FLOSS in Software Engineering Education

Supporting the Instructor in the Quest for Providing Real Experience for Students

Fernanda Gomes Silva
Federal University of Bahia
Salvador, Brazil
fernanda_gomes@unit.br

Moara Sousa Brito
Federal University of Bahia
Salvador, Brazil
moara.brito@ufba.br

Jenifer Vieira Toledo Tavares
Federal University of Bahia
Salvador, Brazil
jenifer_vieira@unit.br

Christina von Flach G. Chavez
Federal University of Bahia
Salvador, Brazil
flach@ufba.br

ABSTRACT

Software engineering courses play an important role in computer science programs and are expected to provide the required basic knowledge and skills for professional practice in software industry. However, teaching software engineering principles, concepts and practices, and relating them to real-world scenarios are challenging tasks. The adoption of open source software projects may address such challenges. In this paper we report on an experience of the teaching object-oriented modeling with Unified Modeling Language (UML) Class Diagrams using open source projects. We conducted a case study with students of the software engineering discipline of the Computer Science course. We supported the instructor in some activities related to syllabus planning, including the selection of a Free/Libre/Open Source Software (FLOSS) projects and the creation of examples to be used in the classroom. The instructor selected and used a small FLOSS project to support the modeling activities. Then, the instructor applied an evaluation activity and a perception questionnaire about the methodology used. After the end of the classes, we conducted an interview with the instructor to present a brief report of his experience in the classroom. In the perception of students, the experience with FLOSS projects enhanced their ability to handle real projects and third-party code, and to deal with the job market. They also reported developing skills such as proactivity and communication. From the instructor's perspective, the group was enthusiastic and dynamic, and interacted more during practical activities.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Software and its engineering** → **Open source model**; • **General and reference** → *Surveys and overviews*;

SBES 2019, September 23–27, 2019, Salvador, Brazil

ACM ISBN 978-1-4503-7651-8/19/09…$15.00
https://doi.org/10.1145/3350768.3353815

KEYWORDS

Free/Libre/Open Source Software, FLOSS, Class Diagram, Experience report.

ACM Reference Format:
Fernanda Gomes Silva, Moara Sousa Brito, Jenifer Vieira Toledo Tavares, and Christina von Flach G. Chavez. 2019. FLOSS in Software Engineering Education: Supporting the Instructor in the Quest for Providing Real Experience for Students. In *XXXII BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING (SBES 2019), September 23–27, 2019, Salvador, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3350768.3353815

1 INTRODUCTION

Proper education and training can significantly improve the software engineering practice and are considered prerequisites for advancing the state of the art in the software industry [16]. Yet, there are several challenges related to teaching and learning Software Engineering (SE) because of the varied methodologies, concepts, technologies, tools and the need to teach students how to deal with existing and complex software systems [11, 22, 25].

Software Engineering Education (SEE) should promote the training of software engineers with the ability to contribute to the development and evolution of real systems [7]. Curriculum guidelines for software engineering, computer science and related courses [8, 9, 29] 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. Moreover, the educator should not only provide a solid theoretical basis in SE, but also ensure contact with problem solving, communication skills and teamwork to prepare students for career [12]. Last but not least, in order to improve the effectiveness of learning it is necessary to employ methods in which the students actively take part in the learning process.

However, instructors may face several difficulties to address these requirements. For instance, incorporating realistic practices into formal education is a challenging task for the instructor, in face of the need to provide a mapping between pedagogical objectives and the actual practical experience, and to cover course contents during the period of the school year [24]. Moreover, to support the alignment of SE theory and practice, instructors should consider skills and attitudes that go beyond concepts, methods

and techniques, considering the current software development scenario, the complexity of social interactions, and how development of collaborative software occurs in a real-world environment [19].

It is not easy to use projects developed in the industry in the classroom or to interact with companies in the educational context [11]. Free/Libre/Open Source Software (FLOSS) projects can be a viable solution to introduce realistic practices in the learning and teaching process of Software Engineering [19]. The use of FLOSS projects allows the approximation between the teaching-learning process of higher education and the construction of software in the real world, having characteristics similar to those found in industry and hardly available in academia, such as: evolving medium and large software, with many contributors performing development, bug fixing, testing, and documentation tasks. In addition, its lifecycle goes beyond a semester's duration and the FLOSS project can be used in other semesters or other disciplines, allowing students to experience third party software [21]. This approach is considered as an experiential learning strategy and an opportunity to provide students with a vivid experience, connected with possible future experiences [19]. Since this is an authentic environment where real software is being produced, faculty can usually cover most Software Engineering knowledge areas [29].

Participation in FLOSS projects allows students to interact with real systems, real problems and real software development teams interested in building high-quality products [22]. In this learning process, students usually behave actively, increasing the probability of mastering content knowledge and making the use of free software projects as a study object a good initiative to learn Software Engineering [20].

This paper reports the experience of using FLOSS projects as an object of study in the teaching of UML class diagrams for students of the software engineering discipline. In our case study, we selected the content related to UML class diagrams because of the need to train professionals capable of modeling complex systems, so they need to have contact with real world projects.

UML [3] models should be developed so that software engineers can understand and deal, at a higher level of abstraction, with the magnitude and complexity of the software to be developed and evolved [23]. UML class diagrams are useful for whiteboarding and sharing ideas in a common language, therefore supporting comprehension, communication and development of clear, concise and consistent software artifacts [29].

We also present in detail the preparation of the material to carry out the activities in the classroom, and a brief report about the instructor's experience. In addition to using the FLOSS project in the classroom, the students were submitted to an evaluation activity and were asked about their perception about the software used in their design and modeling tasks. We interviewed the instructor in order to collect and report her experience with using a FLOSS project hosted in repository for teaching SE, with diagrams of the UML and with the execution of the case study proposed.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 provides an overview of the methodology used, describing the process of selecting the FLOSS project, planning classroom activities, evaluating the content and perception of the instructor and students involved in the project. Section 4 discusses the results identified in the case study implemented. Section 5 presents the conclusion of this work.

2 RELATED WORK

The research work Ho-Quang [13] have investigated the relevance of UML models in open source project, while others have investigated different strategies to teach UML modeling [26], the student's perspective on use of FLOSS in SEE [19] and the instructor's perspective on use of FLOSS in SEE [22]. Carrington and Kim [5], Krutz et al. [15] and Sowe and Stamelos [30], used FLOSS projects in teaching software testing.

Ho-Quang and colleagues [13] highlighted the importance of UML for real-world open source projects. They performed a survey among open source developers, with focus on projects that used the UML for software modeling. Since the use of UML in FLOSS projects is not well known, they sought to find out how UML is used and considered useful. The study reports that UML really helps new contributors and its use in open source projects is partially similar to industrial use.

Silva *et al.* [26] investigated two different teaching methods (Problem Based Learning and Learning from Erroneous Examples) with respect to the degree of correctness and completeness of UML diagrams produced by students, and analyzed the students' perceptions about each method. Students have presented difficulties while creating the UML diagrams, such as: difficulty in understanding the diagram's syntax and semantics, difficulty in organizing information on the diagrams, difficulty in correctly using association of the type generalization-specialization, among others [26]. However, they did not discuss the instructor's perception on benefits and difficulties faced while preparing activities and examples to be used in the classroom.

Sowe and Stamelos [30] discusses FLOSS projects as bazaars of learning that offer a meaningful learning context. Carrington and Kim [5] used examples of FLOSS projects in an introductory course on software design and testing. The goal was to expose students to realistic software systems and give them experience dealing with large quantities of code written by other people.

Nascimento et al. [19], investigated SE students' perceptions of their contact with open source projects as a real-world experience. They conducted three case studies of mixed methods with three different undergraduate classes. The first group of students focused on maintenance and evolution of software, the second class with software testing and the third group with reverse engineering of software requirements.

Pinto *et al.* [22], with the goal of understanding benefits, challenges, and opportunities of the transition from traditional courses to novel SE initiatives, such as courses that use FLOSS, interviewed seven software engineering instructors that changed their academic setting so that students performed comprehension and maintenance tasks on FLOSS as part of their SE course. We highlight, among their findings, that (i) there are different ways to make use of FLOSS projects in SE courses in terms of project selection, assessment, and learning goals, and (ii) there is evidence on clear benefits of such approach, including improving students' social and technical skills.

The studies presented in this section are mainly related to the use of UML and experience reports using FLOSS projects in teaching

various areas of Software Engineering. To the best of our knowledge, our work is unique with the focus on using FLOSS projects in teaching the UML class diagrams, detailing the methodology used through worked example, presenting the procedures performed in the configuration of the environment and reporting the instructor's and students's perception about the methodology used.

Therefore, we report that this paper has the potential to provide a broad understanding of the use of FLOSS projects in the teaching of Software Engineering, to assist in the replicability of the case study and to collaborate with the improvement of research in Engineering Education Software.

3 METHODOLOGY

In this section we present the overall study design (Section 3.1). Moreover, we present the study execution, that comprises part of the syllabus design performed by instructor with the support of the research team (Section 3.2) and procedures for data collection and analysis (Section 3.3).

3.1 Study Design

We conducted an exploratory study in the academic setting with the goal of identifying the challenges faced by the instructor in the use of realistic projects in the classroom.

The main question that drove our exploratory study was

> *How to teach, identify and apply the concepts for modeling the UML class diagrams with a real-world open source software project?*

The study was executed in a software engineering course in the context of teaching and learning UML class diagrams. The research team was responsible for supporting the instructor during syllabus planning with respect to practical activities related to UML. There was no interaction between the research team and the students. With this choice, the research team was able to focus on the instructor and her perceptions, and collect the necessary data for the study.

To perform syllabus planning that covered UML class diagrams, we provided support to the instructor for the following activities: selection of FLOSS project to be used in the classroom (Section 3.1.1) and creation of useful examples based on the selected FLOSS project (Section 3.1.2).

The research team was also responsible for the preparation and application of an interview with the instructor, and a questionnaire to grasp the students' perception (Section 3.1.3).

3.1.1 Project Selection Process. The use of FLOSS projects for educational purposes requires the selection of uniform designs in terms of size and complexity [28]. We decided to focus our search on projects that are stored in software repositories, namely those hosted on GitHub [1]. GitHub has grown rapidly and is one of the largest open source projects [6], with a variety of complex designs and systems simple [31].

Due to the diversity of projects available in GitHub, we adopted and recommended project selection based on the levels of instructor control over the student's activities in the project, as pointed out by Nascimento et al. [18]. For this case study, we recommended the "Inside Control" strategy as the defined level of control over the activities to be developed, which includes branching the FLOSS code, preparing the activities/examples and assessing locally the students' contribution. Moreover, we recommended four selection criteria:

Programming language – the predominant programming language used in the FLOSS project as reported by Ellis [10];

Project size – the number of lines of code or number of classes in the project, etc. According to Smith [28], the use of large or complex projects may hinder student understanding and the use of small or simple projects may not demonstrate the need to use the principles of SE;

Maturity – the number of releases of a project. Projects with a large amount of version history and release may indicate that the tasks under progress are complex and require more knowledge from the students. Jaccheri [14] argues that selecting an immature FLOSS project may have the disadvantage of being significantly different from an ideally executed FLOSS project;

Domain – the project focus based on its description. According to Buchta [4] the software domain should be intuitive and easy to understand in order to relieve the student of having to learn many domain-specific concepts.

3.1.2 Creation of UML Worked Examples. A worked example typically consists of a problem formulation, solution steps, and the final answer itself [1]. In this proposed approach, the example is compared to a similar problem for students completing the other solutions [27].

In our study, we used a template for guiding the instructor in creating examples for teaching UML class diagrams to be used during lectures and related practical activities given to the students.

To solve the problem formulated in our worked example it was necessary: (i) to analyze the artifacts and the source code of the FLOSS project; (ii) identify the requirements and business rules to be modeled; (iii) map the artifacts and source code considered appropriate for the students profile; (iv) define the correct syntax and semantics of each UML graphical element; (v) select a tool to be used for modeling the diagrams.

Therefore, the proposed example describes a recipe for the instructor to identify teaching opportunities of the UML class diagrams using a FLOSS project selected in GitHub.

Our guideline is defined in the following steps:

Project Comprehension: (i) create a template with questions that will describe the project; (ii) identify tasks and technical issues to be answered; (iii) speculate a selected FLOSS artifact; (iv) speculate the source code, checking if it contains instructions in one of the existing programming languages.

Implement an initial hypothesis: (i) develop a class diagram that serves as an initial hypothesis, relating the source code and other artifacts; (ii) classify the diagram name and later its components (class, class attributes, attribute types, operations, associations); (iii) create the UML diagram manually of with the support of a modeling tool; (iv) Speculate about the UML class diagram with respect to their correctness and consistency.

[1] Accessible in: https://github.com

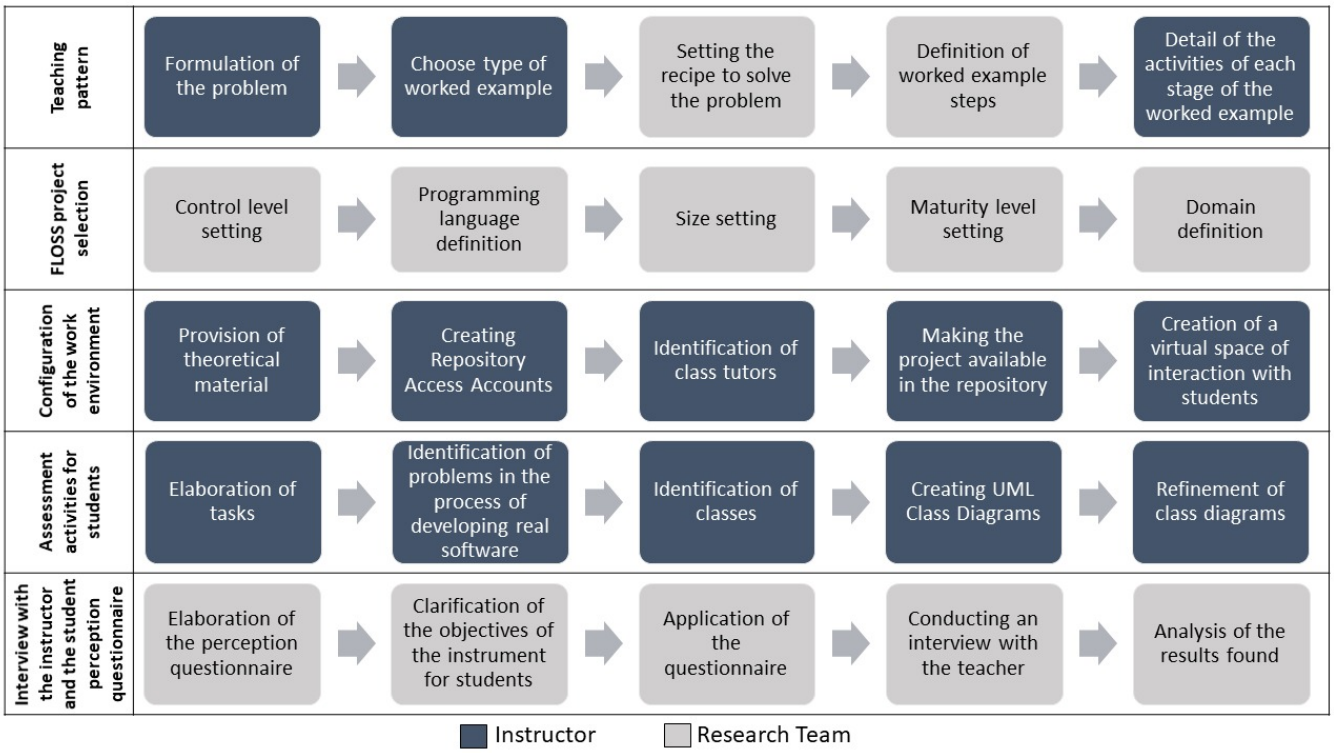

Figure 1: Methodology: Syllabus Planning

Define relationships: (i) relate the created class diagram to source code and project artifacts; (ii) check the traceability between them; (iii) present the incompatibilities found so that the students may have contact with real problems of the process of analysis and development of systems.

Adapt the class diagram: (i) rename the diagram; (ii) remodel the diagram; (iii) extend the diagram; (iv) look for alternatives to adapt the diagram.

Repeat the aforementioned steps until the class diagram is satisfactory for use as an example of the activity to be developed in the classroom.

3.1.3 Assessment for the Teaching and Learning Process. The research team developed questionnaires for application after the creation of the diagrams by the instructor, and after the activities carried out with the students, in order to grasp perceptions on:

(i) the methodology used as a real software development experience; (ii) the activities carried out, as enablers to make the connection between theory and practice; (iii) the methodology used, and if it contributed to the student feeling more prepared for the job market; and (iv) the methodology used, and if it contributed to the student's awareness about the importance of some aspects of software engineering.

The questionnaire, adapted from [21] is presented in the appendix A. Such a questionnaire consists of statements, and the student must indicate through the Likert scale the level of agreement of that statement. We use four points on the scale: "strongly disagree," "disagree," "agree," and "strongly agree".

We clarified for students the objectives of the questionnaire, and we try to raise awareness students of the relevance of their participation and frankness in their answers for validation of the contribution or not of the use of FLOSS projects in education in software engineering.

Finally, we conducted an interview to grasp the instructor' perceptions on: (i) her experience with Free Software hosted in repositories; (ii) her experience with teaching of Software Engineering; (iii) her experience with teaching the UML class diagrams; (iv) a brief opinion report about her experience in the classroom with the proposed intervention.

3.2 Syllabus Planning

For the programmatic planning of the teaching and learning process, the instructor appropriated the subject's syllabus (Section 3.2.1); of the FLOSS Project selection process indicated by the research team (Section 3.2.2);following the creation of the UML example to be followed by the students (Section 3.2.3); configured the work environment (Section 3.2.4); finalizing with the elaboration of the evaluation activity (Section 3.2.5).

Figure 1 presents the steps to define the teaching pattern, FLOSS project selection process, work environment configuration, students' evaluation activity, preparation and application of interview with the instructor and the student perception questionaire. In

the description of each step we indicate if the responsibility of execution was of the instructor or research team.

3.2.1 Teaching and Learning Process Planning. Initially the instructor prepared the lesson planning considering the teaching plan of the Software Engineering discipline. In this teaching plan, the student profile is defined based on training references that consider: (i) the ability to act in a globalized world of work; (ii) the ability to determine requirements, develop, evolve and manage the Information Systems of organizations; (iii) the ability to recognize the importance of computational thinking in everyday life, as well as its application in other domains [8, 9].

The introductory concepts on Software Engineering and Information Systems taught in the lectures were: requirements elicitation techniques, requirements types, requirements documentation, requirements traceability, introduction to the Unified Modeling Language (UML), modeling and specification of use cases and class diagrams.

In order to refine the theoretical concepts studied about UML class diagrams, in the course planning, it was considered the use of FLOSS projects as object of study for a class of students of the Software Engineering discipline.

3.2.2 FLOSS Project Selection. The instructor adopted the "Inside Control" strategy as recommended by the research team. This strategy allows that the branching of the FLOSS code available in the repository is done and that the instructor defines the activities to be carried out in the classroom and evaluates them, and therefore, the interaction with the community is not obligatory. Moreover, the instructor used four criteria suggested by the research team to support project selection:

Programming language. With respect to the programming language, the instructor chose JAVA language due to students's familiarity/pre-requisite.

Project size. With respect to the project size, the number of classes was taken into account. The instructor defined that the FLOSS project should have at least 10 and at most 20 classes.

Maturity. Since the instructor wanted a project still at an early stage of development, she searched on projects that had at most three releases.

Domain. Considering that the focus/domain of the project may have an impact on student learning, the instructor searched on projects related to Games, because, in her perception, projects that implemented Games would be attractive for most of the students.

After providing values to the criteria to be used in the selection process, the instructor performed a manual search using the GitHub platform support. First, JAVA projects in the Games domain were search for, and among the projects returned we checked the ones that matched the number of releases and the number of classes defined. Finally, the instructor selected the "Champs of the Galaxy" project [2] to be used in the classroom.

3.2.3 Creation of UML Worked Examples. The instructor followed the steps presented by the research team (3.1.2) to create examples for working in the UML.

In the end, it was possible to obtain the following structure of worked example:

PROJECT DESCRIPTION

Name of the Project;
Website;
Description.

INITIAL RECOGNITION

How long has the project been in existence?
Briefly describe the project history;
What is the project license?
What is the size of the project? Number of lines of code?
Number of classes?
Has the size of the project increased in the latest versions?
Is there any recent activity in the project?
Is there business collaboration for the project?
Are there more independent developers than company employees?
What are the technologies (language, libraries, databases) used by the software?

TECHNICAL CHARACTERIZATION

How many contributors participated in the project in the last 6 months?
And from the beginning?
What is the release policy?
Is there documentation to help newcomers?
What kind of information is available?
Does the project have an automated test suite?
Does the code contain comments?
Are there parts that need more comments?
Are there parts of the source code that are problematic? Justify.
Can design patterns and architectural patterns be identified from code?
Which ones?
Is there any UML diagram saved in project folder(s)?
In which?
What did you need to install on your system to make the project compile/run?
Is the documentation provided by the project sufficient?

TASKS ON UML CLASS DIAGRAMS

1st STAGE:

Now that the pair already know the Champs Project of the Galaxy, you should work on creating UML class diagrams, according to the concepts presented in the classroom and recommended readings. To support the creation of class diagrams, I suggest the use of an open source modeling tool, e.g. ArgoUML[3] or StarUML[4]. Both students should create the diagrams individually and then discuss each one design decisions. After creating the final diagram, post the image of it by updating the 'Branch' created with a new 'Pull Request'. Remember that commits will be evaluated, so use resources, such as the checklist of both members of the pair to create the class diagrams.

[2] Accessible in: $https://github.com/rodriguesfas/Champs_daGalaxia$

[3] http://argouml.tigris.org

[4] https://sourceforge.net/projects/staruml/

2nd STAGE:
Creating the UML Class Diagram.

He then guided the students, describing step by step how best to solve the problem. The resulting example is shown in Figure 2 and the guidelines for learners can be observed in the Google Classroom tool[5].

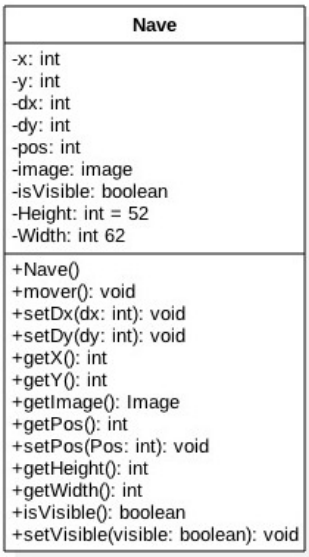

Figure 2: UML Class Diagram: Resulting Example

3.2.4 Setting up the Environment. The teaching material on UML class diagrams was made available in the Google Classroom tool for students. In the classroom, students were instructed by the instructor to create accounts on GitHub. Next, the instructor identified the students with experience of use in the repository to be tutors. The instructor created a discussion forum in Google Classroom for students to post supplementary materials regarding the use of GitHub.

The instructor created a "Software Engineering organization" in GitHub and students were included as collaborating members. For these tasks, students were organized in two-person teams. The following instructions were made available: (i) fork the selected FLOSS project; (ii) create a branch in your own project; (iii) name the branch using the description $Pair - Student01 - Student02$; (iv) create a folder with the name of the branch; (v) individually fill in the selected FLOSS project recognition template (see (vi) make a pull request of the branch created for branch master of the FLOSS project available in the organization.

3.2.5 Elaboration of the Evaluation Activity. The instructor designed an evaluative activity considering the cognitive mastery and skills developed by the students, including intellectual development, knowledge acquisition and the ability to recognize information, patterns and facts [2]. The cognitive domain is segmented into six levels, namely [2]: (i) knowledge: the student must be able to memorize ideas and concepts, and can reproduce them, even if in context other than the original; (ii) understanding: the student must be able to interpret the information received, associating it with his prior knowledge of the subject; (iii) application: the student must use his knowledge in solving a problem; (iv) analysis: the student must be able to analyze, compare and classify information and hypotheses in search of a solution to a given problem; (v) synthesis: the student must be able to synthesize and combine knowledge to reach a resolution; and (vi) evaluation: the student should be able to evaluate and critique information based on established criteria.

In the proposed evaluative activity, the student should be able to know and understand the concepts, apply, analyze and synthesize in the creation of the class diagram and evaluate the generated model to be able to refine the components of the diagram. Therefore, the proposed evaluative activity falls within the 6th level of the Bloom Taxonomy [2].

3.3 Data Collection and Analysis

We used interviews and questionnaires as instruments of data collection. The research team had no relation to the students or their grading. Questionnaires were applied by the instructor with the students after the activities performed using the FLOSS. We used semi-structured interviews with the instructor after lesson planning and after the execution of the activities related both to the use of the FLOSS and UML class diagrams.

For data analysis, we used descriptive statistics for the quantitative data, and discursive analysis for qualitative data.

4 RESULTS AND DISCUSSION

4.1 The Students' Perceptions

The questionnaire was applied to twenty-one students enrolled in the course, but only eighteen students answered the questionnaire.

With respect to the distribution of students per course period, 61.6% of the students were allocated in the 4th and 5th periods of the Computer Science course.

With respect to professional practice, 88.89% of the students declared to not have worked professionally in IT jobs, while the remaining 11.11% of the students declared to work as programmers. Figure 3 presents the insight from students, that have a job and that do not have a job in IT, about their technical ability in UML class diagrams and their readiness for the job market, after contact with the FLOSS project in the classroom.

It can be seen that there was no significant difference in student responses regardless of their professional experience.

With respect to students experience in real projects, 77.78% of the students declared to have no previous experience with real projects, that is, the majority had their first experience with a real project in the classroom.

The students evaluated their own ability to describe concepts and practices related to UML class diagrams, as well as to analyze and generate the diagrams using tools.

Most students stated that they were able to describe concepts and practices related to UML class diagrams, analyze and generate diagrams for midsize software. It is worth mentioning that 100% of students consider themselves capable of using tools to create class diagrams. The students also evaluated whether the methodology

[5]Accessible in: https://sites.google.com/view/esdiagramauml/diagrama-de-classes

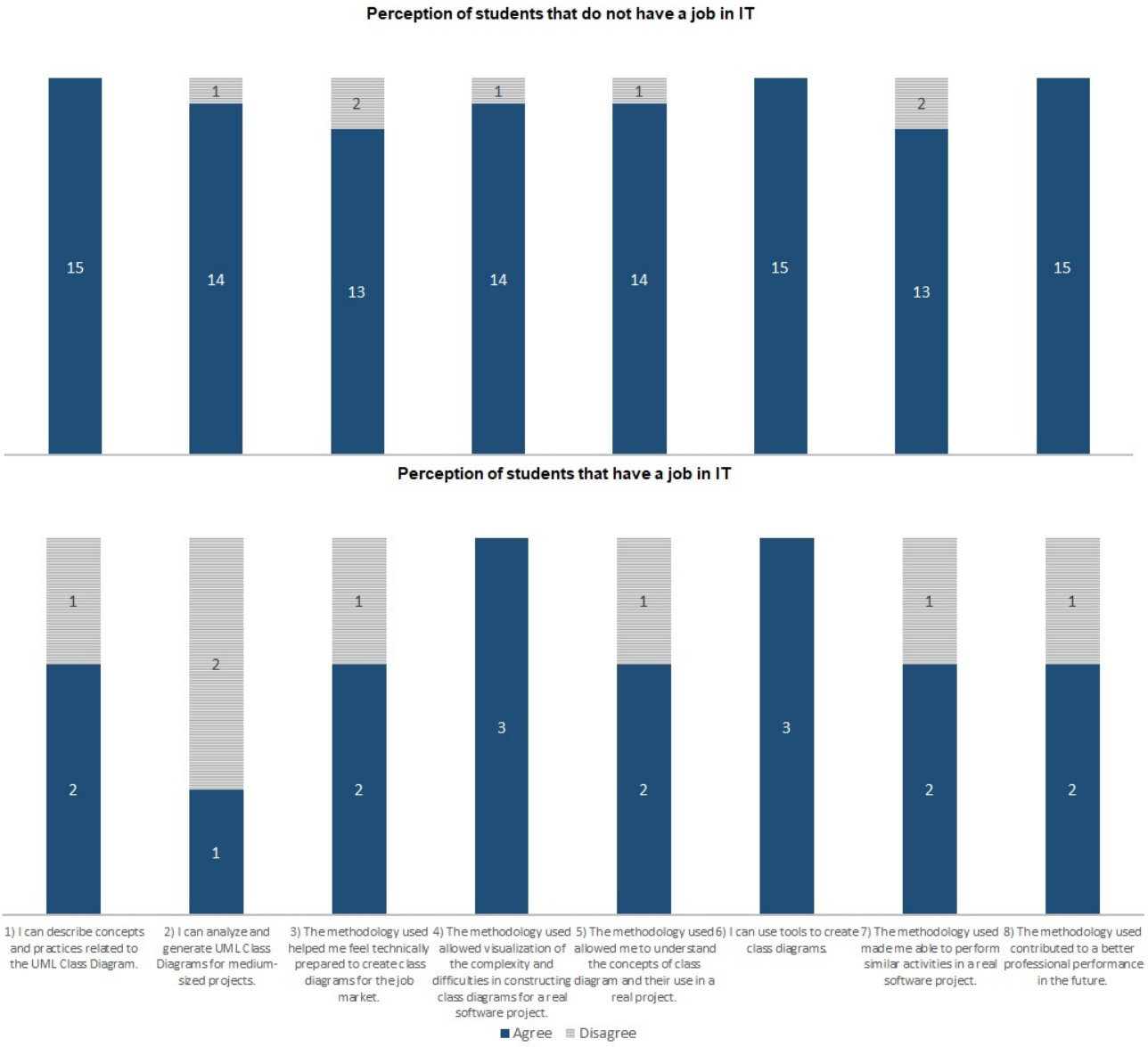

Figure 3: Perception of technical versus professional practice

used in the classroom helped in understanding the concepts, their technical preparation for creating class diagrams for a real software project and contributed to a better professional performance in the future as presented in Figure 4.

In the students' perceptions, the activities performed in the classroom helped to feel more technically prepared for the job market, and to visualize the complexity and difficulties in constructing the diagrams in a real project. But it is important to emphasize that students exposed to the FLOSS project also have acquired skills in the use of tools commonly used in industry, such as GitHub.

Finally, Figure 5 presents the results of students' evaluation of the methodology and its contribution to the development of proactivity, communication, ability to deal with code developed by third parties, to solve problems, to write clear, concise and precise software documentation, and to understand the importance of software modeling.

More than 94% of students stated that the methodology used in the classroom promoted enhanced proactivity. In addition, more than 85% of students reported that the methodology allowed them to have some experience with code developed by third parties. Moreover, 100% of the students stated that the activities carried out contributed to the development of problem solving skills and

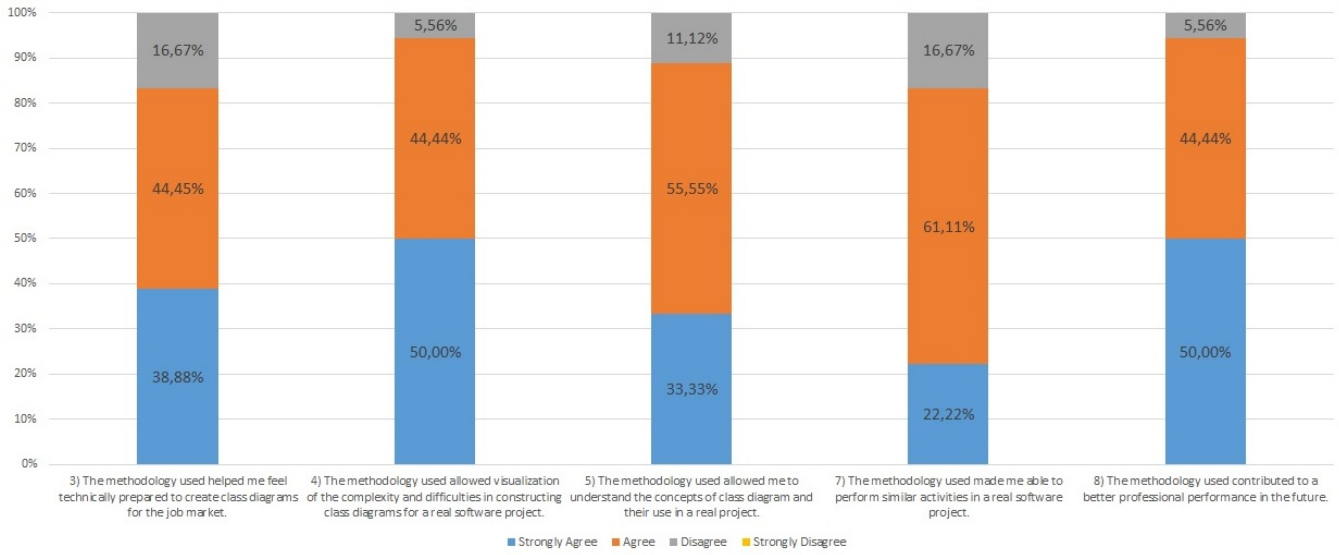

Figure 4: Perception about preparation for the job market

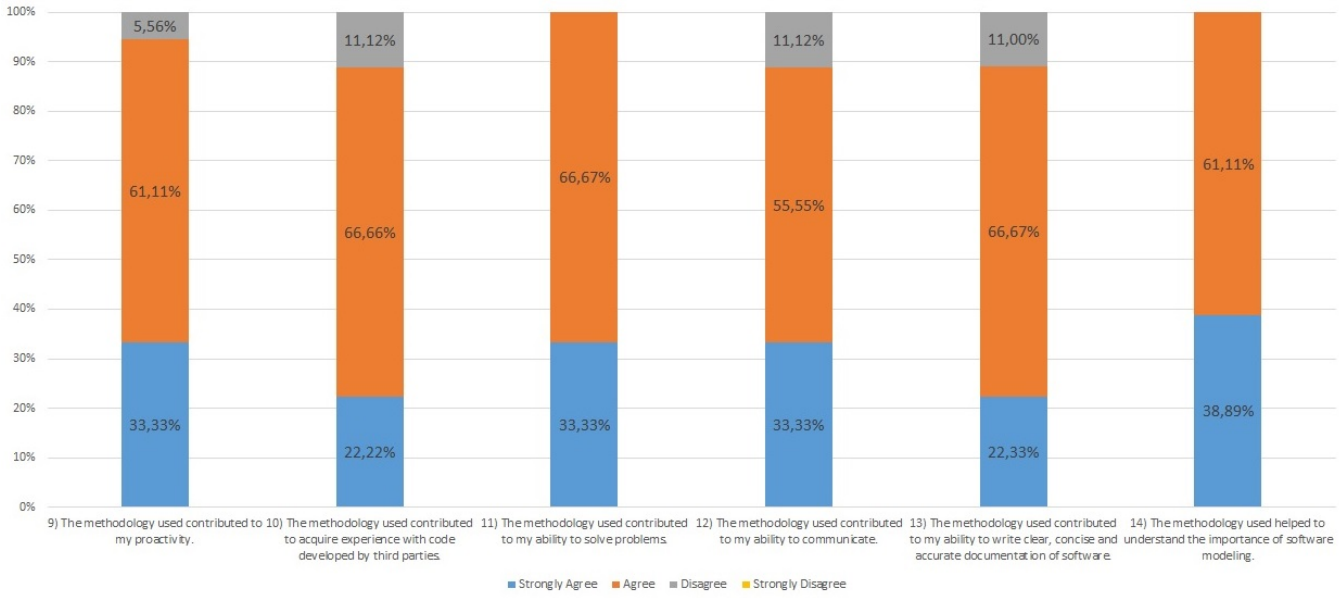

Figure 5: Perception of acquired skills

understanding of the importance of software modeling. Finally, more than 85% of students reported that the methodology used in the classroom contributed to the development of communication skills.

In addition to the accuracy of the data of the students' questionnaire, the evaluation activity performed by the students involved in the case study was analyzed. It was noted that the students made the correct use of notation syntax and semantics in modeling the class diagram applied in a real project, being able to know and understand the concepts and apply them to class diagram.

4.2 The Instructor's Perceptions

The interview was conducted with the instructor after the application of the study. The instructor was motivated to discuss: i) experience with Free Software hosted in repositories; ii) experience with teaching of Software Engineering; iii) experience with teaching UML diagrams; iv) experience with the application of the use of FLOSS projects in teaching the Class Diagram of the UML.

The professor said he had little experience in FLOSS projects, since he had no contact with free software hosted in repositories during the time he worked in the labor market. Despite the 7 years

of teaching experience, the professor declares that he has never used FLOSS projects as an object of study for the teaching of any discipline or specific content taught. He also informed that his interest in the subject was stimulated from a lived experience as a student in a doctorate class.

The instructor has been working with the Software Engineering discipline since 2012, for uninterrupted periods, in classes of several courses, such as: Computer Science, Information Systems, Internet Systems, Informatics Degree and Systems Analysis and Development. Faced with the challenge of considering the particularities of each course, the instructor seeks to improve the teaching and learning process through active methodologies (gamification, host, rotation by stations, case study, among others), using Engineering tools and various applications. In relation to the experience with the teaching of UML diagrams, the instructor reported that he always provided content through dialogic and expository classes, providing lists of exercises for resolution and applying case studies through scenarios.

After applying the proposed intervention, the instructor recognizes that it has been a challenging and rewarding experience. The main benefits observed by the instructor were: i) the opportunity to know and apply new techniques in teaching the UML in the discipline of Software Engineering; ii) to be able to identify criteria for improvement in the learning process of the subject; iii) to increase the interest in studying and researching more about the teaching of Software Engineering using FLOSS projects. The instructor confessed that using a FLOSS project only for teaching the Class Diagram of the UML may not have generated the expected results because the students focused more on getting to know GitHub and doing the initial recognition of the FLOSS project that is dedicated to the correct use of syntax and semantics of the notation presented. Facing this, the instructor showed motivation to participate in other experiences involving the use of FLOSS projects to teach other contents of Software Engineering and interest in better dosing in the planning of activities a coherent time for students to devote themselves to the concepts of the discipline.

5 CONCLUSION

We conclude that using FLOSS projects in the Software Engineering discipline to improve the concepts of UML Class Diagrams, allowed to align theory and practice. In this approach, students have experienced real-world systems modeling, working with third-party software with features similar to those found in the industry and hardly available in academia. In the classroom, students developed social skills such as: communication, interaction, collaboration and proactivity; corroborating what was previously reported by Pinto et al. [22] that the use of FLOSS develops students' social skills. In addition, based on the instructor's report, the students presented abilities related to the tolerance of what is different and acted to the benefit of the work team with focus in the objectives of the group.

As for the challenges faced by the instructor, Morgan, Hislop and Ellis [17] report that instructors also need to help students with the potentially steep learning curve of FLOSS culture and technologies. Therefore, we observe that providing adequate learning support for students using FLOSS projects in the classroom is a challenge as students are unlikely to be familiar with all the details of technology tools.

After using the methodology described in this paper, we believe that FLOSS projects can be used in early or advanced subjects, provided that projects with appropriate size and complexity are selected, considering the students' skills and competences. By considering the use of mature FLOSS projects, where class diagrams tend to become more complex, it is possible to facilitate students' learning and understanding by limiting the scope of the project to drawing the diagram and its components.

Finally, we highlight the instructor effort in the planning and utilization phase of a FLOSS project as object of study in the teaching of UML Class Diagram. However, we realize that in this approach the students behaved actively and were exposed to the types of challenges encountered in the industry, thus being considered a good alternative for learning Software Engineering concepts.

REFERENCES

[1] Robert K Atkinson, Alexander Renkl, and Mary Margaret Merrill. 2003. Transitioning from studying examples to solving problems: Effects of self-explanation prompts and fading worked-out steps. *Journal of educational psychology* 95, 4 (2003), 774.

[2] Benjamin S Bloom, Max D Engelhart, Edward J Furst, Walker H Hill, David R Krathwohl, and Flávia Maria Santana. 1973. *Taxionomia de objetivos educacionais.*

[3] Grady Booch, James Rumbaugh, and Ivar Jacobson. 2005. *Unified Modeling Language User Guide, The (2Nd Edition) (Addison-Wesley Object Technology Series).* Addison-Wesley Professional.

[4] Joseph Buchta, Maksym Petrenko, Denys Poshyvanyk, and Václav Rajlich. 2006. Teaching evolution of open-source projects in software engineering courses. In *null.* IEEE, 136–144.

[5] David Carrington and S-K Kim. 2003. Teaching software design with open source software. In *33rd Annual Frontiers in Education, 2003. FIE 2003.*, Vol. 3. IEEE, S1C–9.

[6] Fragkiskos Chatziasimidis and Ioannis Stamelos. 2015. Data collection and analysis of GitHub repositories and users. In *Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on.* IEEE, 1–6.

[7] CSEET 2016. *The 29th IEEE Conference on Software Engineering Education and Training.* http://paris.utdallas.edu/cseet16/

[8] Ministério da Educação. MEC. 2016. Diretrizes Curriculares - Cursos de Graduação em Engenharia da Computação. (2016). http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=52101-rces005-16-pdf&category_slug=novembro-2016-pdf&Itemid=30192

[9] Sociedade Brasileira de Computação. SBC. 2005. Currículo de Referência para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia da Computação. (2005). http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=52101-rces005-16-pdf&category_slug=novembro-2016-pdf&Itemid=30192

[10] Heidi JC Ellis, Michelle Purcell, and Gregory W Hislop. 2012. An approach for evaluating FOSS projects for student participation. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education.* ACM, 415–420.

[11] Thaís Ferreira, Davi Viana, Juliana Fernandes, and Rodrigo Santos. 2018. Identifying emerging topics and difficulties in software engineering education in Brazil. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering - SBES.* ACM, 230–239.

[12] Mirela Gutica. 2018. Improving students' engagement with large-team software development projects. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education.* ACM, 356–357.

[13] Truong Ho-Quang, Regina Hebig, Gregorio Robles, Michel R. V. Chaudron, and Miguel Angel Fernandez. 2017. Practices and Perceptions of UML Use in Open Source Projects. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP '17).* IEEE Press, Piscataway, NJ, USA, 203–212. https://doi.org/10.1109/ICSE-SEIP.2017.28

[14] Letizia Jaccheri and Thomas Osterlie. 2007. Open source software: A source of possibilities for software engineering education and empirical software engineering. In *Emerging Trends in FLOSS Research and Development, 2007. FLOSS'07. First International Workshop on.* IEEE, 5–5.

[15] Daniel E Krutz, Samuel A Malachowsky, and Thomas Reichlmayr. 2014. Using a real world project in a software testing course. In *Proceedings of the 45th ACM technical symposium on Computer science education.* ACM, 49–54.

[16] Julio Cesar Sampaio do Prado Leite and Cláudia Maria Lima Werner (Eds.). 2008. *Anais do Fórum de Educação em Engenharia de Software, FEES 2008, Rio de Janeiro,*

RJ, Brasil, Outubro, 2008. ftp://ftp.inf.puc-rio.br/pub/docs/techreports/08_43_leite.pdf

[17] Becka Morgan, Gregory W. Hislop, and Heidi J. C. Ellis. 2019. Faculty Development for FLOSS Education. In *IFIP Advances in Information and Communication Technology*. Springer International Publishing, 165–171. https://doi.org/10.1007/978-3-030-20883-7_15

[18] Debora MC Nascimento, Roberto Almeida Bittencourt, and Christina Chavez. 2015. Open source projects in software engineering education: a mapping study. *Computer Science Education* 25, 1 (2015), 67–114.

[19] Debora MC Nascimento, Christina FG Chavez, and Roberto A Bittencourt. 2018. The Adoption of Open Source Projects in Engineering Education: A Real Software Development Experience. In *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[20] Debora Maria Nascimento, Kenia Cox, Thiago Almeida, Wendell Sampaio, Roberto Almeida Bittencourt, Rodrigo Souza, and Christina Chavez. 2013. Using Open Source Projects in software engineering education: A systematic mapping study. In *Frontiers in Education Conference, 2013 IEEE*. IEEE, 1837–1843.

[21] Debora Maria Coelho Nascimento. 2017. *Educacão em engenharia de software com a adoção de projetos de código aberto: uma análise detalhada.* Ph.D. Dissertation. Universidade Federal da Bahia.

[22] G Pinto, F Figueira Filho, I Steinmacher, and M Gerosa. 2017. Training software engineers using open-source software: the professors' perspective. In *Software Engineering Education and Training (CSEE&T), 30th IEEE Conference on*. 1–5.

[23] José Antonio Pow-Sang. 2015. Replacing a traditional lecture class with a jigsaw class to teach analysis class diagrams. In *2015 International Conference on Interactive Collaborative Learning (ICL)*. IEEE, 389–392.

[24] Václav Rajlich. 2013. Teaching developer skills in the first software engineering course. In *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 1109–1116.

[25] Mary Shaw. 2000. Software engineering education: a roadmap. In *ICSE-Future of SE Track*. 371–380.

[26] W. A. F. Silva, I. F. Steinmacher, and T. U. Conte. 2017. Is It Better to Learn from Problems or Erroneous Examples?. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T)*. 222–231. https://doi.org/10.1109/CSEET.2017.42

[27] Ben Skudder and Andrew Luxton-Reilly. 2014. Worked examples in computer science. In *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148*. Australian Computer Society, Inc., 59–64.

[28] Therese Mary Smith, Robert McCartney, Swapna S Gokhale, and Lisa C Kaczmarczyk. 2014. Selecting open source software projects to teach software engineering. In *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 397–402.

[29] IEEE Computer Society and Association for Computing Machinery. 2014. Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. (2014), 134 pages.

[30] Sulayman K Sowe and Ioannis G Stamelos. 2007. Involving software engineering students in open source software projects: Experiences from a pilot study. *Journal of Information Systems Education* 18, 4 (2007), 425.

[31] Jiaxin Zhu, Minghui Zhou, and Audris Mockus. 2014. Patterns of folder use and project popularity: A case study of GitHub repositories. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 30.

A QUESTIONNAIRES

A.1 Students

A.1.1 Part 1: Student profile.

- What is your course?
- What is your period?
- Do you work in the area? If so, what activity do you do? * The area functions are: Programmer, Analyst, Analyst.
- What is your experience by participating in real software projects on other occasions? - Understand real project as: created to meet the need of some user, institution, etc. It does not correspond to a project to meet the requirements of a discipline.

A.1.2 Part 2: About the learning process of class diagrams.

- I can describe concepts and practices related to the UML Class Diagram.
- I can analyze and generate UML Class Diagrams for midsize software.
- The methodology used helped me feel technically prepared to create class diagrams for the job market.
- The methodology used allowed visualization of the complexity and difficulties in constructing class diagrams for a real software project.
- The methodology used allowed me to understand the concepts of class diagram and their use in a real project.
- I can use tools to create class diagrams.
- The methodology used made me able to perform similar activities in a real software project.
- The methodology used contributed to a better professional performance in the future.
- The methodology used contributed to my proactivity.
- The methodology used contributed to acquire experience with code developed by third parties.
- The methodology used contributed to my ability to solve problems.
- The methodology used contributed to my ability to communicate.
- The methodology used contributed to my ability to write clear, concise and accurate documentation of software.
- The methodology used helped to understand the importance of software modeling.

A.2 Instructor

- Experience with Free Software hosted in repositories:
- Software Engineering Teaching Experience:
- Experience with teaching UML diagrams:
- Reporting the experience with the proposed intervention: