The Journal of Systems & Software 144 (2018) 409–422

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Collaborative and teamwork software development in an undergraduate software engineering course

Claudia Raibulet*, Francesca Arcelli Fontana

Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Viale Sarca 336, Edificio 14, Milan 20126, Italy

ARTICLE INFO

Keywords:
Collaborative software development
Teamwork
Software engineering course
GitHub
SonarQube
Microsoft project

ABSTRACT

Two key elements of modern software development are collaboration and teamwork. Current methodologies (e.g., agile) and platforms are based on these key elements. This paper describes our experience in stimulating collaboration and teamwork activities of students in the context of a software engineering course at the third year of an undergraduate program in computer science at the University of Milano-Bicocca in Italy. The students were asked to develop a software project in teams of 3 to 5 students for the final exam of the course. The students used GitHub as a collaborative software development platform. In addition, they analyzed the quality of the developed software through SonarQube. The students were also asked to perform project management tasks (e.g., the Gantt) using the Microsoft Project tool. At the end of the course, we gathered the student feedback through a questionnaire on their collaboration and teamwork experience (through GitHub and Microsoft Project tools) and on the use of a software analysis assessment tool, i.e., SonarQube. From their feedback, the students were enthusiastic about working in teams for their project development and about learning how to use tools which are exploited not only in the academic world but also in industry.

1. Introduction

To limit the gap between the expectations of the industry and the preparation of undergraduate students concerning their first software engineering job, we have introduced, in an undergraduate software engineering course, mechanisms to encourage students to collaborate, share their knowledge, and to work with tools used in the industry for software development. Collaboration and teamwork are as important as the technical skills for the future software engineers, because students will be involved in industrial software projects where it is needed to understand as much as possible the overall project in order to be able to provide the own contribution. As observed in van der Duim et al. (2007), the social aspects make teamwork different from the individual work. Most of the current education systems focus and emphasize individual work, while the industrial environment requires cooperative and collaborative work in small to large teams.

In this paper we describe a case study we performed in order to collect student feedback on their perception of using mechanisms and tools which support their collaboration and communication. The case study was applied at the end of the software engineering course in the context of an undergraduate program in computer science at the University of Milano-Bicocca. Essentially, we proposed to students the following mechanisms to stimulate their collaboration and teamwork:

- perform seminar and lab activities in teams;
- use tools such as GitHub, SonarQube, and Microsoft Project during labs;
- develop a software project as a course exam in teams by exploiting the tools used in the labs.

Students learnt how to use the tools during the lab sessions and then used the tools for the lab exercises and for the final exam project. The aim of this case study is to improve the learning objectives by enhancing the teaching of the tools in the lab, by assisting with the main difficulties they found in using the tools and the most unclear functionality or aspects of the tools. In particular, for software quality analysis, we are interested in understanding if the students could identify possible drawbacks of the tool or some missing functionality.

The choice of these tools was not incidental. GitHub is currently used both in the industry (Cosentino et al., 2017) and in universities (Zagalsky et al., 2015) and is currently considered the most popular social coding platform. In universities, GitHub is used as a collaborative development platform for software projects and as a social meeting platform for courses (Feliciano et al., 2016; Zagalsky et al., 2015). Due to our interest in GitHub regarding the quality of software projects and the use of GitHub for their development (Arcelli Fontana and Raibulet, 2017; Roveda et al., 2017), in this paper we describe how we introduce

* Corresponding author.
E-mail addresses: raibulet@disco.unimib.it (C. Raibulet), arcelli@disco.unimib.it (F. Arcelli Fontana).

https://doi.org/10.1016/j.jss.2018.07.010
Received 23 December 2017; Received in revised form 8 June 2018; Accepted 3 July 2018
Available online 07 July 2018
0164-1212/

GitHub in the context of a software engineering course. We describe the student feedback regarding the use of GitHub for the development of different software Web application projects assigned to teams composed of 3 to 5 members. We focus on the student feedback on the communication and teamwork aspects concerning the project development, and on how students managed and assigned tasks among team members.

SonarQube is one of the most frequently used tools for monitoring and evaluating the quality of the software. It is cited in several scientific articles as one of the most widely used software quality analysis tool in the industry (Ernst et al., 2017; Kosti et al., 2017; Yli-Huumo et al., 2016). SonarQube is exploited in particular during the development phase of software. Furthermore, it is used to compare the quality of the projects during their evolution (Letouzey, 2012). SonarQube provides the ability to highlight different kinds of code violations, known as issues, enabling the systematic improvement of code quality through a process of continuous inspection.

Microsoft Project is a powerful tool for software project management which is widely used in the industrial world. It provides functionalities for resource, time, and budget management. It also enables the sharing of the project information, as well as communication and collaboration among team members.

To obtain feedback on our course enhancement, at the end of the software engineering course, we gathered the student feedback through a questionnaire comprised of 39 questions divided into three categories. We referred to their experience in using the three above mentioned tools for developing software in a collaborative way (i.e., GitHub), for analyzing the quality of the developed software (i.e., SonarQube), and for performing project management activities (i.e., Microsoft Project). The overall feedback has been positive and the students have been responsive to both the application of the collaborative and teamwork mechanisms and to the use of the new tools. Based on this feedback, in the last section of this paper we have outlined the potential improvements in teaching and using these tools for the further editions of the course.

To guide our research, we have defined the following research questions (RQ):

RQ1: What is the student feedback on the benefits of using GitHub for software development regarding collaboration and teamwork?
RQ2: What is the student feedback on using SonarQube for software quality analysis?
RQ3: What is the student feedback on the benefits of using Microsoft Project for project management regarding collaboration and teamwork?

The remainder of this paper is organized as follows; Section 2 presents the related work. Section 3 introduces the software engineering course, including how GitHub, SonarQube, and Microsoft Project were used during the course. Section 4 describes how the student feedback has been collected. Section 5 analyses the student feedback. Threats to validity are dealt with in Section 6. The outcomes of our study including the answers to the RQ are summarized in Section 7, while future development is detailed in Section 8.

2. Related work

The related work for this paper may be organized as follows:

A teaching collaborative software development and teamwork through project development in software engineering courses;
B use of tools i.e., GitHub[1] for teaching collaborative software development in software engineering courses;
C teaching software quality assessment through project analysis in software engineering courses;
D use of tools i.e., SonarQube[2] for teaching software quality assessment in software engineering courses and for teaching students to improve the quality of their projects;
E teaching software project management in software engineering courses;
F use of tools, i.e., Microsoft Project[3] for teaching software project management in software engineering courses.

Items A, C, and F from the above list are vast and imply further refinement. For example, Ludewig and Begicevic (2012) emphasizes the importance of the project work in groups during the whole software engineering curriculum at the University of Stuttgart in Germany. Similarly, Kilamo et al. (2012) describes the use of KommGame environment to support the students collaboration during a software engineering course at Tampere University of Technology. (Silva et al., 2017) investigates the effectiveness of code quality tools to support the evaluation of the student projects in software engineering courses. Moreno et al. (2016) emphasizes how software engineering courses may be enriched with software project management knowledge based on the Project Management Body of Knowledge (PMBOK, 2013). However, these papers do not mention the use of GitHub, nor the use of SonarQube or Microsoft Project. Thus, in this paper we address the B, D, and F items from the above list.

2.1. Exploiting GitHub in education

Our search on the main engines obtained few results (all published in the last 3 years) on using GitHub for collaborative and teamwork software development in the context of software engineering courses in universities (see Table 1 - second column). The keywords mentioned in the search included: GitHub, collaborative, teamwork, software development, students and software engineering course.

Feliciano et al. (2016) and Zagalsky et al. (2015) investigate the use of GitHub for hosting course content and student assignments in software engineering courses at the University of Victoria in Canada. They consider both the instructor/professor and the student perspectives. The instructors and students had no previous experience in using GitHub. We confirm that the benefits of using GitHub in software engineering courses mentioned in Feliciano et al. (2016), (i.e. gaining and demonstrating industry-relevant skills and practices, enabling cross-team collaboration and contributions, encouraging student contributions to course content, breaking down the walled garden, and version controlled assignments), resulted also from our case study. The difference between this paper and the work presented in Feliciano et al. (2016) and Zagalsky et al. (2015) is that our work considers GitHub for a software development project for an exam and that the professors had already used GitHub before in software engineering activities.

Francese et al. (2015a) and Francese et al. (2015b) present a similar work with our case study done in Italy in the context of a Mobile Application Development course. Francese et al. focused their work on the development of a mobile application for Android devices. GitHub has been used as a collaborative development platform by the students to develop their project counting also as the exam for their course. The authors required the student feedback on the developed project through 9 questions. From the GitHub point of view, they defined 3 questions: managing the project with GitHub was easy?, using the software configuration management features of GitHub was easy? and using the communication features offered by GitHub was easy?). To summarize, Francese et al. focused on the development of a mobile application,

[1] https://github.com/

[2] https://www.sonarqube.org/

[3] https://products.office.com/it-it/microsoft-project-2013

Table 1
Search Results for GitHub and SonarQube.

Search engine	GitHub	SonarQube
IEEE Xplore	(Feliciano et al., 2016) (Tirkey and Gary, 2017) (Zakiah and Fauzan, 2016)	(Delgado et al., 2017) (Gomes et al., 2017) (Rong et al., 2017)
ACM Digital Library	(Feliciano et al., 2016) (Zagalsky et al., 2015)	(Falessi and Kruchten, 2015)
Web of Science	(Feliciano et al., 2016) (Francese et al., 2015a) (Zakiah et al., 2016)	(Bai et al., 2017) (Delgado et al., 2017) (Gomes et al., 2017) (Rong et al., 2017)
Science Direct	(Francese et al., 2015a)	–
Scopus	(Feliciano et al., 2016) (Francese et al., 2015a) (Francese et al., 2015b) (Tirkey and Gary, 2017) (Zakiah and Fauzan, 2016)	(Bai et al., 2017) (Delgado et al., 2017) (Falessi and Kruchten, 2015) (Rong et al., 2017)

while we investigate the effectiveness from various points of view of using GitHub in the context of a software engineering course.

Tirkey and Gary (2017) exploits GitHub to version the content of the software engineering courses at the Arizona State University in USA. Not considering GitHub for a software development project for the students, this paper is out of the scope of our paper.

Zakiah and Fauzan (2016) presents a similar work with our case study done in Indonesia. It has involved 72 students attending software engineering courses in 2 universities and 3 lecturers. Zakiah and Fauzan have defined a questionnaire with 10 questions for students and a questionnaire with 10 questions for lecturers. The questions for the students are similar to the questions we have defined as general questions on GitHub. However, the questions are mainly focused on the student feelings in using GitHub (e.g., 4 of the 10 questions are: is studying software engineering with GitHub boring/frustrating/pleasant/surprising?). Zakiah and Fauzan have investigated how GitHub can be used during the lessons and assignments. In our case study, we asked students to use GitHub for the development of a software project counting as an exam.

2.2. Exploiting SonarQube in education

Our search on the main engines obtained few results on using SonarQube for software quality assessment in the context of software engineering courses in universities (see Table 1 - third column). They are all published in 2017 with one exception. The keywords mentioned in the search included: SonarQube, software development, students, and software engineering course.

Bai et al. (2017) indicates that SonarQube is used by the professors to analyze the code written and submitted by students for assignments in a programming course. A similar work we know even if not present in the results of our search on the main engines is (Lauvas, 2015) which describes the improvement of the student code after introducing two additional aspects in a programming course: portfolio assessment and test coverage. The author compares the quality of the code submitted by the students when these two aspects were not present in the syllabus of the course and the quality of the code submitted by the students when these two aspects were discussed during the course. The code produced from the two succeeding classes was analyzed with SonarQube by the professors.

Rong et al. (2017) introduces DevOpsEnvy, an education support system for professors. Through DevOpsEnvy, professors manage and monitor the student teams practicing DevOps. This system integrates various open source tools such as GitHub, Jenkins, SonarQube, Maven, Gradle, and Docker.

Falessi and Kruchten (2015) outlines the importance of introducing the notion of technical debt in the software engineering curriculum. As an example of tool which they propose to the students to assess the quality of their code is SonarQube.

Delgado et al. (2017) presents the evolution of a project-based software engineering course during 3 years. The project-based course was focused on the application of the agile practices during the development of a project in teams. The project was developed during the course, and not at the end of the course as in our case study. The development framework for the projects changed in time from Grails to Rails. When Grails was used, also SonarQube was used by the students to analyze their software quality. When Rails replaced Grails, SonarQube was abandoned. The students were asked to provide their feedback on the application of the agile process (e.g., incremental or iterative development, collective ownership, code refactoring, fixed length iterations, unit testing, continuous delivery and integration, periodic meeting, pair programming, simple design, user stories, planning game) rather than on their experience in using tool support for their teamwork.

de Andrade Gomes et al. (2017) introduces the SMaRT tool for teaching software quality aimed to help students improve their code and programming skills. SMaRT uses SonarQube and Eclipse IDE. The students use SMaRT individually for their source code assignments. The professors analyze the submitted code and provide feedback to students.

In de Andrade Gomes et al. (2017), Bai et al. (2017), Lauvas (2015) and Rong et al. (2017) SonarQube is used by the professors to assess the quality of the individual work of the students in the context of the programming courses. Falessi and Kruchten (2015) proposes to the students the use of SonarQube for analyzing the technical debt of their individual work. The work presented in Degado et al. (2017) mentions the use of SonarQube for the software analysis in an agile project-based software engineering course and does not investigate the advantages or limitations in using it during the project development.

In our paper, we ask students to use themselves SonarQube to assess and improve the quality of a project resulted from their collaboration and teamwork and then submit their work for the exam. In addition, we asked students their feedback on the use of SonarQube.

2.3. Exploiting Microsoft Project in education

Our search on the main engines obtained several results on using Microsoft Project in education, especially related to project management courses. Those closest to the content of our paper are introduced in this section. The keywords mentioned in the search include: Microsoft Project, project management, students, and software engineering course.

In Groth and Hottell (2006) students were encouraged to do planning tasks for a capstone project course by using Microsoft Project tool. Groth and Hottell (2006) focuses on the definition of a capstone project course and its evolution in time, rather than introducing project management activities in the context of a software engineering course.

Car et al. (2007) describes the introduction of a project management course for undergraduate and graduate students. Similarly, Stoica and Islam (2012) introduces a project management course for undergraduate students together with a basic and an advanced course on software engineering. Garcia (2015) presents an experience of teaching a project management course for undergraduate students. The objectives of these courses are significantly larger than the introduction of the basic notions of software project management in our software engineering course. However, we outline that both courses use the same tool support, i.e., Microsoft Project.

To summarize, as far as concerns our knowledge, there is no related work on the use of all the three tools together GitHub, SonarQube, and Microsoft Project in software engineering courses in universities. We have found very few papers on exploiting GitHub for collaborative software development and on using SonarQube for software quality assessment in the context of software engineering courses. Microsoft

Project is typically used in dedicated project management courses.

3. The software engineering course

The Software Engineering course is collocated at the third year of an undergraduate degree in Computer Science at the University of Milano-Bicocca in Italy. The Software Engineering course focuses on software design and introduces advanced principles, techniques, and tools for software development and software quality evaluation. More information about the course description and structure is available at the web site in Italian and English[4]

Co-related to this course, in the previous two years of the undergraduate curriculum, the students have attended an Object-Oriented Programming in Java[5] course introducing the basic concepts of object-orientation and a Software Analysis and Design[6] course introducing UML concepts, design patterns, and subversion (SVN) fundamentals.

The course was a semester long. It was composed of 2 hours of lessons, 2 hours of seminars, and 2 hours of labs per week for a total of 10 weeks. The course has been enhanced during the 2016–2017 academic year with mechanisms, activities, and tools which encouraged the collaborative and teamwork software development. During this academic year, 41 students (4 female and 37 male students) were involved. The main mechanisms, activities, and tools introduced during the course to enable the collaboration and teamwork of students were the following.

GitHub has been adopted as a collaborative development platform for software projects. GitHub has been presented to the students during the first laboratories and further used during the entire course. The focus has been concentrated on branches, staging, commit, push and pull functionality and conflicts resolution. Further, students are introduced to Maven, a build system for Java software.

SonarQube has been adopted as a software quality assessment tool for Java projects. SonarQube has been presented to the students during the laboratories. The focus has been concentrated on the installation, configuration and use of SonarQube (e.g., the various plug-ins of SonarQube, the configuration of the database used by SonarQube, the analysis of Java projects through the Maven plug-in). The students also performed the analysis of various available Java open source projects during the laboratories.

Microsoft Project has been adopted as a tool for software project management. During two seminars, the students were taught the basic concepts of project management, focusing mostly on project life cycle and organization, time, cost, human, communications and risk management (as described in PMBOK (2013)). Furthermore, the students were shown two simple case studies and asked to do themselves a case study using Microsoft Project.

For the exam, the students were asked to complete several exercises assigned during the lab activity and develop a Java software project in teams by applying the mechanisms, knowledge, and tools seen during the course. The students were asked to organize themselves in teams of 3 to 5 students and communicate the professors/instructors the member names of each team via email. In this way students were encouraged to choose their team members based on their preferences and academic knowledge/skills of their mates. Students were required to use:

- GitHub, as a development collaborative platform for their Java software project;
- SonarQube, as a software quality assessment tool for their software project;
- Microsoft Project, as a project management tool for the development of the software project.

Each student team received a different software project to be developed, but the projects were similar in the requirements and work load. The projects consisted of the development of different software Web applications similar to those used by students in their everyday life, such as:

- the booking and selling of cinema/theatre tickets (2 projects);
- the booking and selling of train/flight tickets (2 projects);
- the booking of cars or bikes in a car/bike sharing system (2 projects);
- the booking of rooms in a hotel (1 project);
- the management of services offered by a gym/fitness center (1 project);
- the management of the papers submissions for a conference/journal (2 projects).

The students had 1 month time to develop the software project from the moment they received the project description. The requirements of the project concerned the analysis, design, and implementation of the project. The students were required to provide a documentation concerning the design of the project. The documentation has to include the UML diagrams (i.e., use case diagram, the domain model as a class diagram, the package diagram, the class diagram of the solution, activity diagrams, interaction and state diagrams). The UML diagrams have to be designed with a tool such as Rational Software Architect[7] (RSA). Furthermore, in the documentation they have to outline if they applied some specific architectural or design patterns. Whenever the project description was not clear or complete, the students were allowed to make assumptions described in the final documentation.

Within a week from the beginning of the project, the students were asked to send by email a Gantt diagram with an estimation of the project activities and duration. This Gantt diagram had to be updated during the project development according to the real effort needed by the project.

The final version of the project has been required to be executable and testable by the software engineering professors/tutors. This was mandatory for the evaluation of the project. Furthermore, the students had to submit the documentation about the installation and execution of the project. This means that, starting from the GitHub repository of a project, it was possible to build, setup, and execute the project without being aware of how it has been implemented.

To finalize the delivery of the project, one member of the each team had to send at the end of the project an email, “Delivery-email” with the name of the project, number of the team and names of the members, the GitHub TAG to be considered for the delivery, a pdf file with the documentation, i.e., the analysis and design of the project through UML diagrams. This file should have been attached to the email and also included in the GitHub repository. Other information related to the setting of the project (e.g., installation/execution) have to be included in this document and clearly outlined in the Delivery-email.

Further, each team had to send the updated Gantt diagram, with the effective effort spent to develop the software project. At the end of the project, the students received by email three different questionnaires on GitHub, SonarQube, and Microsoft Project to be filled and sent by email to the professors of the course.

Finally, after the evaluation of the projects (usually after one week from the delivery of the project by the students), each team sustained an oral exam session with the discussion on their project with questions or comments on how they used GitHub and SonarQube, questions on the documentation of the project (the UML diagrams), questions on the

[4] https://www.disco.unimib.it/upload/pag/46042/e3/e3101q119ingegneriadelswprogramma20162017.pdf

[5] https://www.disco.unimib.it/upload/pag/46026/e3/e3101q106programmazione2programma20162017.pdf

[6] https://www.disco.unimib.it/upload/pag/46028/e3/e3101q109analisieprogettazionedelswprogramma20162017.pdf

[7] https://www.ibm.com/developerworks/downloads/r/architect/index.html

design choices (design patterns or architectural patterns adopted) and on any other problem related to the project development or execution.

We would like to highlight that this software project is (1) the first project the students had to develop for an exam and (2) the first team project where they collaborated to reach a common goal: to pass an exam. We also would like to highlight that the final evaluations are individual reflecting the contribution of each student to the development of the project emerged during the oral session, on how they used GitHub and on the individual evaluations of the exercises assigned during the lab sessions. As mentioned also in Delgado et al. (2017) and Rajlich (2013), it is very important to adopt mechanisms enabling individual grades in team projects in order to maintain fairness.

3.1. How GitHub was used during the project development

Each team received a GitHub repository to manage the source code of the project. Each team member received access instruction to his/her repository. The students have been notified that they would be evaluated based on: the way they used GitHub for the code management, the use of branches, and the types of uploaded files. Each team member has been evaluated based on the way he or she has used the repository, i.e., not only considering the number of operations performed, but also the quality of them.

3.2. How SonarQube was used during the project development

We asked students to use SonarQube to evaluate and monitor the software quality of their projects. At the end of the project, before its delivery, they had to indicate the setting/configuration necessary to analyze their project with SonarQube and include the configuration files in the project (e.g., pom.xml and sonar-project.properties). Moreover, we asked the students to deliver the project with a rating level of A (the maximum) among the different indicators available in SonarQube and a project without critical violations, always according to the classification of the violations provided by SonarQube.

3.3. How Microsoft Project was used during the project development

We asked students to do a Gantt diagram at the beginning of the software project development with the estimation of the effort required. This initial diagram had to be sent by email to the professors during the first week of the project development. Further, the students had to update the Gantt diagram during the entire project and send to the professors the final version once they finished the project. We asked for these two versions in order to see how close their initial estimation was to the real effort.

4. Data collection

All the students who attended the course and sustained the exam have received a questionnaire with the questions specified in this section on GitHub, SonarQube, and Microsoft Project. Each student has received by email the questionnaire after the final version of the exam project has been submitted and before the discussion of the project with the software engineering professors. All the students have answered the questions.

4.1. Data collection for GitHub

The questions for the students concerning GitHub have been organized in three main groups:

- *general*, regarding the student knowledge about the existence of GitHub and its previous use;
- *technical skills*, regarding the student knowledge about GitHub functionality (e.g., shared repository model, pull requests);
- *communication and teamwork skills*, regarding the student knowledge on communication with the team members, tasks distribution).

For some questions we asked to provide a short motivation of their answers.

4.1.1. GitHub: general questions (GQ)

GQ1. Had you heard about GitHub before it has been mentioned during the software engineering courses? If yes, when and in which context?
GQ2. Have you used GitHub before it has been mentioned during the software engineering courses? If, yes, when and in which context?
GQ3. Which was your familiarity with GitHub before starting using it for the exam project (on a range from 1 to 5, 1[*low*], 2[*average*], 3[*good*], 4[*very good*], 5[*expert*])?
GQ4. Which is your familiarity with GitHub after finishing the exam project (on a range from 1 to 5, 1[*low*], 2[*average*], 3[*good*], 4[*very good*], 5[*expert*])?
GQ5. In your opinion using GitHub for the exam project is useful? If yes why, if not why?
GQ6. Do you think you will use GitHub further in other courses during your degree or for your final thesis?
GQ7. Do you think you will use GitHub in future projects (also for work projects)?
GQ8. In your opinion how much is GitHub used in industry (on a range from 1 to 5, 1[*not used*], 2[*seldom used*], 3[*used often*], 4[*used very often*], 5[*used in almost all companies*])?
GQ9. Do you know that companies may ask you to show them your GitHub account during interviews?

4.1.2. GitHub: technical skills questions (TSQ)

TSQ1. What operating system do you use?
TSQ2. Did you have problems installing the necessary tools for the project development? If yes, which tools and which problems?
TSQ3. Did you have problems using GitHub on your computer? If yes, which problems?
TSQ4. Which is, in your opinion, the most useful functionality of GitHub you have used? Why? When and for what did you use it?
TSQ5. Please provide examples of new (here "new" means not known before starting the project) functionalities of GitHub you discovered and used during the development of the exam project?
TSQ6. Which functionality would you add to GitHub to improve its usability?
TSQ7. Did you use the Pull Request mechanism of GitHub? When and why?
TSQ8. Did you use the Shared Repository Model of GitHub?
TSQ9. Do you think that having a history and a version control mechanism helped you in improving your programming skills (by learning from your mistakes done during the project development)?

4.1.3. GitHub: communication and teamwork questions (CTQ)

CTQ1. Did you read the code and documentation published by the other group members through GitHub for your exam project?
CTQ2. Do you think that the other group members have read the code and documentation you published?
CTQ3. Do you have any concern about the fact that your team members read your code and documentation?
CTQ4. Did you use notifications in GitHub? How did you manage them?
CTQ5. Do you consider that GitHub helped you in the management of tasks distribution in the team?
CTQ6. Do you consider that GitHub helped you in the

communication with the other members of the project team? How?
CTQ7. How often did you meet physically with your team members (e.g., once a day, once a week, once a month)?

4.2. Data collection for SonarQube

We proposed the students the following questions related to SonarQube. For some questions we asked to provide a short motivation of their answers.

SQ1. Have you found difficulties in using SonarQube? If yes, which ones?
SQ2. How many issues have you removed in your project?
SQ3. Have you found difficulties in removing the issues? If yes, which ones?
SQ4. How do you consider the usefulness of SonarQube with respect to software quality analysis? Low/Medium/High. Please briefly explain your answer.
SQ5. In a future context, during an academic project for another exam, do you think you would use SonarQube? Please briefly explain your answer.
SQ6. In a future context, in a real industrial project (when you will start working), do you think you would use SonarQube or will you suggest to use SonarQube? Please briefly explain your answer.
SQ7. Do you consider the rating provided by SonarQube (with the letters from A to E) useful to evaluate the quality of your project? They effectively reflect the quality to monitor how the project improves or not. (Yes/No). Please briefly explain your answer
SQ8. Which are, in your opinion, the principal drawbacks, if any, of SonarQube?
SQ9. What is it missing? Can you identify one or more functionalities or features not provided by SonarQube and useful for software quality analysis?

4.3. Data collection for Microsoft Project

We proposed the students the following questions related to Microsoft Project. For some questions we asked to provide a short motivation of their answers.

PM1. Had you heard about Microsoft Project before it has been mentioned during the software engineering courses? If yes, when and in which context?
PM2. Have you used Microsoft Project before it has been mentioned during the software engineering courses? If, yes, when and in which context?
PM3. Did you know what a Gantt diagram was before this project?
PM4. Did project management tasks help you in the organization of the teamwork?
PM5. Did project management tasks help you in the collaboration with the other team members?

5. Results

In this section, we summarize the student feedback on using GitHub, SonarQube, and Microsoft Project for the software project development. For each question, we indicate the most meaningful aspects of the provided answers.

5.1. Student feedback on the use of GitHub

5.1.1. GitHub: general questions (GQ)

GQ1: Considering the general questions on GitHub, 27 students out of 41 had heard about GitHub before attending the software engineering course. The students mentioned that they found out about the existence of GitHub:

- from other students (e.g., who work in software companies or heard or read themselves about it) (15 students);
- from reading on the Internet about sharing code, open source community, and Linux (3 students);
- by searching on the Internet for specific projects which were available on GitHub (7 students);
- by being involved in the development of a project outside the university by other software developers (2 students).

GQ2: From the 27 students who knew about the existence of GitHub, only 6 of them have used it. 2 of the 6 students have used GitHub for software development, while 4 of them for just saving the downloaded projects.

GQ3: Considering the familiarity of the students with GitHub after attending the software engineering course and before the development of the project for the exam, 25 students sustain that they had a low level knowledge about GitHub, 14 students had an average knowledge, while 2 students had a good knowledge. None of the students has evaluated his GitHub skills as very good or expert.

GQ4: After the development of the software engineering project, 1 student sustains that he had a low level knowledge about GitHub, 1 student had an average knowledge, 26 students had a good knowledge, while 13 had a very good knowledge. None of the students has evaluated his GitHub skills as expert. Except 1 student, all the others have improved their GitHub skills after the project development. This is true also for the student that sustains that after the project development, he has an average knowledge about GitHub. There is just 1 student who had not improved his GitHub skills during the project development. The motivation is based on the fact that the student has used just the basic functionality of GitHub due to his limited experience in managing branches and commits.

We notice also that the 2 students who used previously GitHub for projects development have improved their GitHub skills during the course project development. Therefore, we can summarize that the use of GitHub has been useful for all the students independently of their knowledge about GitHub before attending the software engineering course.

GQ5: This last affirmation is confirmed also by the student answers to question GQ5. Except 1 student, all the others have confirmed the usefulness of GitHub in the development of a project. The motivations indicated by the students are summarized as follows:

- the availability of the updated code always and from everywhere;
- the awareness of all team members about the changes in the project;
- the tracking of different versions of the code and the progress of the project;
- the improvement of the communication and coordination among the group members;
- the concurrent, simultaneous work on a common project;
- the use of a tool which is not very complex to understand.

The student who mentioned that the use of GitHub was not useful indicated that GitHub is appropriate for very large projects in industry and thus the professors may recommend its use, but not impose its use. As software engineering professors, we have imposed the use of tools such as GitHub, SonarQube, and Microsoft Project for two main reasons. First, we wanted to simulate the development and communications issues raised by a real software development project which is implemented in collaboration with many software engineers who may not be always in the same place. Hence, we wanted to bridge the gap between the academic and the industrial worlds. Second, we chose and imposed the above mentioned tools in order to be able to apply common and uniform evaluation mechanisms for all the students attending the course.

GQ6: 34 students out of 41 think that they will use GitHub further during their academic career (e.g., exams or final project). 7 students

think that they will not use GitHub until their degree indicating that their remaining courses and exams do not require the development of a software project.

GQ7: All the students, with no exceptions, mentioned that they will use GitHub in further software projects.

GQ8: The students answered that they think GitHub is used often (11 students), very often (20 students), or used in almost all companies (10 students). No one thinks that it is used seldom or not used at all. We interpreted these answers positively because we consider that students realized the importance and usefulness of a collaborative tool for software development. They become aware of the support provided by such a tool and in addition they are able to exploit the functionality (even at a basic or average levels) it provides.

GQ9: 17 students out of 41 think that companies may ask them about their GitHub account during interviews. 2 students were already asked about their GitHub accounts during interviews. One of them had already an account and was able to provide further information, while the other did not had an account at the interview time. 24 students became aware of the possibility of getting their GitHub accounts in the context of the software engineering course to be used for job interviews. We interpreted this as positive because in this way more students already became aware of GitHub and we are sure that they will inform other students about its existence, functionality, and advantages.

Fig. 1 summarizes the answers (in percentage) of the GQ for which the answers may be categorized in *Yes* and *No*, while Fig. 2 summarizes the answers (in percentage) of the GQ for which the answers may be categorized in a range from 1 (minimum) to 5 (maximum).

5.1.2. GitHub: technical skills questions (TSQ)

TSQ1: 37 students out of 41 use the Windows operating system. 2 students use MacOS, while 1 student Linux and 1 student Ubuntu.

TSQ2: 37 students out of 41 had no problems installing the necessary tools for the project development on their computers. The 4 students who had problems used the Windows platform. The problems concerned Maven, Spring libraries, and Hibernate. Hence, they were not concerned actually with GitHub directly. However, all the problems have been solved during the course labs before starting the project development.

TSQ3: 39 students out of 41 had no problems using GitHub on their computer. 2 students had difficulties when using merging functionality especially with jsp files and when pushing the project because they merged wrong files.

TSQ4: The students mentioned as the most useful functionalities of GitHub that they used during the software project development the following ones:

- commit messages to keep track of the work and to be notified by the changes made by the other team members (17 students);
- branching and merging functions because they allowed for the concurrent working of the team members and the management of eventual conflicts (17 students);
- versioning of code and recovering old versions of software easily, i.e., having the history of the project (3 students);
- the pull functionality, because it enabled students to download updated versions of code without losing their changes; it provides a fast integration among the team members (2 students);
- code browsing and organization (1 student);
- checking the code written by the other team members (1 student).

The first two listed functionalities were those often used by the students during the project development, and hence, students have mentioned them as the most useful.

TSQ5: The students mentioned as new functionalities, i.e., not used by themselves before:

- the pull functionality (6 students);
- the revert of a pull request (2 students);
- the issues functionality (1 student);
- none, they did not use any functionality which was not explained during the software engineering laboratories; what has been explained during the labs was enough to develop the project (27 students);
- every functionality was new for them (5 students).

From the last answer, i.e., all the functionality was new, we noticed that the question may have not been clearly understood by 5 students. We intended to ask students about new functionalities they discovered in GitHub during the development of the exam project (hence, after attending the software engineering course). Or, the 5 students have not attended all the classes of the software engineering course and hence, everything was new for them during the project development.

TSQ6: 27 students out of 41 have indicated no further functionality to add to GitHub. 3 students indicated as a new functionality the undo of the last commit, meaning that they were not able to discover by

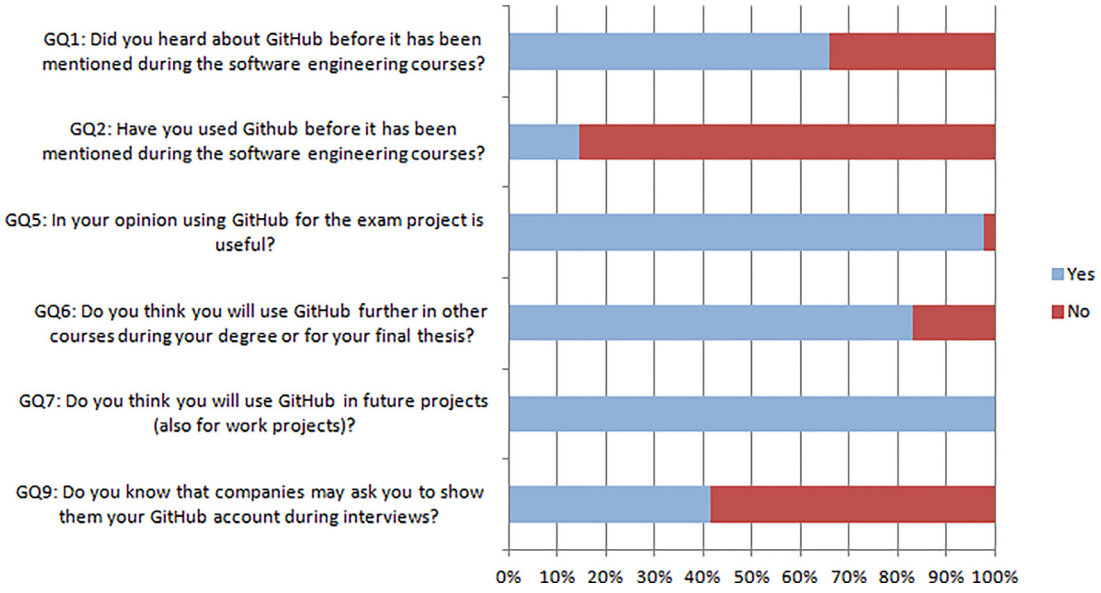

Fig. 1. GitHub: GQ responses for yes/no questions.

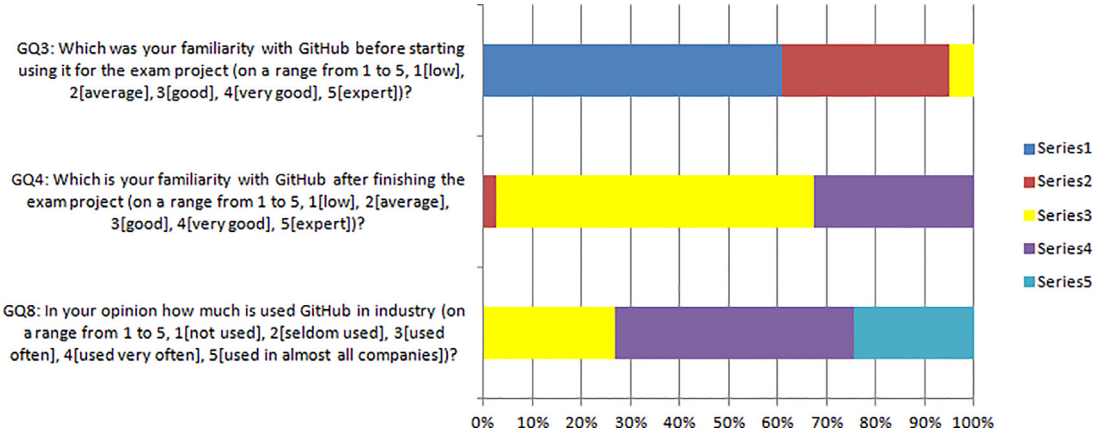

Fig. 2. GitHub: GQ responses for range [1–5] questions.

themselves the revert function.

11 students have indicated the improvement of the existing functionalities regarding:

- the commit history visualization and in general a clearer interface about team members contributions;
- the commit explanations;
- the pull operation;
- the merge and conflicts management;
- the user interface.

TSQ7: 38 students out of 41 had used the pull request mechanism of GitHub during the project development, to update their local repository and to be aware of the changes made by their team members. 3 students did not use the pull request. There may be 3 explanations of this answer. First, the students were not curious to know the work done by their team members, as far as each member did his part of the project. Second, the other team members contributed less to the project development and hence there was no need to update the local repository. Third, the students did not know the functionality and hence they preferred not to use it.

TSQ8: 40 students out of 41 indicated that they knew the shared repository model and used it. 1 student declared that he did not know it and hence did not use it. This answer is worrying because in the project development instructions, the students have been notified about a shared repository to be used among the team members. The only explanation for such an answer is that the student has not understood what a shared repository is and why it is used.

TSQ9: 36 students out of 41 considered that having a history and a version control mechanism in GitHub helped them in improving their programming skills by learning from their mistakes done during the project development. The students who answered no at this question also explained that they did not actually use this GitHub functionality; or, that this functionality helped them more to manage the project development.

Fig. 3 summarizes the answers (in percentage) of the TSQ for which the answers may be categorized in *Yes* and *No*.

5.1.3. GitHub: communication and teamwork questions (CTQ)

CTQ1: 39 students out of 41 read the code and documentation published by the other team members on GitHub motivating that it was fundamental for understanding the work on the project. The students who did not read the team members code and documentation did not provide any explanation.

CTQ2: 39 students out of 41 students think that the other team members read their code and documentation. The students who not read the other members code and documentation also think that the other students behaved in the same way.

CTQ3: 4 students out of 41 have some concerns about the fact that their team members read their code or documentation. This may be due to their limited experience in coding and software development.

CTQ4: 37 students out of 41 have used notifications during the software development. Students exploited notifications for pull requests and open issues. The students who did not used notifications mentioned that they used the fetch functionality of GitHub.

CTQ5: 32 students out of 41 consider that GitHub helped them in the management of the tasks distribution of their projects and clarify their contributions to the project development. The students who answered no to this question mentioned that they consider to be very important to meet each other to organize the teamwork. 2 students indicated that they expected more support from GitHub for the task distribution management. A student mentioned that the project was of limited dimensions, hence the benefits of task distribution through GitHub were not significant.

CTQ6: 27 students out of 41 consider that GitHub helped them in the communication with the other team members in the following ways:

- to see immediately the updates in the project done by the other team members, including the comments about the made changes to discuss eventually the performed changes;
- to synchronize their work;
- to track the information needed to work on their software parts.

The students who answered no to this question mentioned that they had used GitHub to merge their work, but used instead chats to communicate among them. In our opinion, students used chats for communication because they used chats before the project and hence, they continued with this already known and used mechanism for communication. Hence, it was more natural to communicate among them as they did before the project development. Some students mentioned that they used Telegram, because GitHub is configurable with Telegram[8]

CTQ7: 9 students out of 41 declared that they met their team members daily. 19 students met 2 or 3 times a week. 9 students met once a week. 4 students met only 3 times during the project duration.

Fig. 4 summarizes the answers (in percentage) of the CTQ for which the answers may be categorized in *Yes* and *No*.

5.2. Student feedback on the use of SonarQube

We provide below a synthesis of the answers we collected from the SonarQube questions. In our opinion some answers were provided quickly without much attention and some students did not answer all

[8] https://github.com/DrKLO/Telegram

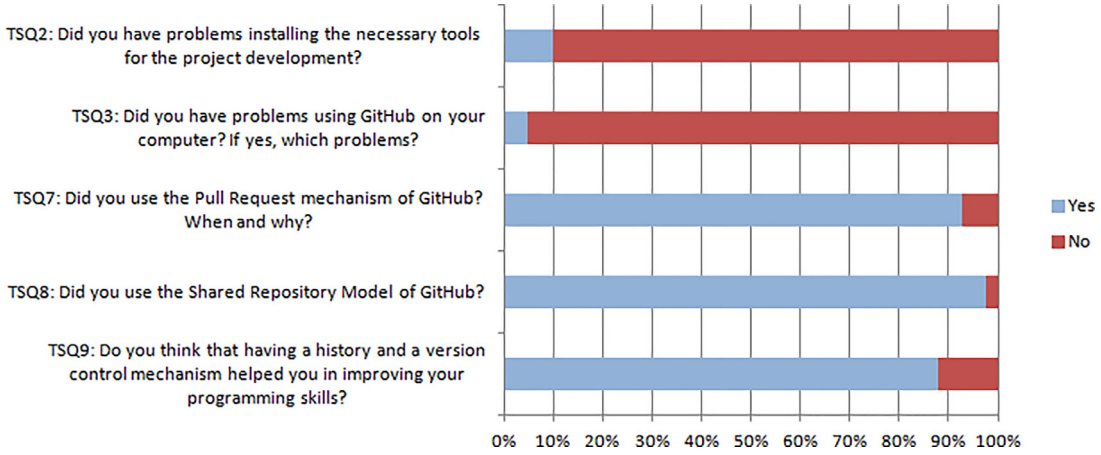

Fig. 3. GitHub: TSQ responses for yes/no questions.

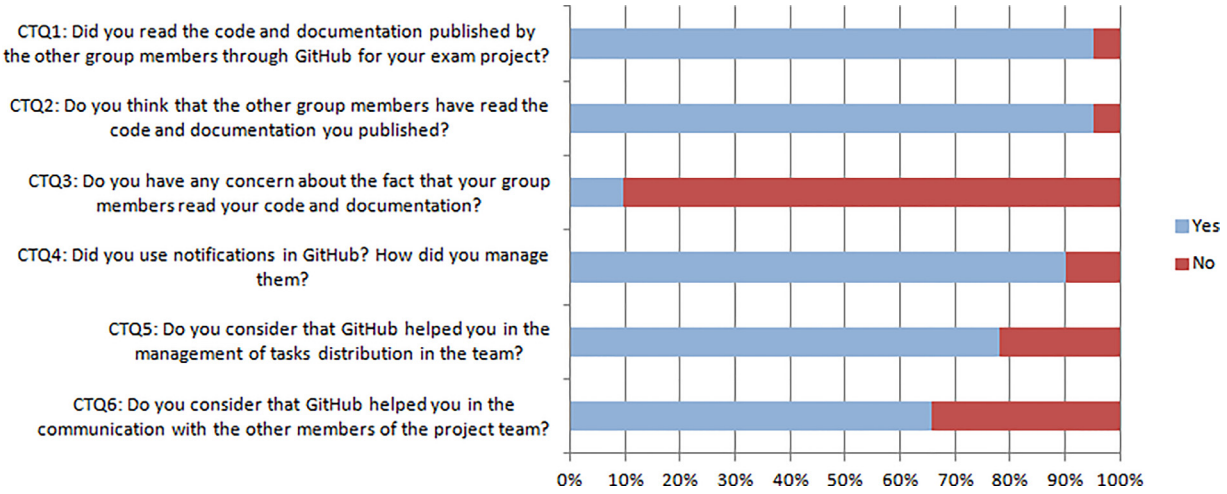

Fig. 4. GitHub: CTQ responses for yes/no questions.

the questions (SQ5 and SQ6).

SQ1: Most of the students (35 students out of 41) did not found any kind of difficulty in using SonarQube. However, they observed that the first configuration of the software may be a little tricky. A few students (6 students) found some difficulties on the quality profile settings and plug-in installation.

SQ2: Usually, all the students removed more than 100 issues with SonarQube, with an average between 100 and 150 revealed issues.

SQ3: All the students have found no difficulties in removing simple issues mostly related to the code style. Generally, they found difficulties to remove some issues related to a unique “big problem or a large part of the project” which needs too much time to be solved. Some students (12 students out of 41) met difficulties in finding the appropriate solution to some issue that were not always clear from the examples provided by SonarQube.

SQ4: With respect to the usefulness of SonarQube to analyze software quality, all the students considered SonarQube useful to write code in a proper manner, in particular during the development process, since SonarQube is a highly customizable tool that allows, if used correctly, to run many analyzes on many projects very quickly. They outlined that keeping the code clean could be an investment that pays on the long run of the software lifetime. But some students (18 students out of 41) did not find it useful to identify and solve more complex problems such as different code smells (Fowler, 1999), except for the detection of duplicated code.

SQ5: Generally, students (37 students out of 41) plan to keep using SonarQube for future software projects. They asserted that they will use SonarQube to analyze future software projects developed for other exams and for the third-year stage (e.g., Bachelor thesis). Four students did not answer this question.

SQ6: Generally, students (37 students out of 41) answered positively concerning the further use of SonarQube in real industrial projects. The motivations they indicated are: SonarQube is simple to use, it helps managing small issues which summed may lead to bigger problems in the future and it can facilitate a continuous integration process. Four students did not answer this question.

SQ7: On the rating provided by SonarQube to evaluate the quality of a software project, many students (30 students out of 41) did not consider it useful because the tool detects a lot of minor issues related to the code style. Once these minor issues are solved, the rating usually becomes “A” or “B”, while actually there may be still critical problems to be solved. They observed that having a single index is very good, but this should not be the only parameter used to assess software quality. Having partial indexes/metrics values could help understanding the code quality better.

SQ8: The main drawback of SonarQube indicated by the students (30 students out of 41) is related to the fact that the tool gives the developer the illusion of good code when actually it is not, since SonarQube provides too many code level issues and few design and architecture level issues (Arcelli Fontana et al., 2016a; Arcelli Fontana et al., 2016b; Garcia et al., 2009).

SQ9: With respect to the last question on SonarQube regarding what is missing and what may be the most useful functionalities or features to add to this tool for software quality analysis, the students mentioned

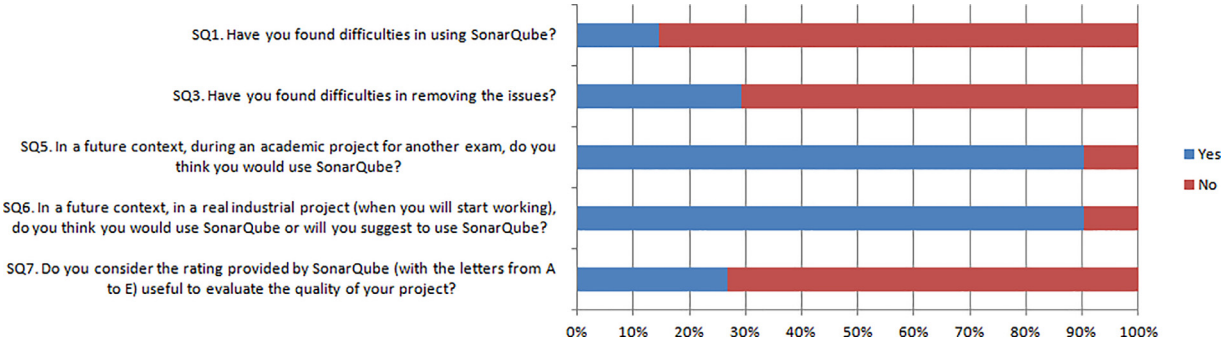

Fig. 5. SonarQube: SQ responses for yes/no questions.

that it should provide information about issues at design level. Architectural issues detection, as architectural smells (Garcia et al., 2009), would be a good addition feature to SonarQube. Moreover, the students observed that it could be relevant to have more useful refactoring advices to solve the problems.

Fig. 5 summarizes the answers (in percentage) of the SQ questions for which the answers may be categorized in *Yes* and *No*.

5.3. Student feedback on the use of Microsoft Project

We provide below a synthesis of the answers we collected from the Microsoft Project questions.

PM1. Half of the students (21 out of 41) have heard about Microsoft Project before the software engineering course during their work experience or from their university colleagues. Three of the students indicated that they discovered Microsoft Project from the Microsoft software installation package.

PM2. None of the students have used Microsoft Project before the software engineering course.

PM3. 4 out of 41 students knew what a Gantt diagram was before the software engineering course from their work experience. These students mentioned that they used other tools for the Gantt diagram such as Google Gantt charts or Excel Gantt chart templates.

PM4. Half of the students (20 out of 41) answered that project management tasks have helped them in the organization of their teamwork. The remaining 21 students mentioned that the project management related activities represented additional tasks to do and that for their teams were not fundamental for the organization of their work. However, they consider project management very important for larger projects than the one developed in the context of the software engineering course.

PM5. Less than 30% of the students (12 out of 41) mentioned that project management tasks helped them in the collaboration with the other team members in the initial phases for the tasks distribution.

Fig. 6 summarizes the answers (in percentage) of the PM questions.

6. Threats to validity

In this section we outline some threads to validity concerning our case study. First, the questionnaire may contain biased questions. To limit this aspect before sending the questionnaire to the students, we asked the opinion of two colleagues (one PhD student and one PostDoc) not involved in our course, but involved in other undergraduated and graduated software engineering courses. After this check, we changed some questions, without removing any question.

Second, the students received a questionnaire with three sets of questions concerning GitHub, SonarQube, and Microsoft Project. They had to answer the questions off-line, at home. Hence, if they had doubts on what and how to respond they had not the chance to ask for more details to overcome their doubts. Therefore, we plan to introduce also interviews to discuss the questions directly with the students.

Third, students had not been motivated to pay a particular attention on answering the questions (e.g., not counting for the evaluation of their work during the project development). Hence, we plan to stimulate them to consider seriously the answers to the questions.

Fourth, students had worked in teams. It is possible that they collaborated also to fill in the questionnaire. Students may have been influenced by other team members. Related to this aspect, some members of a team may have worked harder than the others also during the software development and quality assessment. However, this aspect is not transparent from the answers provided by the students. We have not considered the organization in teams when we analyzed the results, because each student received a questionnaire and had to fill it in by his or her own self.

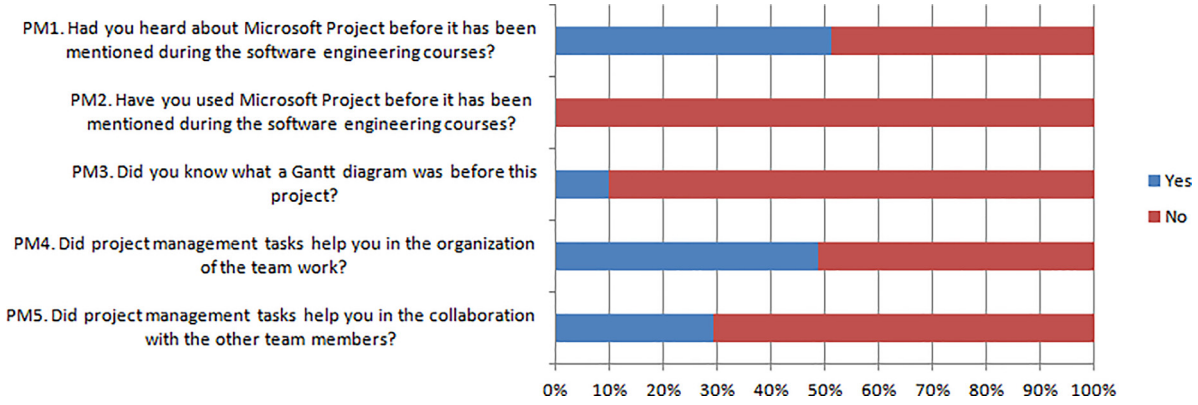

Fig. 6. Microsoft project: PMQ responses for questions.

Fifth, the case study has been applied once. Hence, we need to re-apply it to verify and eventually refine these initial findings and conclusions.

We describe in details in Section 8 the steps that we identified to minimize the impact of the limitations on the threats to validity of the current study.

7. Concluding remarks: learning outcomes

In this section, we analyze the student feedback on using GitHub, SonarQube, and Microsoft Project during the development of a software project in teams. Generally, we realized that the students had very limited knowledge on GitHub and Microsoft Project and no knowledge at all on SonarQube before they have been introduced in the software engineering course. We think that the knowledge and the skills they got during the course, in particular during the laboratory sessions and the project development for the exam, are very important for them, both for future projects in their academic path (e.g., Master courses) and in their future work in companies.

They all found useful the three tools. They realized the advantages one can get in using these tools. Some of the students were able to realize and identify also the drawbacks of these tools. Further details on these affirmations are provided in the following subsections, where we summarize the outcomes and the answers to the research questions introduced previously in the paper.

7.1. Lessons learnt on using GitHub

RQ1: What is the student feedback on the benefits of using GitHub for software development regarding collaboration and teamwork?

From the general questions on the use of GitHub we may summarize the following outcomes.

More than half of the students who heard about GitHub before attending the software engineering course discovered its existence through direct communications with other colleagues who knew GitHub. This means that collaboration and communication among students is very important and efficient for the improvement of their knowledge and skills. It also comes to confirm our choice to introduce GitHub as a collaborative software development platform and to encourage students to work together to prepare an exam.

Less than 5% of the students attending the software engineering course have used GitHub before as a collaborative software development platform. Hence, the students are appropriate candidates for the development of a software engineering project through GitHub.

Independently on their initial knowledge on GitHub, all students, except 1, have improved their skills in using this tool mostly during the development of the software project for the exam. Hence, the knowledge and skills acquired during the laboratories have been further improved by the development of the software project.

During the project development, students had the opportunity to realize the advantages offered by a tool such as GitHub. They plan to exploit these advantages in further projects in the academic or working context.

Furthermore, students became aware of the usage of GitHub in the industrial world. They also became aware of the requirement of having a GitHub account for job interviews. After this experience, the students get a GitHub account and can provide information about at least one software project already developed during their academic experience.

From the technical skills questions on the use of GitHub we may summarize the following outcomes.

Students used heterogeneous operating systems. They did not have major problems in installing the tools required for the project development on their computers. They also had no problems in exploiting GitHub. Hence, from the point of view on exploiting new tools, the students were prepared enough and ready to start for this experience.

Students used the main functionality of GitHub during the project development. Hence, they understood the objective, the functionality and the advantages of GitHub and used them properly. They also were able to indicate some improvement issues for this tool.

From the communication and teamwork questions on the use of GitHub we may summarize the following outcomes.

The students collaborated and communicated well during their project development through GitHub functionalities. They were interested in all the parts of the project even if they had not been actually involved in the implementation of all the parts.

Only a few students were concerned about the fact that the team members saw their contributions to the project. We concluded that the teamwork had encouraged these students to improve their development skills. In addition, they started to get used to these aspects of software development during their academic experience, and thus, be better prepared for their future jobs.

More than 75% of the students consider that GitHub helped them in the management of tasks distribution for their projects. Further, more than 65% of students consider that GitHub helped them in the communication with the other team members. We interpret these feedbacks as positive considering also that it is the first time, students have to manage by themselves such issues. In addition, they know very well each other and they also met physically very often, if not daily.

7.2. Lessons learnt on using SonarQube

RQ2: What is the student feedback on using SonarQube for software quality analysis?

From the point of view of the tool usage, students did not find particular difficulties in using SonarQube, even if it was the first time for all of them using this software quality analysis tool on a project. We outline that the students who attended the software engineering course did not know about the existence of this tool before.

From the point of view of the advantages provided by using such a tool for software quality analysis, the students realized that keeping the code clean through SonarQube is an investment that has significant advantages on the long term of the software lifetime. The students understood and used the functionalities provided by SonarQube. Therefore, they generally plan to use the tool on future projects during their academic experience or their future jobs.

From the answers to SQ7, SQ8, and SQ9, we can summarize that students were able to identify significant problems and drawbacks in using SonarQube, as listed in the following:

- the removal of issues related to a big part of the project or to a big problem could be a complex task;
- the identification and removal of various kinds of code smells (Fowler, 1999) could be difficult;
- the detection of design and architectural issues is missing;
- the rating does not reflect always the quality of the code; hence, the rating is not very useful since SonarQube detects a lot of minor issues related to the code style; after solving easily these minor issues, the rating may become very high, even if there may be still critical problems to be solved in the code.
- the need of more useful refactoring suggestions to solve some issues.

Furthermore, from the collaboration and teamwork point of view, SonarQube analyzed the output of the entire software project made by the contribution of all the team members. Hence, it helped students to maintain a uniform quality for all the software parts developed by each member.

7.3. Lessons learnt on using Microsoft Project

RQ3: What is the student feedback on the benefits of using Microsoft Project for project management regarding collaboration and teamwork?

From their feedback, it results that some of the students have understood the role and the benefits of the project management activities for their software development. However, many students have seen the project management activities as of secondary importance for their final objective. Students concentrated most on the development of the project and on its quality and less on its management.

In addition, students know each other very well and they met and worked together very often. Hence, they interacted directly most of the time. Mainly, they organized and coordinated themselves informally using also chat applications.

7.4. Concluding remarks

As a conclusion of our experience, we strongly recommend the development of a software project in teams counting as an exam and the use of collaborative development tools in software engineering courses for undergraduate students. Students are enthusiastic to learn and experience new approaches and tools in software development. The more they are stimulated, the more they learn. They were able to capture the advantages of these tools, and in the same time they were able to identify the limitations of the used tools. Students understood the objective of the tools and the advantages in using them. In conclusion, we recommend the introduction of the tools used during the courses. In this way professors transmit the appropriate message to the students and the students do not get bored and lost in the attempt to understand alone the tools.

According to our objective outlined in the Introduction on the learning process improvement based on the student feedback, we get many useful hints to enhance the teaching and the learning practices of the tools. We will pay particular attention to the comments of the students outlining problems and to the aspects or functionalities that require deeper explanations. Few students did not capture the concrete advantages in using these tools, but also according to this aspect more attention from the professors is needed.

8. Future work: lessons learnt for a future course case study

We would like to perform the case study described in this paper in the next software engineering course in our university. According to our previous experience described in this paper, we will consider the following directions to improve the case study.

- From the point of view of the collaboration and teamwork through the use of tools, we aim to better explain some complex GitHub functionalities (e.g., management of task distribution and communication in GitHub) during the lab sessions. Based on the student feedback, we observed that such functionalities were not captured in the appropriate way, since not all the students understood or used them during the project development. The same objective will be considered for SonarQube, too. We will also try to emphasize more the role and benefits of the project management activities during the project development and ask students to choose a project manager for each team.
- From the point of view of the tool for software quality analysis, with respect to SonarQube we would like to add new questions, to monitor the number of issues found during the project development, and to ask the students to report not only the number of issues solved, but also the number of issues existing in the project at the delivery of the project. Further, we will consider the most critical/difficult issues not solved to improve the lab sessions also according to this important aspect. The refactoring of some parts of the code could be really a complex task. Moreover, we plan to introduce other tools for software quality analysis to be explained during the lab sessions and to be used during the project development. Among these tools we may consider a tool for metrics computation and for getting different views on the project as for example those provided by Understand[9] (e.g. dependency graphs, control flow and hierarchy graphs). These views can be useful to capture different problems in the project design and implementation. Since SonarQube considers essentially code violations, we plan to teach students how they may detect not only code anomalies, but also architectural violations through a tool such as for example Arcan (Arcelli Fontana et al., 2016a) developed for architectural smell detection. In this way, a drawback identified by some students according to SonarQube (SQ8 and SQ9) may be faced and explored by exploiting a different tool.

  Finally, if we will introduce another tool, such as for example Understand, we may define a framework and ask the students to compare the tools according to different features useful to develop and monitor their projects. According to the answer to SQ7, since the students did not find very useful the rating provided by SonarQube, we can consider tools providing different rating or quality indexes on the analyzed projects (Arcelli Fontana et al., 2016b; Roveda et al., 2018) and check their usefulness.

- From the questionnaire point of view, we will try to revise the questions, i.e., add further questions, or re-phrase the existent ones in order to make them more clear. Furthermore, we plan to encourage students to answer the questions by paying the maximum attention. According to this aspect, we will ask the students to answer the questionnaire during the oral session of the exam, before starting the discussion on their project. In this way, the students cannot be influenced by each other and probably they will answer all the questions by paying more attention (being in front of their professors). Moreover, we plan to assign a weight (in the range of [0, 2] points) to the questionnaire to be included in the final evaluation. The final evaluation for each student currently includes: an individual evaluation of the exercises assigned to each student during the lab sessions (in the range of [0, 4] points), the evaluation of the documentation on the analysis and design of the project through UML diagrams (in the range of [0–6] points), the evaluation of the project considering also how they used GitHub, SonarQube, and Microsoft Project and the overall evaluation of the project including its execution and usability (in the range of [0–20] points). We may also consider the possibility to introduce interviews as those described in Runeson et al. (2012).
- From the point of view of the evaluation of team performance, we plan to investigate the teamwork quality intended as the quality of interactions among the team members. As indicated in Lindsjørna (2016) there are six aspects which may be considered for the evaluation of teamwork quality: communication, coordination, balance of member contribution, mutual support, effort, and cohesion.

  Communication may be analyzed through the frequency of the communication among the team members and the formalization of the information exchange. The current version of the questionnaire contains several questions, i.e., CTQ1 - CTQ7 and PM5 concerning the communication among the team members. As results from their feedback, students met and communicate often mostly through an informal exchange of information.
  Coordination may be seen through the management of the project activities and the dependencies among them. In the questionnaire, several questions, i.e., CTQ4-CTQ5 and PM4 concern the coordination among team members. As results from their feedback, 90% of the students used GitHub notifications, 78% of the students consider that GitHub is useful for the management of tasks distribution, and 50% of the students confirm that Microsoft Project helped them in the organization and coordination of their teamwork.

[9] https://scitools.com/feature-category/code-knowledge/

The *balance of the team members contribution* may be analyzed through the knowledge and experience of the team members regarding the project development. From this point of view, we consider that all the students have similar background and experiences, thus they all participated actually and actively in the decision-making process. In addition, they know each other well and they formed the teams by their own. These affirmations motivate also the achievement of the *mutual support* aspect, which may be analyzed through the tight collaboration among the team members.

The *effort* and *cohesion* aspects may be analyzed by the ability of the team to work together in time to achieve the project objectives. From this point of view, the students had a common goal: the exam. Thus, they were all committed to the team tasks and objectives. The cohesion should have been assured by the fact that students have chosen their team members.

As a future work, we plan to investigate further the team performances and gather also statistical data in order to see the evolution of the teamwork quality and achievements.

References

Arcelli Fontana, F., Pigazzini, I., Roveda, R., Zanoni, M., 2016a. Automatic detection of instability architectural smells. In: Proceedings of the 32nd International Conference on Software Maintenance and Evolution (ICSME 2016) - ERA Track. Raleigh, North Carolina, USA. pp. 433–437. https://doi.org/10.1109/ICSME.2016.33. October 2-10, 2016.

Arcelli Fontana, F., Pigazzini, I., Roveda, R., Zanoni, M., 2016b. Technical debt indexes provided by tools: a preliminary discussion. In: *Proceedings of the 8th IEEE International Workshop on Managing Technical Debt*, (MTD 2016). Raleigh, NC, USA. pp. 28–31. https://doi.org/10.1109/MTD.2016.11. October 4, 2016.

Arcelli Fontana, F., Raibulet, C., 2017. Students' feedback in using github in a project development for a software engineering course. In: Proceedings of the 22nd Annual Conference on Innovation and Technology in Computer Science Education. Bologna, Italy. pp. 380. https://doi.org/10.1145/3059009.3072984. July, 3rd-5th, 2017.

Bai, Y., Chen, L., Yin, G., Mao, X., Deng, Y., Wang, T., Lu, Y., Wang, H., 2017. Quantitative analysis of learning data in a programming course. In: Bao, Z., Trajcevski, G., Chang, L., Hua, W. (Eds.), Database Systems for Advanced Applications 10179 Springer, Cham. https://doi.org/10.1007/978-3-319-55705-2_37. DASFAA 2017. Lecture Notes in Computer Science.

Car, Ž., Belani, H., Pripužić, K., 2007. Teaching project management in academic ICT environments. In: Proceedings of EUROCON 2007 The International Conference on "Computer as a Tool". Warsaw, Poland. https://doi.org/10.1109/EURCON.2007.4400500. September 9-12.

Cosentino, V., Cánovas Izquierdo, J.L., Cabot, J. 2017. A systematic mapping study of software development with GitHub. IEEE Access, 5, DOI: 10.1109/ACCESS.2017.2682323.

de Andreade Gomes, P.H., Garcia, R.E., Spadon, G., Medeiros Eler, D., Olivete, C., Messias Correia, R.C., 2017. Teaching software quality via source code inspection tool. In: Proceedings of the Frontiers in Education Conference (FIE). Indianapolis, IN, USA. https://doi.org/10.1109/FIE.2017.8190658. October 18-21th, 2017.

Delgado, D., Velasco, A., Aponte, J., Marcus, A., 2017. Evolving a project-based software engineering course: a case study. In: Proceedings of the 30th Conference on Software Engineering and Education. Savannah, Georgia. pp. 77–86. https://doi.org/10.1109/CSEET.2017.22. November 7-9.

Ernst, N.A., Bellomo, S., Ozkaya, I., Nord, R.L., 2017. What to fix? distinguishing between design and non-design rules in automated tools. In: Proceedings of the 2017 IEEE International Conference on Software Architecture. Gothenburg, Sweden. pp. 165–168. https://doi.org/10.1109/ICSA.2017.25. April, 3rd-7th, 2017.

Falessi, D., Kruchten, P., 2015. Five reasons for including technical debt in the software engineering curriculum. In: Proceedings of the 2015 European Conference on Software Architecture Workshops. Dubrovnik, Cavtat, CroatiaNew York, NY, USA. ACM. https://doi.org/10.1145/2797433.2797462. September 7 - 11, 2015.

Feliciano, J., Storey, M.-A., Zagalsky, A., 2016. Student experiences using GitHub in software engineering courses: a case study. In: Proceedings of the IEEE/ACM 38th International Conference on Software Engineering Companion). Austin, Texas, USA. pp. 422–431. https://doi.org/10.1145/2889160.2889195. May 14-22, 2016.

Fowler, M., 1999. Refactoring: Improving the Design of Existing Code. Addison-Wesley, Boston, MA, USA.

Francese, R., Gravino, C., Risi, M., Scanniello, G., Tortora, G., 2015a. Using project-based-learning in a mobile application development course-an experience report. J. Visu. Lang. Comput. 31, 196–205. https://doi.org/10.1016/j.jvlc.2015.10.019. Dec 2015.

Francese, R., Gravino, C., Risi, M., Scanniello, G., Tortora, G., 2015b. On the experience of using Git-Hub in the context of an academic course for the development of apps for smart devices. In: Proceedings of the 21st International Conference on Distributed Multimedia Systems. Vancouver; Canada. pp. 292–299. https://doi.org/10.18293/DMS2015-003. 31 August - 2 September 2015.

Garcia, J., Popescu, D., Edwards, G., Medvidovic, N., 2009. Toward a catalogue of architectural bad smells. In: In: *Proceedings of the 5th International Conference on the Quality of Software Architectures (QoSA 2009)*. East Stroudsburg, PA, USA. pp. 146–162. https://doi.org/10.1007/978-3-642-02351-4_10. June 24-26, 2009.

Garcia, R.E., Messias Correia, R.C., Olivete, C., Costacurta Brandi, A., Marques Prates, J., 2015. Teaching and learning software project management: a hands-on approach. In: 2015 IEEE Frontiers in Education Conference (FIE). El Paso, Texas, USA. https://doi.org/10.1109/FIE.2015.7344412. October, 21-24, 2015.

Groth, D.P., Hottell, M.P., 2006. Designing and developing an informatics capstone project course. In: *Proceedings of the 19th Conference on Software Engineering Education and Training* (CSEET'06. Turtle Bay, HI, USA. https://doi.org/10.1109/CSEET.2006.15. April 19-21, 2006.

Kilamo, T., Hammouda, I., Chatti, M.A., 2012. Teaching collaborative software development: a case study. In: Proceedings of the 34th IEEE International Conference on Software Engineering. Zurich, Switzerland. pp. 1165–1174. https://doi.org/10.1109/ICSE.2012.6227026. June, 9th, 2012.

Kosti, M.V., Ampatzoglou, A., Chatzigeorgiou, A., Pallas, G., Stamelos, I., Angelis, L., 2017. Technical debt principal assessment through structural metrics. In: Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications. Vienna, Austria. pp. 329–333. https://doi.org/10.1109/SEAA.2017.59. August 30th - September 1st, 2017.

Lauvås, P, 2015. Analyzing student code after introducing portfolio assessment and automated testing. NOKOBIT 23 (1) November 2015, ISSN: 1894-7719. https://ojs.bibsys.no/index.php/Nokobit/article/view/286.

Letouzey, J.L., 2012. The SQALE method for evaluating technical debt. In: Proceedings of the 3rd International Workshop on Managing Technical Debt. Zurich, Switzerland. https://doi.org/10.1109/MTD.2012.6225997. June, 5th, 2012.

Lindsjørna, Y., Sjøbergab, D.I.K., Dingsøyrbc, T., Bergersena, G.R., Dybå, T., 2016. Teamwork quality and project success in software development: a survey of agile development teams. J. Syst. Softw. 122, 28–274. https://doi.org/10.1016/j.jss.2016.09.028. December 2016.

Ludewig, J., Bogicevic, I., 2012. Teaching software engineering with projects. In: Proceedings of the First International Workshop on Software Engineering Education Based in Real-World Experiences. Zurich, Switzerland. pp. 25–28. https://doi.org/10.1109/EduRex.2012.6225701. June, 9th, 2012.

Moreno, A.M., Sánchez-Segura, M.-I., Peters, L., Araujo, J., 2016. Enriching traditional software engineering curricula with software project management knowledge. In: 38th International Conference on Software Engineering Companion. Austin, TX, USA. pp. 404–410. https://doi.org/10.1145/2889160.2889193. May, 14th-22nd, 2016.

PMBOK, 2013. A Guide to the Project Management Body of Knowledge, Fifth Edition. Project Management Institute.

Rajlich, V., 2013. Teaching developer skills in the first software engineering course. In: ICSE '13 Proceedings of the 2013 International Conference on Software Engineering. San Francisco, CA, USA. pp. 1109–1116. https://doi.org/10.1109/ICSE.2013.6606661. May 18-26, 2013.

Rong, G., Gu, S., Zhang, H., Shao, D., 2017. DevOpsEnvy: an education support system for DevOps. In: Inç Proceedings of 30th IEEE Conference on Software Engineering Education and Training. Savannah, GA, USA. https://doi.org/10.1109/CSEET.2017.17. November 7-9, 2017.

Roveda, R., Arcelli Fontana, F., Raibulet, C., Zanoni, M., Rampazzo, F., 2017. Does the migration to github relate to internal software quality? In: 12th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2017. Porto, Portugal. pp. 293–300. https://doi.org/10.5220/0006367402930300. April, 28th-29th, 2017.

Roveda, R., Fontana, Arcelli, F., Pigazzini, I., C., Zanoni, M., 2018. Towards an architectural debt index. In: to appear in the IEEE Proceedings of the Euromicro Conference on Software Engineering and Advanced Application. Prague. August 2018.

Runeson, P., Host, M., Rainer, A. Regnell, B. 2012. Case study research in software engineering: guidelines and examples. John Wiley & Sons, 2012.

Silva, D., Nunes, I., Terra, R., 2017. Investigating code quality tools in the context of software engineering education. In: Computer Applications in Engineering Education. 25. pp. 230–241. https://doi.org/10.1002/cae.21793. Feb. 2017.

Stoica, A.-I., Islam, S., 2012. Educational methods for software and systems development. In: Proceedings of the 15th International Conference on Interactive Collaborative Learning (ICL). Villach, Austria. https://doi.org/10.1109/ICL.2012.6402127. September 26-28th, 2012.

Tirkey, A., Gary, K.A., 2017. Curricular change management with git and drupal: a tool to support flexible curricular development workflows. In: *Proceedings of the 15th International Conference on Software Engineering Research, Management and Applications* (SERA 2017. London, UK. https://doi.org/10.1109/SERA.2017.7965734. June 7-9.

van der Duim, L., Andersson, J., Sinnema, M., 2007. Good practices for educational software engineering projects. In: Inç Proceedings of the 29th International Conference on Software Engineering (ICSE} 2007). Minneapolis, MN, USA. pp. 698–707. https://doi.org/10.1109/ICSE.2007.40. May 20-26, 2007.

Yli-Huumo, J., Maglyas, A., Smolander, K., 2016. How do software development teams manage technical debt? - an empirical study. J. Syst. Softw. 120, 195–218. https://doi.org/10.1016/j.jss.2016.05.018. October 2016.

Zagalsky, A., Feliciano, J., Storey, M.-A., Zhan, Y., Wang, W., 2015. The emergence of GitHub as a collaborative platform for education. In: Proceedings of the 18th ACM Conference on Computer-Supported Cooperative Work. Vancouver, BC, Canada. pp. 1906–1917. https://doi.org/10.1145/2675133.2675284. March 14-18th, 2015.

Zakiah, A., Fauzan, M.N., 2016. Collaborative learning model of software engineering using GitHub for informatics student. In: Proceedings of the International Conference on Cyber and IT Service Management. Bandung, Indonesia. https://doi.org/10.1109/CITSM.2016.7577521. April 26-27, 2016.

Claudia Raibulet is an Assistant Professor at the Universitá degli Studi di Milano-Bicocca in Italy. She received her master degree in Computer Science from POLITEHNICA University of Bucarest, Romania in 1997 and her PhD degree from Politecnico di Torino, Italy in 2002. She is involved in Software Engineering and Reverse Engineering courses. Her research interests concern various software engineering areas including software architectures, object-oriented methodologies, model-driven solutions, development of adaptive systems, mobile systems, distributed systems, reverse engineering, software architecture reconstruction, design pattern detection, and software engineering education. Claudia Raibulet co-authored more than eighty research papers published in international journals, conferences, and workshops. She is involved in referee activities for various international journals, as well as in organizing and program committees for international conferences and workshops.

Francesca Arcelli Fontana has her degree and Ph.D in Computer Science taken at the University of Milano, Italy. She is currently in the position of Associate Professor at University of Milano-Bicocca. The actual research activity principally concerns the software engineering field. In particular in software evolution, software quality assessment, model-driven reverse engineering, managing technical debt, design patterns and architectural smells detection. She is the head of the *Software Evolution and Reverse Engineering* Lab (http://essere.disco.unimib.it/) at University of Milano-Bicocca and she is a Member of the IEEE Computer Society.