Capstone Projects Aimed at Contributing to Consolidated Open Source Projects: a Practical Experience

Juanan Pereira
Escuela de Ingeniería de Bilbao, UPV/EHU
Calle Rafael Moreno Pitxitxi, 2-3
juanan.pereira@ehu.eus
0000-0002-7935-3612

***Abstract*—Due to time constraints (one term or quarterly subjects) software projects used in university classes of Software Engineering are usually limited to small developments, with few people involved and without any previous code base to build upon, that is, without taking into account important aspects like software maintenance or software evolution. Open source software (OSS) is currently being considered as a way of involving students in the realities of professional software development, confronting them with a constantly evolving code base, maintenance, portability problems (compatibility with multiple operating systems), localization and programming styles. It is also remarkable the amount of learning obtained by collaborating in a distributed software development, carried out among a group of developers from different parts of the world. This in turn, allows students to be trained in communication skills to be able to interact with the OSS community. The problem is that, again, it is difficult to integrate this OSS project-based learning into a quarterly course. This work advocates a more feasible scenario, proposing that students that have to develop their capstone projects build them by contributing to consolidated OSS applications. In this context, a practical experience developed with 3 capstone students is shown, detailing the benefits obtained, both from the point of view of the students and the project itself. A series of recommendations are also presented, provided by the students and the teacher involved, so that any interested teacher can replicate the experience with a higher guarantee of success.**

Keywords: Open Source, OSS, FLOSS, Software Engineering, Capstone Projects

I. Introduction

In Software Engineering university classes, Free Libre and Open Source Software (FLOSS) or similarly Open Source Software (OSS) is being considered as a way to immerse students in the realities of software development [1]–[3]. This effort is relevant to address the lack of commitment that can arise when students do not perceive the real usefulness of the subjects of the study program. To combat this lack of motivation, teachers are seeking to integrate course assignments and syllabi within the development of an OSS project.

The benefits are manifold. Reading the project documentation and exploring its source code enables students to learn about programming style, feature design, and other good development practices. Working on OSS projects also allows them to have practical experience in issues related to code maintenance and evolution, portability, localization and internationalization. Students must strive to ensure that their contributions are compatible with the current project design, maintaining its evolution rules.

Also noteworthy is the learning obtained by collaborating in a distributed software development, seen as a collaborative process, carried out among a group of developers distributed throughout the world. This allows students to be trained in communication skills that they must put into practice to interact with the OSS community.

All these benefits have made teachers promote the use of OSS in the classroom. In Spain, it is worth highlighting the Free Software University Contest [4], a national initiative in which, since 2006, students from Spanish universities have participated in developing free systems.

However, instructors warn of potential problems. One of the drawbacks is that there are not yet enough integration experiences [2], which makes it difficult to develop the entire software engineering syllabus based on OSS. Another initial pitfall is selecting the appropriate OSS project to contribute to [5].

Problems related to the size of the task (contributing to an OSS project) versus the duration of the course (quarterly) and the students' prior knowledge are also cited [6]. It is difficult to fit all the pieces.

With the aim of exploring new ways of supporting the teaching of software engineering based on contributions to OSS projects, as well as looking for an alternative to its use in class (where time restrictions and prior knowledge condition its use), this work documents a practical framed within three Capstone Projects (FDP, Final Degree Projects) where three students contributed features to a single OSS project.

The rest of this work has been divided into four parts. The following section details the context in which the FDPs were developed and how the project on which the contributions were made was selected. Section three details the benefits obtained by the students and the project in this. In the fourth section, a compendium of recommendations is made for those who are interested in repeating the experience and in the last section the overall conclusions are reflected.

II. Context and Project Selection

Every year at the Bilbao School of Engineering, 4th grade students in Computer Engineering must develop their Final Degree Project (FDP). Many times the project consists of implementing some type of application, usually from scratch. Something that does not match what they will find in the labor market [7]–[9].

There are usually very good jobs but the vast majority end up in the drawers of the library or online repository.

One way to motivate students to develop their knowledge of software engineering is to involve them in the development and maintenance of consolidated open source software projects [10], [11]. In the 2019/2020 academic year this author decided to test that option with three final degree projects and document the experience in this article.

The idea was to convince three brilliant students so that their FDP was aimed at improving an open source application used internationally. The first problem is choosing a good project. We can talk at length about what constitutes a good OSS project to use in the area of software engineering education [2], [6].

In the case at hand, the project had to meet several requirements: be a live project (have commits - code contributions - distributed throughout the year), be developed in Java (the language that students master during the career), not be a trivial project (the one used as the basis of the TFG contains almost 800 Java classes and 95000 lines of code) and, if possible, be a project known to the students. GanttProject (https://github.com/bardsoftware/ganttproject) has all these characteristics and it has also been used as a tool in previous subjects for Gantt chart designing.

Three students, O., A., and U. had been hard at work selecting and fixing bugs from the GanttProject bug list. Working to improve a free software application has brought us multiple benefits, both for the students and for the project itself. The next section details this list of benefits.

A. *Benefits for the student*

1) Learn how to work with the Github Flow

The first thing they have learned is to work with the GitHub Flow workflow (https://guides.github.com/introduction/flow/), a lightweight, branch-based workflow that helps manage input from code of distributed teams and workgroups. Specifically, they studied how to create a *fork*, how to keep it updated with the original version (*upstream*, see Fig. 1), how to create branches and how to generate *Pull Requests*. They had never implemented this workflow before (the most used among open source projects).

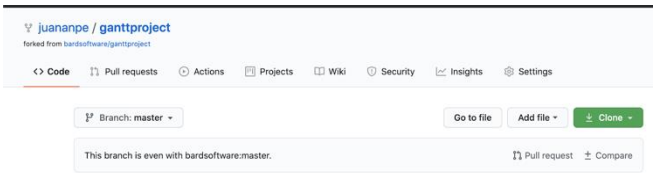

Fig. 1. Learning how to create Pull Requests, Issues or keeping the branches of a fork updated with the main repository are just some of the tasks that they learn in these FDPs.

2) Defend and discuss ideas

Students learn to cope with the review of their code by external programmers (see Fig. 2) each time they propose a contribution in the form of Pull Request. This type of informal code review, known as *modern code review[12]*, simplifies the formal review process, complementing it with automation and bug detection tools and allowing the lead developers of the project to analyze contributions (Pull Requests) as soon as they these are proposed. On many occasions the code that the students contribute works correctly but the main developers find elements for improvement and suggest refactorings to make the code more readable, maintainable and reusable (non-functional requirements that in many cases go unnoticed by the students during the grade)

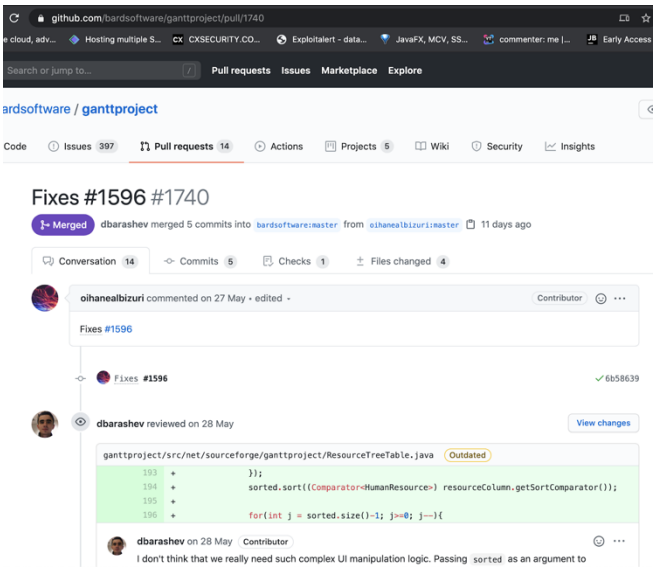

Fig. 1. The quality control of the consolidated open source projects includes a review of the code of the contributions. For the students, this was the first time that an external developer reviewed their contributions in detail..

3) Automatic tests

GanttProject includes an automated tests section (see Fig. 3) as part of the continuous integration process for contributions. Students should understand how tests work and learn how to use them on their own code before submitting their contribution as a PullRequest.

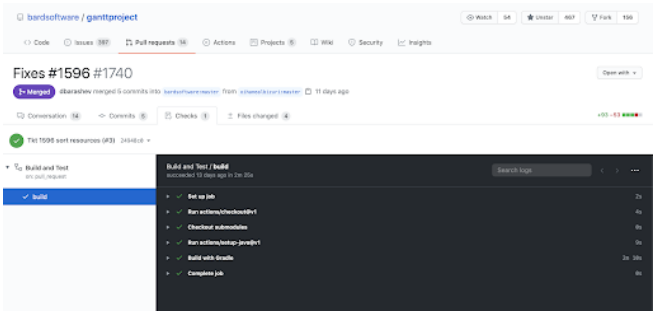

Fig. 2. The tests include the verification of functionalities on different platforms (Windows, macOS, Linux), something that is not always worked on in academic environments.

4) Communicate with external developers

Communicating in English and proposing corrections or designs for new functionalities in the platform's support forum has been another novel task for the students involved (see Fig. 4).

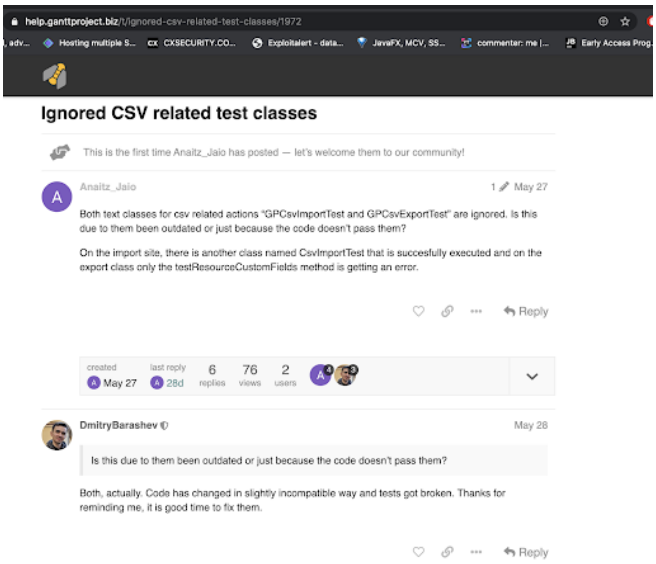

Fig. 3. Before solving a problem or beginning an implementation, students should comment on their intentions in the forum.

B. *Benefits for the project*

Finally, GanttProject itself, as an open source project, has also benefited from the following gains.

1) Get reusable documentation

One of the students' goals was to create documentation that will really serve the future. Specifically, it was aimed so that other developers (future students) can begin to contribute with a smoother learning curve than these three pioneers.

Among this documentation are the classic UML diagrams (class, sequence, communication ...) but also especially useful diagrams that mix the graphical interface with the Java classes that implement the. An example can be seen in Figure 5.

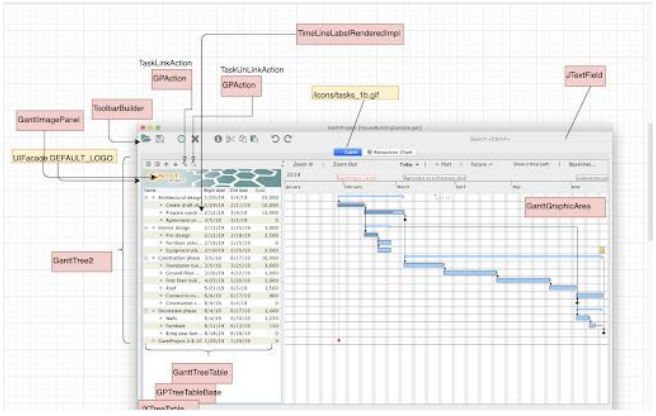

Fig. 4. This type of diagram with screenshots of the application labeled with the name of the main classes that implement some of the components can be of great help for new contributors.

The diagrams have been collaboratively generated between the 3 students who have formed the initiative.

2) New features or bug fixes

These are the commits in the master branch of each of the students (which in turn serve to polish their curriculum with the possibility of public verification). All links correspond to the repository https://github.com/bardsoftware.

- /commits?author=Anaitz98 (8 *commits*) (3 accepted PRs)
- /commits?author=oihanealbizuri (7 *commits*) (3 PRs, 2 accepted and 1 rejected)
- /commits?author=urtzipu
- /commits?author=upuente001 (4 *commits*) (3 PR, 2 accepted and 1 rejected)

In Table 1 we can see the complete list of Pull Requests sent.

III. TEACHER RECOMMENDATIONS

After the experience, we gather below some reflections and suggestions that may be of help to anyone who wants to replicate the same process of development of FDPs based on improvement of FLOSS applications:

- The teacher must involve himself / herself in the project before his / her students. They have to dive into the code, trying to understand the basic architecture. Fix an issue, create a PR themself and meet the main developers. That will help smooth out the entry to new programmers. Example: https://github.com/bardsoftware/ganttproject/commits?author=juananpe
- It is advisable to help students to analyze the list of project issues to locate a small collection of "feasible" errors, in the sense that they can be corrected in 3 months (approximately the duration of the FDP). The author selected a couple of issues for each student, but they were asked to further analyze the whole list and create a set of 4 or 5 each. The goal was that after discussing them in a group, they aimed to implement at least 3 of them. And they succeeded.
- The selected issues must be ordered by difficulty: easy, medium, difficult. Some projects have easy issues labeled with labels such as beginner-friendly, easyfix, beginner, easy, difficulty / low, etc.

  It is recommended that students initially get involved with an easy-to-solve issue, such as eliminating dead code and generating a PR with that patch, so that they learn the basic rudiments of GitHub Workflow and the workflow of the OSS project in particular.
- If possible, the first time, students should be guided with an outline of the solution of any of the issues to be corrected. A sketch of the solution design would suffice. It involves a lot of work for the teacher, but it should be seen as an investment that can be reused in the future and as a measure to ward off possible fears from the students.
- Create a group on Telegram - or another instant messaging tool - with the students who joined the initiative. They will help each other, a sense of community will be created and students will feel supported [13].

IV. STUDENT RECOMMENDATIONS

At the end of the projects, the three FDP students completed a survey asking for recommendations for future

students. Specifically, students were asked to answer these questions:

- What aspect(s) of the project have you found most complex(s)?
- What recommendations would you give to next year's students who want to carry out their FDP improving GanttProject or another free software project?
- With what aspect of the project have you noticed the most learning?

A. *More complex aspects of the project*

"The most complex of all, in my opinion, has been finding the exact point where the error was located or the place where the improvement should be developed"

"Initially, prepare the development environment, thus being able to compile the application. "

"- Understanding of the project structure (and code)

- 'Break' the fear barrier and take the initiative to collaborate on a real project "

B. *Recommendations for students of future courses*

- "Perhaps choosing another OSS software that does not have so many classes and allows you to get a more general picture of the project, since there are many aspects of GanttProject that I still do not know despite having worked on it. "
- "Understand from the beginning the structure of the application with which you are going to work, prepare the development environment well and carry out well-planned work, without leaving everything for last. "
- "I think it would be a good idea to start with a task that is something like a current code analysis, in which each student does research on the project by generating or expanding parts of a previously provided class diagram. "

C. *Aspects of the project that provided the greatest opportunities to learn*

- "As I have progressed in development, I have realized that understanding other people's code and locating parts of the code has become easier and easier for me, and I have done it in less time. "
- "The use of tools such as Git and sdkman - a tool to easily manage different versions of the Java Development Kit and Gradle-, as well as the use of the debugger to see the application processes and thus understand the function of various methods and classes in a large project."
- "To read the project structure, management with git and versions and to use different data structures from those studied in class."

V. CONCLUSIONS

A practical experience has been analyzed with three students who have focused their final degree projects in software engineering towards the development of improvements to a consolidated OSS project. In this first approach, it has been the teacher who has helped to choose the target project to contribute to (GanttProject). Each student had to complete the development or fixing of three issues and follow the usual workflow in order to integrate their contributions into the master branch of the project. The benefits are multiple, both for students (learning to understand and locate features of a project with hundreds or thousands of classes, passing code quality controls, defending design ideas in work groups, working with a distributed version control system,. ..) as for the chosen OSS project itself (get reusable documentation, get fixes or new functionalities).

Specific recommendations have been proposed, both by the teacher and the students involved, so that anyone interested can replicate the process.

Among the most complex aspects of the project (which coincides with the most valued factor in student learning) is the understanding of foreign code structures and locating those parts of the code that affect the error to be corrected or functionality to be implemented. It is therefore important to focus on the study of foreign code (not your own) in subjects of the degree in computer science, something in which the study of open source software can be of great help. Analyzing how to achieve this integration between the study of open source applications and the teaching plan of a subject (such as the Software Engineering subject) is precisely one of the lines of future work. The teaching objectives and competences to be obtained in the subject must be linked with possible exercises, tasks and interactions related to the project to which they will contribute, taking into account the time limitations inherent to a four-month course or the different degree of initial knowledge exhibited by the students.

TABLE I. CONTRIBUTIONS TO HTTPS://GITHUB.COM/BARDSOFTWARE/GANTTPROJECT

Student	***Issue***	***URL***
oihanealbizuri	Sort resources by name	/commit/2a92442aae078d001f7c403925b2a1a1afa2aaf5
oihanealbizuri	Refactor	/commit/7d1cbe3fbeaffcf322cd082c4a298f6895ecdd84
oihanealbizuri	Option to change cost display format #1659 (not merged yet)	/issues/1659
upuente001	Tck1665keyboardshortcuts	/commit/909af906329323adcf4e5e767cd0c36bba5b8809
upuente001	Remove unused code	/commit/d7cac4250ca77d4df728b183ca660a90fb3ab49e
upuente001	Tkt 1610 change logo without restart (not merged yet)	/ganttproject/pull/1723
Anaitz98	coderefactor	/commit/748a9cea57282fc372f36c61d

Student	*Issue*	*URL*
		d270b98bee34650
Anaitz98	ExportCSVtestsolving	/commit/2bee42e34347c279f89b24e0bde2d571f91d9b18
Anaitz98	tkt1667rememberLastImportFolder	/commit/fa9f3df2f845d4c53e2285dff0c27e90d83bf3b5

REFERENCES

[1] D. M. Nascimento *et al.*, "Using Open Source Projects in software engineering education: A systematic mapping study," in *2013 IEEE Frontiers in Education Conference (FIE)*, 2013, pp. 1837–1843, doi: 10/gf8h86.

[2] G. H. L. Pinto, F. Figueira Filho, I. Steinmacher, and M. A. Gerosa, "Training software engineers using open-source software: the professors' perspective," in *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, 2017, pp. 117–121, doi: 10/gf4m2v.

[3] A. Sarma, M. A. Gerosa, I. Steinmacher, and R. Leano, "Training the future workforce through task Curation in an OSS ecosystem," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 932–935, doi: 10/gf8h89.

[4] P. Neira Ayuso and M. Palomo Duarte, "Innovación educativa con software libre," *Actas de la VI Jornadas Internacionales de Innovación Universitaria (JIIU 2009). Villaviciosa de Odón (Universidad Europea de Madrid)*, 2009.

[5] K. Toth, "Experiences with open source software engineering tools," *IEEE software*, vol. 23, no. 6, pp. 44–52, 2006, doi: 10/b6rx2s.

[6] T. M. Smith, R. McCartney, S. S. Gokhale, and L. C. Kaczmarczyk, "Selecting Open Source Software Projects to Teach Software Engineering," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, New York, NY, USA, 2014, pp. 397–402, doi: 10/gf4m2r.

[7] H. J. C. Ellis, G. W. Hislop, and M. Purcell, "Project selection for student involvement in humanitarian FOSS," in *2013 26th International Conference on Software Engineering Education and Training (CSEE T)*, May 2013, pp. 359–361, doi: 10/gf4m3f.

[8] H. Ellis, R. A. Morelli, and G. Hislop, "Support for educating software engineers through humanitarian open source projects," in *2008 21st IEEE-CS conference on software engineering education and training workshop*, 2008, pp. 1–4, doi: 10/c8xwdd.

[9] C. Chavez, A. Terceiro, P. Meirelles, C. Santos Jr, and F. Kon, "Free/libre/open source software development in software engineering education: Opportunities and experiences," *Fórum de Educaçao em Engenharia de Software (CBSoft'11-SBES-FEES)*, 2011.

[10] 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*, 2011, pp. 268–272.

[11] M. Müller, C. Schindler, and W. Slany, "Engaging students in open source: Establishing foss development at a university," 2019, doi: 10/ghchgq.

[12] P. Rigby, B. Cleary, F. Painchaud, M.-A. Storey, and D. German, "Contemporary peer review in action: Lessons from open source development," *IEEE software*, vol. 29, no. 6, pp. 56–61, 2012, doi: 10/ghcmww.

[13] J. Pereira, "Motivating users to online participation. A practice-based comparison between moodle forums and telegram groups," *The International journal of engineering education*, vol. 35, no. 1, pp. 409–416, 2019.