more difficult to be forgotten. Learning is effective when the learner is able to organize information by identifying logical relationships in the content [24]. We highlight two issues with respect to the construction of meaning. First, concrete experiences are critical to meaningful learning [15, 20]. They enable the learner to better understand, analyze the importance of the studied content and evaluate consequences. Second, although the construction of meaning is individual, experience takes place within a social context and therefore, each person is strongly influenced by the social and cultural context of the learning environment [15]. Beliefs and values of the individuals are generated from the social and cultural context in which they live, and influence the interpretation of the facts and, consequently, the construction of meaning.

3 Related Work

Systematic mapping studies identified various initiatives to software engineering education with OSP [2, 19]. Most of the primary studies presented solution proposals or experience reports; other studies presented a more general view of how the approach could be incorporated into the curriculum, or how to bridge the training gaps pointed out by industry.

Recent studies have focused on capturing students' perceptions on the use of OSP in formal education [18, 22]. Nascimento *et al.* [18] investigated whether undergraduate students regarded the activities performed in open source projects as a real-world experience. The results provided evidence based on students' perceptions that OSP have a set of features similar to industrial software. They recognized the closeness of the activities to be carried out in industry and, consequently, of the typical difficulties in working with real projects, and the skills

they need to develop. Pinto *et al.* [22] conducted an exploratory investigation on students' perceptions about the need to contribute to an OSP as a mandatory activity in a software engineering course. The authors highlighted students' recognition of improving their technical skills and increasing their self-confidence. The study also highlighted the complexity and diversity of students' engagement in carrying out the activities. While these studies [18,22] present evidence of students' perception on the 'cognitive engagement' and 'vivid and real experience' provided by the adoption of OSP in formal education, our study aims to provide evidence grounded on students' perception on the 'significance' of the contents studied with the support of OSP.

Nandigam *et al.* [17] adopted a practical approach to teach a subset of basic software engineering principles by using OSP with the belief that students would perceive how such principles could be used in practice. The authors provided a detailed report on the activities performed and concluded that the expected results were achieved. However, they collected such perceptions based on students' oral presentations and follow-up discussions [17]. Our study provides evidence based on students' own perceptions about the connection between theory and practice when performing activities with the OSP.

4 Methodology

We conducted two case studies in the academic setting to explore students' perceptions on the use of OSP and whether it provides a connection between theory and practice. The first case study (CS1) was executed in a software testing course, while the second one (CS2) was executed in a course on software requirements. In both CS1 and CS2 students used `JabRef`[4], a software for managing bibliographic references in the Bibtex format, to perform practical activities. The research team was responsible for defining the activities to be carried out and supporting students with `JabRef` issues.

In CS1, students could choose to either perform or not the requested activities with `JabRef`. The instructor used classical lectures followed by exercises, with theory presented independent from the OSP. We added an optional activity to implement automated tests for `JabRef` that consisted of two steps. First, each team of students was responsible for building automated unit and integration tests for some assigned feature module. Within the team, students discussed and decided which features should be tested and who should write the tests. In the second step, they should build automated functional tests based on the application interface, analyze coverage of the implemented tests, and develop and run at least one regression test plan. In CS2, the use of `JabRef` to perform activities was compulsory. Students split into teams performed two practical assignments. In the second assignment, they should reverse-engineer the requirements of a legacy software, `JabRef`. Students identified functional and non-functional requirements, created a requirements traceability matrix, developed system se-

[4] `http://www.jabref.org/`

quence diagrams for use cases, identifying classes and methods involved; and proposed improvements to the built-in help function of the use cases.

We used interviews and questionnaires as instruments of data collection. The research team had no relation to the students or their grading. Questionnaires were used in the first contact with the students, before and after the intervention using the OSP. In the first one, the goal was to identify their previous experience with real software projects. In the second, the goal was to gather students' perceptions about the knowledge and skills regarding the subject, before applying the OSP approach. In the third questionnaire, the goals were twofold: to capture students' perceptions on the knowledge and skills acquired during intervention, and to identify whether they perceived the interaction with the OSP as a means to bridge the gap between theory and practice. We used semi-structured interviews after the intervention. In each case study, we invited at least one member from each team with different levels of previous experience with real projects. The interviews were recorded, transcribed and reviewed. A total number of 30 students participated in the case studies and answered the questionnaires provided by the research team. The characterization of the participants in each case study and other supplementary material are available at [21].

For data analysis, we used descriptive statistics for the quantitative data, and the inductive-deductive process described by Merriam [14] for the qualitative data. We used open coding 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 performing data analysis and interpretation for each study, we triangulated the results to identify intersections between studies and the particularities of each study.

5 Results

We identified two key themes related to students' perceptions on the connection between theory and practice with the use of OSP: 'the importance of practice for learning the theory' (Section 5.1) and 'the importance of theory to practice' (Section 5.2).

5.1 Importance of practice for learning the theory

Practice provides concrete examples. In the two case studies, students pointed out that practice with the OSP provided a 'concrete example' of application of the theory so that the object of study was no longer just an account of the instructor, and reduced the abstraction level by reifying the theory.

"You only catch a glimpse in class, you ... grasp the subject, study and imagine, right? ... but when you examine it in practice ... it strengthens more" (ST1).

"As I had already said, it was not just in the ... knowledge was not just in theory. It ... let's say it got out of the mind and actually happened" (SR5).

"Because what has been done (...) it moved from abstract to real" (SR2).

Practice helps the student to understand theory. Possibly by providing a concrete example, we found evidence in both CS1 and CS2 studies that practice with OSP helped students to 'understand the concepts' studied, a situation also reported by Hepting et al [10].

"(..) practice helped a lot to understand the test types, white box, black box, interface tests, so ... you take a closer look at the tests ... integration ... " (ST1).

"The tests clarify further, when you look at what you did and see how it fits into that (..) when you have a range of tests to differentiate a group of tests from another group, such as integration and unit tests. (..) When you just have an explanation, you don't ... oh, that's it, and it's a very similar thing, that has a different question. ' (..) when you start writing a test, you already ... 'oh, okay, I need to do this, so this is an integration test' (..) You are already differentiating by groups ... It kind of enriches your ... theoretical knowledge" (ST2).

"It was something that greatly improved my understanding, and I think of my friends' too, at least in my team" (SR2).

"(..) JabRef ... I think we could apply a lot of topics (..) and put into practice (...) as I said, there were unclear things, topics of the course that were not so clear. And there with the project, (...) trying to do ... to carry the work out, things got clear" (SR4).

"(...) several theoretical topics that I believed I had understood, then when I went to practice and used that understanding, I realized it was not quite that ... That eases theory understanding " (SR7).

Practice consolidate contents. In CS2, students emphasized that the use of the OSP in practical activities enabled them to 'consolidate knowledge'.

"(..) we study a lot of topics, and we understand, but ... with the project, I think it further consolidates the knowledge" (SR3).

"(..) I was able to ... connect the dots and ... fill in the gaps, let's say ... to consolidate the knowledge (..) I think it became more ... let's say so ... concrete. I could actually ... consolidate this kind of information" (SR8).

"And with practice ... it's as if it [the project] creates a box and keep it into the person's knowledge, into the mind, which ... turns it into ... something already known, it is no longer something new for you, and ... [with respect to] the knowledge of the course contents ... it has enriched a lot" (SR5).

In these last two extracts, students mention that, with the OSP, they were able to 'connect the dots and fill in the gaps' (SR8) or that the execution of the project allowed 'to create a box and keep it into the person's knowledge, into the mind' (SR5). Without realizing it, students illustrated the process of knowledge assimilation and accommodation in their cognitive structure, according to the constructivist understanding of the learning process [13].

Practice helps content retention. Students recognized the relevance of practice with the OSP for content retention. For CS1 participants, practical activities with `JabRef` were more meaningful than simply studying for an exam, and possibly forgetting the content soon afterwards: *"let's say ... it's better for retaining knowledge. Because if you give something more concrete to keep what was taught in class, it is not just for passing the exams, for example"* (ST5).

"Because if you only focus on theory (..), in a month, you will forget it, but not with practice, when you need it, it will be there" (ST3).

A student explained that retention happens because it is necessary to understand the content to be able to do the practical activity: *"It was good to remember because... you see a thousand concepts (..) and you stay there with no attention sometimes ... then you have to remember what is an unit test, what is a regression test ... I'll have to write one (..) how do you write it, you have to understand ... doing helped me remember it all"* (ST4).

Practice promotes student's active participation. In CS2, two students indicated as a relevant factor for content retention, the need for 'active participation': *"(..) when you go to class to watch ... just slides (..) that knowledge will vanish at a certain point in time. There will come a time that you will no longer remember ... what you've seen, what you've heard, if you do not consolidate through practice ... you take ... from theory and put into practice what you see, what you hear, what you're learning, and I think that contributed a lot to knowledge"* (SR5).

"This project was very important because we could do, with our own hands, what we learned in class or with third parties, you know? It's much better for you to learn by doing than by listening ... Because one thing is for you to use slides and to be there speaking, blah, blah, blah ... But it's neither our fault nor the instructor's fault. It's because the model applied in class is totally different from what you do. You learn much more by doing than by seeing" (SR2).

In traditional classroom, the student only 'receives' information and the knowledge 'will vanish at a certain point in time' (SR5). Active participation is distinct from such passive attitude and *'you learn a lot more by doing than by listening'* (SR2). Budd [3] adds that active participation in any OSP will require that students become self-taught because they will be continuously challenged to learn some tool or develop some skill.

Practice confirms the applicability of theory. In CS2, students were able to perceive that the concepts and principles studied were actually applied in practice, that is, to confirm the 'applicability of theory': *"It contributed because we really did practice. We set out to practice and realized that they really are applied ... the whole concept, the whole theory that was presented in class"* (SR1).

"At least I did not have that skill. (...) we had to run after stuff we didn't know and put into practice in the project" (SR2).

"Practice helped a lot to figure out how the diagrams and stuff works." (SR7)

Practice is essential to learning some subjects. In CS1, students recognized that 'lack of practice leads to partial learning'. Lectures enabled understanding about tests, but doubts emerged with practical activities: *"You under-*

stand visually, but when you put into practice what was presented in class, then the doubts begin to arise" (ST7).

To 'learn how to do', one needs to exercise the testing techniques: *"I think a unit testing without practice for me would be ... wouldn't be good, I think ... I would learn half of it ..."* (ST1). The learner will only be sure if he or she knows or does not know, after experiencing [8].

Practice helps students to grasp the problem. By working with projects that are close to reality, students could 'see how the problem really is' and confirm that things are not as simple as they seem, when they consider only theory. According to Morelli *et al* [16], students witness that challenging problems rarely yield to solutions presented in textbooks. Students who do not have experience with real projects have unrealistic expectations regarding the quality of the source code and are surprised when they do not find an elegant code like those presented in textbooks.

"(..) you only see the problem when you have the problem at hand. Knowing the theory on how to do the tests, what is needed, is nice, but you will only really know what is needed, when you are there in the situation (..). So, I think theory is good, but practice makes it worthwhile" (ST2).

"(..) because in theory everything is simpler ... The examples we get are always simple ... it's an integer, then a number between zero and ten. Now this is not a number [commenting on the project], it's a string, there's a database that, with an input, with something else as input, so it stays ... it was difficult to know ... what exactly to use from theory" (ST6).

"(..) we see how complicated it is (..) eliciting requirements ... knowing the use cases and everything ... knowing a little bit of what the guy who developed had to know ... It's pretty cool, because sometimes we see the theory, but with practice, it's when you see that, it's more difficult than in theory ... it's something that we get the content and think we know, but when you see it in practice, there the doubt arises ... it helped a lot. " (SR3)

Practice enables discussion. Concrete examples, a more realistic view of problems to be faced, judgments of which techniques should be applied, and the experience with a common project 'enable discussion'. From the practical project, students have their own experience and elaborate their point of view. In addition, all students experience a little of the project, different perspectives can be exposed, favoring discussions that take place on concrete examples. In CS1 and CS2, specific classes were devoted to discuss the activities carried out in the project, and students reported their satisfaction with the learning in the light of such discussions: *"I think it contributed a lot, the issue of different perspectives regarding the project ..."* (ST8).

"... you do, implement things, see results and have your opinion. When you see that from another person as well, you increase ... you see more things too, that you have not seen and that other person has" (ST1).

A student reported that discussions in class allowed him to compare his solution with those of his classmates: *"And we can also check whether what we've done is right (..) With discussion, listening to other people's opinions is*

also interesting because sometimes they point out things that we haven't perceived, and that should be there ... I found it quite interesting" (SR3).

5.2 Importance of theory to practice

Theory is necessary for practice. In CS1 and CS2, this category emerged from several excerpts: *"You can't do testing without knowing the theory ... Even if you haven't learned it previously in the course, you come back, because you have to see the strategies, study and implement them (..) practice without theory does not exist ... for me, it's important"* (ST1).

"When I actually moved to practice, that demanded me to come back to theory to get more knowledge about that subject ..." (ST3).

"Things I've seen in class ... a few, I managed to put them in the project. Other stuff, I had to get from third parties, in websites and articles, so I could be able to do them in the project. I had to complement [my knowledge] about this, so I could implement it in the project " (SR2).

Theory enables planning the things to be performed. Theory is important to practice because it allows 'planning practice' (CS1) and defines 'what' and 'how': *"I had to come back to what I've seen in class, to know what had to be done: oh, I have to do such thing with such a JabRef class ... How does that class access that method? How do I test it?"* (ST4).

"(..) when you write the tests... sometimes you don't know exactly what you're doing there... When you have the theoretical background... you need to plan to do the tests and then you already have a sense of what you're doing, so it helps a lot, you get an idea of how you're testing" (ST2).

In this last extract, the student emphasized that without theory, he tests with no clear purpose. But when he knows the theory, he knows how to plan what should be tested and how.

5.3 Discussion

Practice with an OSP lets you learn how to do, which leads to the development of technical skills such as testing or systematically changing software. Dealing with a real problem can develop pro-activity and creativity in the search for alternative solutions. The real-world context allows students to see problems as they really are and enables them to perceive the applicability of the theory, promoting reflection and development of critical thinking. Practice with the OSP enables discussions about the project which allow students to express their opinions and broaden their views from their classmates' opinions. Thus, several skills linked to professional practice can be developed. Students also recognize that theory is necessary for practice and enables planning things to be performed.

Other studies corroborate our results. In the study by Chen *et al.* [4], one of the students stated that participation in the project allowed putting into practice the concepts learned in class. Hislop *et al.* [11] conclude that after practice with OSP, students can see the reasons for applying the theory. Ellis *et al.* [7] complement that principles of SE incorporated into the project become evident to

students as the project is executed, that is, principles are learned by experience and not by the instructor's voice. For Nandigam *et al.* [17], activities with OSP provide a solid background for the discipline of SE, and the lectures and discussions on design metrics, code maintainability, documentation and concern with design-code synchronization, with the articulation of their own points of view, have become more significant for students after the activities in the project.

On the other hand, our results cannot be generalized to other contexts. Given the methodological rigor suggested for qualitative research and followed in this work, CS1 and CS2 results can only be extrapolated to similar conditions. We believe, however, that even particular results may be useful to researchers and practitioners. Finally, because of space constraints, we presented only a few excerpts from interviews to give consistency to our findings.

6 Conclusions

This study investigated whether the adoption of open source projects in SE education enables students to to connect software engineering theory with practice, under students' own perceptions. Three elements from learning theories provided the background for our study: the experience lived by the student, the depth of the student's cognitive engagement throughout this experience, and the significance of the contents studied.

We reported the results of two case studies conducted in two different SE courses, in which students used `JabRef` to perform activities related to software testing and reverse engineering of requirements. These studies brought grounded evidence to support that the adoption of OSP in practical activities in formal SE education (i) stands for a concrete experience equivalent to industry-like situations to be later experienced by graduates; (ii) allows high cognitive engagement with active student participation while analyzing, testing, modifying, and documenting the source code, among other activities that can be performed; (iii) favors content understanding and retention; and, finally, (iv) leads to the recognition of the importance of principles of software engineering, that is, the construction of meanings that allow more effective learning.

In addition, our study revealed that the OSP approach allowed students to experience that reality is more complex than how theory is usually presented; it promoted the development of technical skills; it supported discussions based on each student's practical experience, and promoted the development of critical thinking, broadening their view to other possibilities. It is worth highlighting the evidence of a student describing the process of assimilation and accommodation of knowledge supported by constructivist theories for the learning process.

From the point of view of our initial research question, we conclude that within the two case studies that used `JabRef`, the object of study became less dependent on instructors' accounts and less abstract, and became something real, concrete and meaningful. Therefore, in that context, the use of OSP through practice, supported students to make connections between theory and practice, narrowing the gap between them.

References

1. Boud, D., Cohen, R., Walker, D.: Understanding Learning from Experience. In: Boud, D., Cohen, R., Walker, D. (eds.) Using Experience for Learning, pp. 1–18. Open University Press (1993)
2. Brito, M.S., Silva, F.G., Nascimento, D.M.C., Chavez, C.F.G., Bittencourt, R.A.: FLOSS in Software Engineering Education: An Update of a Systematic Mapping Study. In: Proc. 32nd Brazilian Symposium on Software Engineering (SBES). pp. 250–259. Sao Carlos, Brasil (2018)
3. Budd, T.A.: A Course in Open Source Development. In: Integrating FOSS into the Undergraduate Computing Curriculum, Free and Open Source Software (FOSS) Symposium. Chattanooga (2009), `http://www.cs.trincoll.edu/{~}ram/hfoss/Budd-FOSS-Course.pdf`
4. Chen, Y., Roytman, A., Fong, P., Hong, J., Garcia, D., Poll, D.: 200 Students Can't Be Wrong! GamesCrafters, a Computational Game Theory Undergraduate Research and Development Group. In: AAAI Spring Symp. - Tech. Report (2008)
5. Davis, H.A., Summers, J.J., Miller, L.M.: An Interpersonal Approach to Classroom Management: Strategies for Improving Student Engagement. Corwin (2012)
6. Dewey, J.: Experience and Education. Macmillan (1938), `http://ruby.fgcu.edu/courses/ndemers/colloquium/experienceducationdewey.pdf`
7. Ellis, H.J., Morelli, R.A., de Lanerolle, T.R., Hislop, G.W.: Holistic Software Engineering Education Based on a Humanitarian Open Source Project. In: 20th Conference on Software Engineering Education & Training (CSEET'07). pp. 327–335. IEEE, Dublin (jul 2007). https://doi.org/10.1109/CSEET.2007.26
8. Gentry, J.W.: What is Experiential Learning? In: Guide to Business Gaming and Experiential Learning, chap. 2, p. 370. Nichols Pub Co (1990)
9. Hannafin, M.J.: Instructional strategies and emerging instructional technologies: Psychological perspectives. Canadian Journal of Educational Communication **18**, 167–179 (1989)
10. Hepting, D.H., Peng, L., Maciag, T.J., Gerhard, D., Maguire, B.: Creating Synergy Between Usability Courses and Open Source Software Projects. ACM SIGCSE Bulletin **40**(2), 120–123 (jun 2008). https://doi.org/10.1145/1383602.1383649
11. Hislop, G.W., Ellis, H.J., Morelli, R.A.: Evaluating Student Experiences in Developing Software for Humanity. ACM SIGCSE Bulletin **41**(3), 263–267 (aug 2009). https://doi.org/10.1145/1595496.1562959
12. Jarvis, P.: The Paradoxes of Learning. Jossey-Bass, San Francisco: (1992)
13. Lefrançois, G.R.: Teorias da Aprendizagem. Cengage Learning, São Paulo, 5 ed. edn. (2012)
14. Merriam, S.B.: Qualitative Research: A Guide to Design and Implementation. Jossey-Bass, San Francisco (2009)
15. Moon, J.: A Handbook of Reflective and Experiential Learning: Theory and Practice. RoutledgeFalmer, London (2004)
16. Morelli, R., Tucker, A., Danner, N., De Lanerolle, T.R., Ellis, H.J.C., Izmirli, O., Krizanc, D., Parker, G.: Revitalizing Computing Education through Free and Open Source Software for Humanity. Communications of the ACM **52**(8), 67–75 (aug 2009). https://doi.org/10.1145/1536616.1536635
17. Nandigam, J., Gudivada, V.N., Hamou-Lhadj, A.: Learning Software Engineering Principles Using Open Source Software. In: Proc 38th Annual Frontiers in Education Conference (FIE). pp. 18–23. IEEE (oct 2008). https://doi.org/10.1109/FIE.2008.4720643

18. Nascimento, D.M.C., Chavez, C.F., Bittencourt, R.A.: The Adoption of Open Source Projects in Engineering Education: A Real Software Development Experience. In: Proc. 48rd Annual Frontiers In Education Conference (FIE). pp. 1091–1100. San Jose (2018)
19. Nascimento, D.M.C., Bittencourt, R.A., Chavez, C.F.: Open Source Projects in Software Engineering Education: A Mapping Study. Computer Science Education **25**, 67–114 (2015)
20. Experiential Learning (2012), `http://www.niu.edu/facdev/resources/guide`
21. Does FLOSS in SEE narrow the Theory-Practice Gap? Supplementary Material (2019), `http://sites.google.com/site/oss2019osp/`
22. Pinto, G., Ferreira, C., Souza, C., Steinmacher, I., Meirelles, P.: Training Software Engineers using Open-Source Software: The Students' Perspective. In: Proc. of the International Conference on Software Engineering, Software Engineering Education and Training (SEET). Montreal, Canada (2019)
23. Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (2015)
24. Uden, L., Beaumont, C.: Technology and Problem-Based Learning. Information Science Publishing, London (2006)