Applying GitHub Services to Support Teaching-learning Strategies in Computer Science Courses

Manel Mena[a], Javier Criado[b], Isabel M. del Águila[c], Joaquín Cañadas[d] and Luis Iribarne[e]
Department of Informatics, University of Almería, Spain

Keywords: GitHub, Classroom, Feedback, Automation, Tests, Teamwork Collaboration.

Abstract: Computer Science students need to acquire both theoretical and practical use of the knowledge that is covered by the subjects or courses of their degrees. However, sometimes they attend lectures about theoretical concepts that they cannot apply with real development tools that are part of the industry. Therefore, we believe that students have to start working as soon as possible in a similar way to the one they will use in their work once they have finished their studies, considering each course or each deliverable activity as a software development project. This paper describes the solutions adopted and applied to five Computer Science courses in two innovation teaching projects of the University of Almería. Furthermore, we present a series of support teaching tools for managing and creating GitHub projects as GitHub is the core technology for developing teaching-learning activities because of its widespread use. Each course manages its own strategy according to its specific characteristics (i.e., learning objectives, number of students, schedule or programming languages).

1 INTRODUCTION

Computer Science courses should provide not only both solid theoretical and practical foundations in many knowledge areas, but also teamwork to prepare the students for their careers (Gutica, 2018). Besides, with the demands of technological advances in software, it is required that software development teams work in a collaborative way, skills that every student must learn from the very beginning in their Computer Science courses (CC2020 Task Force, 2020).

Nowadays, one of the most widely used cloud services to store collaborative software artifacts is GitHub. GitHub is an online repository support platform for managing projects and controlling code versions, reaching the social network level designed for developers because millions of people worldwide use its services to cooperate on it. It offers distributed version control and source code management functionalities, plus its own features and integrations with many external applications. It also provides access control and several collaborative services such as feature requests, tasks management, code merging or continuous integration.

We believe that the students should start as soon as possible to work with this kind of platform daily, that is like they are going to work as engineers once they finish their studies. This work shows the solutions adopted for the embedding of GitHub in the learning processes of several courses for the Computer Science discipline at the University of Almería (UAL). Each one manages their own strategy and takes advantage of the specific features of GitHub according to the semester it is scheduled and the particular learning objectives defined for the course.

The goal of this work is to develop strategies and methodologies based on GitHub where the steps followed by the students in the execution of the different courses activities are linked with the steps of a software development process. The approach is under evaluation, and the proposed strategies have been instantiated in the courses described in this article from different perspectives. In some courses, the maturity level in the use of GitHub services is higher than in others, as well as the number of applied features. Thanks to the involved innovation teaching projects, an effort is being made to unify the strategies establishing a common methodology.

[a] https://orcid.org/0000-0003-1084-8489
[b] https://orcid.org/0000-0002-8035-5260
[c] https://orcid.org/0000-0001-9896-7196
[d] https://orcid.org/0000-0001-5391-965X
[e] https://orcid.org/0000-0003-1815-4721

Mena, M., Criado, J., Águila, I., Cañadas, J. and Iribarne, L.
Applying GitHub Services to Support Teaching-learning Strategies in Computer Science Courses.
DOI: 10.5220/0011071000003182
In *Proceedings of the 14th International Conference on Computer Supported Education (CSEDU 2022) - Volume 1*, pages 289-296
ISBN: 978-989-758-562-3; ISSN: 2184-5026

2 GitHub SERVICES FOR CLASSROOM MANAGEMENT

Nowadays, GitHub is the most popular Web-based social code sharing service in the software industry, and many companies in technology areas use it[1]. It is not only used to develop software and to write technical documentation, but also to collaborate in a variety of domains (Feliciano et al., 2016).

One of these soaring knowledge-area that exploits GitHub services is education, specifically for Computer Science courses that involve software development (Zagalsky et al., 2015; Tushev et al., 2020) and programming (Angulo and Aktunc, 2019; Glazunova et al., 2021), but other disciplines have also deployed successful solutions (Fiksel et al., 2019; Nelson and Ponciano, 2021).

Just from the boost of WWW in the mid-nineties, online facilities had been applying to support teaching. Those software tools allowed teachers to host and distribute the material to their students and facilitated interrelationships through emails and forums threads. Learning Management Systems (LMS) such as Canvas, Moodle or Blackboard were groundbreaking tools still widely used today. These tools established a new milestone in education because they provide instructors with various services for managing and administering courses.

The features offered by LMS are evaluating students, evaluating courses and instructors, creating class discussions, creating computer-based instruction, and so on (Malikowski et al., 2007). However, while these tools allow communication between teachers and students, they do not promote collaboration because both roles are separated. GitHub goes a bit further because besides enabling future professionals to learn how to manage and collaborate on their software development projects, its services allow teachers to define innovative teaching-learning processes that can be customized according to specific characteristics of their courses. We have defined five categories of supporting areas that have been instantiated in the different courses described in Section 3.

2.1 Supporting Shareable Material

The most usual approach to sharing course contents in this kind of platform is to include the documents and material in a specific repository giving access to the enrolled students.

Public repositories offer the possibility to share content publicly without any access restrictions. On the other hand, private repositories enable sharing contents only to enrolled students, to students and teachers, or only between teachers, giving desired users the proper access to the repository.

However, were an entire repository to be shared, for instance, a software tool, this approach would not be valid. A better solution is to define an organization that could support the different materials arranged in several repositories. The definition of an organization also facilitates the use of GitHub Classroom.

Besides the definition of an organization to support a given course, multiple teaching resources and reference material on can be shared using GitHub. Specifically, the feature called GitHub Pages makes it easy and free to publish a static website hosted as ". github.io". This feature has been the starting point to release several teaching resources to promote the use of serious games[2] or to share coding problems as learning objects (Águila et al., 2021).

2.2 Project Management

GitHub collaborative features can be leveraged during the entire life cycle of a project. One of the most critical stages in software development is project management, and GitHub provides the Projects view for this goal. For example, it is possible to create project boards with a Kanban style (Malakar, 2021) and link the task written on the board with issues related to project development. Furthermore, issues can be described with additional information (e.g., relevance, priority, or related milestones) and assigned to specific collaborators. Finally, the status of the task and issues can be tracked and managed.

This way, project management activities can be applied to support the activities assignment, the tracking and monitoring of the exercises, as well as the analysis of the learning objectives that users are achieving (e.g. using the milestone concept).

2.3 Collaborative Working

Once the project scope has been identified in the management stage, each functionality will be managed and implemented (i.e., solved). In a teaching-learning process, this work execution can be analogous to the moment the students build the solutions to the problems presented. In Computer Science courses, the collaborative work includes the development of different artifacts involved in the course, for instance, the specific solutions or reports for students' assignments or the software components that are part of a software application.

[1]GitHub Octoverse – https://octoverse.github.com/

[2]Serious Games Repo – https://cutt.ly/TIO98SI

Some exercises have to be solved individually, but others require forming teams or working groups that can solve a problem collaboratively. In this sense, GitHub provides the necessary infrastructure, allowing (*i*) assignment of issues to several students, (*ii*) repository sharing, (*iii*) forks to build alternatives, (*iv*) merging mechanism to gather partial solutions from different students, and (*v*) reporting of contributions by author, among other possible features.

2.4 Guidelines and Patterns

When students start to develop a solution for an exercise, especially when it is related to courses of the first academic years, they need to receive frequent feedback and specific information about the exercise they are solving. Therefore, it is not enough to make general and theoretical contents available for them, because their application could be difficult.

GitHub provides two approaches that can be used for this purpose. The direct strategy is related to creating help content in the form of incomplete solutions or templates that can and should be reused by students. Teachers can create repositories to provide these templates so that users can clone them and have this set of guides available to them. Another option is for students to start directly by forking these repositories, then developing their specific solutions and pushing them in a new repository or as a pull request of the original one.

The indirect strategy relies on the application of continuous integration and continuous deployment (CI/CD) processes to run validation tests on the material pushed to the repositories. The application of automated testing helps the students progress and enables early error detection. Moreover, the feedback information provided from the test executions can be used as the guidelines to solve the corresponding tasks, and the guidelines define a set of good practices that help transmit (teachers) and acquire (students) patterns related to software development. This strategy can be achieved using different alternatives such as GitHub Classroom autograding, GitHub Actions, Jenkins, or GitLab CI/CD.

2.5 Support Teaching Tools

In Section 2, we have identified the different ways in which we use GitHub. To support those different purposes, we have developed an application (ITSI-GITSI) for the generation and automatic management of course repositories on GitHub that allows us to perform a series of repetitive tasks around functionality offered by GitHub. Among the features of the application, we have the following:

```
1  apiVersion: 1
2  op: create
3  org: ualits
4  metadata:
5    repos:
6      -
7        name: "test2"
8        users:
9          - "javicriado"
10         - "manelme"
11       private: false
12   labels:
13     -
14       name: "1"
15       description: "low difficulty issue"
16       color: "ededed"
17   milestones:
18     -
19       title: "FirstMilestone"
20       description: "A milestone for the repository"
21   issues:
22     -
23       title: "Pruebal test1"
24       description: "Issue description"
25       labels:
26         - "1"
```

Listing 1: Config file for repository creation.

- Automatic generation of students' and teachers' repositories in a specific organization.
- Bulk generation of repositories using a .csv file.
- Generation of issues and milestones related to course assignments, either when creating a repository or a posteriori.
- Calculator of scores for the users of the repositories, taking into account the issues resolved, either simply counting one by one the issues or using possible score tags of the particular issue.
- Automatic modification of a markdown file to capture the scores.
- Fast deletion of multiple repositories.

This application uses some configuration files to execute the different named functionalities . The tool executes the configuration file passed in the parameter. This file has to be in `.yml` format. In it, we establish the functionality to be executed and the configuration parameters for that functionality, using the paradigm known as configuration as code (CaC) (Rahman et al., 2018). As an example, the Listing 1 reflects the proposed configuration for the creation of a repository. In the listing, we can see how in line 2, we establish the operation, in this case, create. In line 3, we define the organization where the repositories will be created. A posteriori, we enter the metadata of each of the repositories that will be created (lines 4-26). Moreover, inside this metadata, we define the repositories themselves (lines 5-11), the possible labels (lines 15-16), the milestones of the project (lines

17-20) and the issues to which we can assign the labels and milestones previously created. In the ITSI-GITSI[3] public repository, we defined a series of examples with the possible operations to be executed, as well as the requirements of the application.

3 EXPERIENCE IN COURSES

As part of the work of the teaching groups, the strategies described in Section 2 are being applied in different Computer Science courses. Table 1 shows the strategies applied in the involved courses (from several degrees and semesters). This section describes the specific actions carried out for each of these courses and analyzes some of the results obtained.

3.1 Software Engineering

The course Software Engineering allowed us to apply a different perspective to the common use of a GitHub repository. Normally, a repository is intended for version control of the source code files of a software system. However, in the Software Engineering course, this type of source file is not needed because the main objective is to teach students to model software systems using Unified Modeling Language (UML) (Seidl et al., 2015) use case, class and sequence diagrams.

With this aim, students receive the system specifications that they have to model, and build the corresponding UML diagrams. From these diagrams, students generate the UML models in an XML-based format that can be managed by the version control of GitHub. In addition, the UML models uploaded to the repositories can be automatically checked by executing a continuous integration process with Jenkins (Smart, 2011).

Following this approach, we developed a Domain-Specific Language, or DSL (Gronback, 2009), with two objectives: (1) describe the infrastructure of GitHub repositories and Jenkins jobs for each student, and (2) describe the internal structure of the tests applied to the diagrams. Using this language, we can automatically generate the required infrastructure for the assessment of UML models (Criado et al., 2020).

The proposed workflow of the teaching-learning processes is described as follows. Starting from a problem specification, teachers will generate a model with the tests (considering the optimal solution for each required UML diagram) that have to be executed. This model is used for generating the JUnit tests (Ammann and Offutt, 2016) that will be executed when the models constructed by the students are pushed to their repositories. The results of test executions provided by Jenkins will be analyzed by the students for evaluating the correctness of their solutions. These steps will be repeated until the students considers that the solution is correct, i.e., the tests have been passed, and the specification has been fulfilled.

As a summary, GitHub has been used with the following perspectives. In the first place, teachers use repositories to share and manage the course material. Secondly, the students repositories are used for the collaborative development of UML diagrams and models. And third, the feedback obtained for the tests executions should be used as the guidelines to understand which modeling directives and design patterns have to be applied from the specifications.

In the next Software Engineering courses, we are planning to use the GitHub project view to (a) assign the exercises to the accomplished by students, (b) track the progress of these exercises, and (c) automatically score the finished exercises by updating the description for each student's repository. Furthermore, GitHub actions features are being evaluated to run the automatic process of checking UML models stored in students' repositories. This would remove the necessity for using an external tool like Jenkins for such tasks, simplifying the supporting infrastructure in the course.

The application of this approach in the Software Engineering course is providing positive results. During the current academic year (2021-2022), a high percentage of students have used this proposal to improve their modeling skills. Of 165 students, 107 passed all the tests (65%). Of those who passed all the tests, 63 passed the subject's final exam, for which they had to apply the acquired modeling knowledge. This is a 58% from the set of the students passing the tests, and a 38% from the total of students.

3.2 Software Eng. Tools and Methods

Version control systems are one of the main topics of the Software Engineering Tools and Methods course, together with software testing, build automation and continuous integration and deployment. Teamwork is an important soft skill that students should reach along the course. With that aim, several software development activities and a final project are proposed to students' teams to learn and put into practice the main contents of the course.

Students reach the course with some previous knowledge of Git and GitHub basics, so at the beginning, we focus on conflict resolution when merging changes from different users, presenting different

[3]ITSI-GITSI - https://github.com/ualits/itsi-gitsi

Table 1: Involved courses.

Course	Degree	Semester	Used approach(es)
Software Engineering	Computer Science	3rd	Project management, Autograding, Teamwork collab, CI/CD
Software Engineering Tools and Methods	Computer Science	6th	Project management, Teamwork collab, CI/CD
Rapid Application Development	Computer Science	8th	Project management, Autograding, Teamwork collab, CI/CD, Course material
Software Engineering Processes II	Computer Science	8th	Project management, Teamwork collab, Course material
Programming	Mechanical, Electric, Industrial Electronics and Industrial Chemical Engineering	2nd	Project management, Teamwork collab, Course material

situations when conflicts can appear and how to resolve them. Different branches strategies that allow developers to isolate their work and to minimize conflicts when working in teams are reviewed and applied. GitHub is used for enabling team collaboration, presenting two different approaches: 1) The central repository model, when all teammates have 'write' permissions on the same remote GitHub repository, and 2) The "fork and pull request" model, when only one member has 'write' access to the main repository, called upstream, and the rest of team members have only 'read' access in the upstream repository. We focus our interest on the last one since it is widely used in the industry.

To promote commitment and individual effort in collaborative activities, we review the GitHub feature called Insights that provides a report of contributions by each author along the time, showing summarized information of what contributions were made by each team member and when. Moreover, to ensure that the author of a change is whom GitHub says, students learn how to sign their commits.

Some alternatives to GitHub, such as GitLab and Bitbucket, were evaluated several years ago. For some years, GitLab was used in the course, which provides a standalone version that can be downloaded and installed into a machine managed by the students' team. Operating their own service supporting Git repositories was quite interesting for students, as far as the feedback we received, even from local companies that did not want to upload their proprietary code into an online service such as GitHub, which was historically known for hosting open source software projects. However, this activity negatively affected the course schedule since students used to spend much more time on managing the server for installing GitLab, setting up GitLab itself, creating users and managing permissions, and they were not able to start working on Git topics until everything was properly set up. As an alternative of a GitLab service operated by each student team was using Gitlab.com, the online GitLab service; however, the widespread use of GitHub and the additional educational resources that GitHub provides to higher institution students made us choose GitHub against others.

3.3 Rapid Application Development

In the course of Rapid Application Development, we use the tools provided by GitHub to monitor and evaluate students. The fundamentals of this course, as the name suggests, lies in the students being able to develop applications in an efficient and fast way through the use of a series of frameworks that in turn implement development patterns widely used in the world of programming.

To do this, we divide the course into three different milestones:

- Backend. It includes all artifacts necessary for the deployment of a backend, e.g., establishing databases, message queues, APIs or whatever middleware required.
- Frontend. It includes the software artifacts related to a front web application that allows a user to access and interact directly with the data provided by the backend.
- Deployment. Artifacts developed in the two previous points, generating an infrastructure based on containers for the different applications.

In order for students to meet the established objectives in the three milestones, apart from teaching the theoretical concepts related to them, we propose to the student the creation of two applications that, in turn, will be used to evaluate them. The first one is developed individually by the student. The student establishes the theme and technologies, this way, the student becomes much more involved in the learning process. We only demand from the student that the application meets the three milestones. On the other hand, the second application is a project that will be made collaboratively. The teacher defines the theme and technologies and coincides with the ones learned in the course.

In the case of the individual application, the student creates a private repository required to comply with a mainline branching strategy (Fowler, 2020). In this strategy, the student have to generate a new branch for each feature to implement and possible bugs detected in their application. The teacher, throughout the course, is carrying out a detailed follow-up of each private repository, in which he/she controls if the student is fulfilling the milestones. In turn, if any error is detected in the development of the individual project, the teacher can create issues that the student must resolve for the positive evaluation.

The modus operandi of the development of the collective application is a little different. As we will see below, we introduce the student in a more real situation about the development of a project in a company. The teacher generates a team in the course organization. Next, a public repository is generated that has an associated project to which the previously established team is assigned. The milestones are established in that repository, backend, frontend, and deployment. In the associated project, we generate a Kanban board to track students' tasks. In this board, the teacher generates five columns in which we establish a Scrum (Subra and Vannieuwenhuyse, 2018) Board, agile development methodology that the subject follows for the project.

We have a series of fully automated columns in which both students and teachers can follow the tasks that are being developed at any given time. These columns are as follows:

- Backlog: This column establishes all the issues that both students and teachers generate in the project development.
- Sprint Backlog: This column has all the issues belonging to the milestone that we are developing. In the development of the collective project, we established three different sprints of one month each that coincides with the three milestones.
- In Progress: The moment an issue is assigned to a student, it goes to this column.
- Under Review: When the student makes a Pull Request indicating the issue number that he or she has solved, the task goes to this column, being the moment in which the teacher decides if this task has a positive evaluation (accepts Pull Request) or negative (rejects the Pull Request).
- Done: Once the Pull Request is accepted, the issue is considered completed.

For the student evaluation, the teacher assigns a score (in the range 0 to 5) to each issue generated in the collective project. that indicates the issue difficulty. Besides, each project issue is associated with one of the course milestones. Furthermore, the student has the power to assign him/herself an issue whenever he/she wants, but only one at a time. In turn, it is required that the student meets a minimum score in each milestone so that part of the subject is considered passed. For the resolution of each issue, students must first fork of the course public repository, in which they must work resolving the relevant issue. Then, once they consider that they have solved the particular issue, the student makes a Pull Request, with the issue number they have resolved must appear, that the teacher will validate. Throughout the course, the teacher launches the evaluation program presented in Section 2 to check the student' score in each milestone. Score that will be taken into account when evaluating them, and that we can check manually in the project Kanban in the tasks that are arranged in the *Done* column. As you can see, the objective we have when studying this course is not only that the student learns the concepts that are seen in it, but that they acquire the necessary competence to be able to work on a collaborative project.

3.4 Software Engineering Processes 2

The topics covered by this course are vast and diverse, but they are arranged in two well differentiated aspects. First, fundamental issues for the exercise of software engineers' profession that include international and national standards or the profession's ethics. Besides, it covers the measurements and metrics techniques that allow quality control, and the study of how the software product development processes have been applied, including all the analytic processes of those metrics and the refactoring actions selection.

Because this course is planned in the last semester and coincidental with Rapid Application Development, students are used to managing and developing software supported by the GitHub platform. We outline the use of GitHub with a multi-project point of view, which means that a student has to develop projects in different domains (not always software related) belonging to various aleatory built teams. On the one hand, students are forced to self-organize the teams they belong to, and each team has the power to decide how they will work. On the other hand, each student individually suffers the pressure of the deadlines of the issues assigned to them by their classmates.

Project boards are the most used GitHub feature in this course because the only rule we set is that all the tasks performed in the project have to be recorded, assigned, monitored and commented using notes in a Kanban board. This rule allows teachers to control and evaluate students' work. Furthermore, teachers can sometimes define ad-hoc tasks assigned to the team or specific students given feedback or improvements/warnings to the built solutions.

In addition to multi projects coordination, another point that students have to deal with is the connection and management of additional applications such as Overleaf[4], Codacy[5], or SonarCloud[6]. The first one is used to develop a report about how to instantiate the set of standard terminology and guidelines for project management, Project Management Body of Knowledge, to a project for elaborating their final undergraduate dissertation. In this assignment, the use of latex as an external tool supported by a GitHub repository and the project management for a not software project are combined.

The Codacy application is used to automate code reviews (bugs and vulnerabilities) on the commits and pull requests of an easy software project. Each team (two students) decides the programming language and who is responsible for code and for review. The student that reviews for a team is the coder in another team. The followed rule is using GitHub issues and tasks to communicate each other. Finally, SonarCloud, which has similar features to Codacy, is used in a given software application, and a student has to apply refactor actions (Fowler, 2018) that have to be justified using issues and commit comments.

In all the assignments/projects, the GitHub repositories are the submissions content to be evaluated by the teachers, while commits and issued tracks are also evaluation instruments.

3.5 Programming

Programming is an introductory course of the first year in the Mechanical Engineering, Electric Engineering, Industrial Electronics Engineering and Industrial Chemical Engineering degrees at the UAL. Because current teaching methodologies focus on the student's work, the course is based on problem-based learning and collaborative team-working (García-Lázaro et al., 2015; Sol. et al., 2021).

This course is included in the competencies "Software Development Fundamentals" area of the Computing Curricula, which recommends the fluency in at least a programming language, accomplished with essential programming skills and programming concepts, such as basic computation, simple and file I/O, standard conditional and iterative structures, the definition of functions, and parameter passing.

[4]Overleaf – www.overleaf.com
[5]Codacy – www.codacy.com/
[6]SonarCloud – sonarcloud.io/

The lectures and laboratory lessons are based on problems that require programming skills. The problems are scheduled in 13 cooperative working tasks (CWT) and 15 autonomous tasks (AT). Each CWT is described in a written instruction document that includes 3 or 6 problems that have to be collectively solved by a team of 3 or 4 aleatory selected students. Because the plan is common to all the students, it is not pure problem-based learning. Nevertheless, team members may decide the particular techniques or C coding strategy. The AT assignments are no-classroom activities. Instead, each one collects the exercises or problems to be done/coded autonomously.

Until last year, students used some Integrated Development Environment (IDE) in C/C++ programming language that they can install in their personal computers, and the solutions were reviewed manually by the teachers. However, for the academic course 2022, a new workflow has been defined based on the GitHub services (Águila et al., 2022). Most of the contents are going to be released on a GitHub organization. A repository containing a set of problems in the field of Industrial Engineering has been released on the GitHub page and can be used as learning objects (Águila et al., 2021). The CWT and AT instructions have been included as readable files in two repositories. These contents are written in Markdown, a markup language that facilitates the production of texts for the web. Because it is destination independent, these texts can be reused to create other learning materials, such as teachers' or students' notes.

Another GitHub feature that is going to be applied is the version control to help and promote cooperative team working (Águila et al., 2022). The students are arranged in several teams sharing a common repository that the different teammates will complete. Each student has to manage two repositories, the first one to collect AT solutions and a second one to solve group assignments, CWT. These tasks are structured in smaller problems where positive interdependence is needed. Finally, the exercises are distributed among the members, and the solutions have to be agreed upon and merged. The use of teamwork and commits history will help teachers analyze how effective their learning strategy is.

4 CONCLUSIONS

This article describes the set of solutions and strategies that have been applied to five courses related to the Computer Science domain as part of the research work of two innovation teaching projects of the University of Almería (UAL).

These strategies apply GitHub services to support and improve the teaching-learning processes, due to the provided features and the widespread use of this technology. In this article, we define the different strategies and approaches supported by the use of GitHub, like project management, collaborative working, the use of CI/CD processes for automatic validation of exercises, or simply by sharing material thanks to the capabilities provided by the platform. Furthermore, we provide a tool for creating and managing GitHub repositories, to avoid repetitive tasks such as creating an issue in multiple repositories.

The results obtained by applying the strategies in the different courses are promising. They have made it possible to unify teaching-learning strategies within the Degree of Computer Science of the UAL.

There are still open research lines for building specific protocols and methodologies for each course, and for extrapolate the established strategies to other courses with similar characteristics.

ACKNOWLEDGEMENTS

This work has been funded by 21-22-2-13C and 21-22-1-11C projects, call for the Creation of Teaching Groups for Innovation and Good Practices and for the Creation of Teaching Materials of the UAL. Manel Mena has been funded by a grant (ref. FPU17/02010) from the Spanish Government.

REFERENCES

Águila, I., Cañadas, J., and García, J. R. (2022). Github repository with learning objects for industrial engineering degrees in programming. In *Proc. IEEE EDUCON. Global Engineering Education*.

Águila, I., García, J. R., Guirado, R., and Miranda, C. M. (2021). Github repository with learning objects for industrial engineering degrees in programming. In *Proc. VII CINAIC'2021*, pages 86–90.

Ammann, P. and Offutt, J. (2016). *Introduction to software testing*. Cambridge University Press.

Angulo, M. A. and Aktunc, O. (2019). Using github as a teaching tool for programming courses. In *2018 Gulf Southwest Section Conference*.

CC2020 Task Force (2020). Computing curricula 2020: Paradigms for global computing education.

Criado, J., Cañadas, J., Iribarne, L., and Miranda, C. (2020). Automatic assessment of uml models for improving the learning of software engineering students. In *Proc. ICERI2020*, pages 1855–1858.

Feliciano, J., Storey, M.-A., and Zagalsky, A. (2016). Student experiences using github in software engineering courses: a case study. In *Proc. IEEE/ACM 38th ICSE-C*, pages 422–431. IEEE.

Fiksel, J., Jager, L. R., Hardin, J. S., and Taub, M. A. (2019). Using github classroom to teach statistics. *Journal of Statistics Education*, 27(2):110–119.

Fowler, M. (2018). *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

Fowler, M. (2020). Patterns for managing source code branches. https://cutt.ly/jIO6X55. [Online; accessed 19-January-2022].

García-Lázaro, J., Moreno-Ruiz, J., Aguilla Cano, I., and Guirado Clavijo, R. (2015). Teaching computer programming in the degrees of industrial engineering with collaborative and problem-based learning. In *Proc. EDULEARN15*, pages 7623–7633.

Glazunova, O., Parhomenko, O., Korolchuk, V., and Voloshyna, T. (2021). The effectiveness of github cloud services for implementing a programming training project: students' point of view. *J. Phys.: Conf. Ser.*, 1840:012030.

Gronback, R. C. (2009). *Eclipse modeling project: a domain-specific language (DSL) toolkit*. Pearson Education.

Gutica, M. (2018). Improving students' engagement with large-team software development projects. In *Proc. 23rd ACM ITiCSE*, pages 356–357.

Malakar, S. (2021). *AGILE in Practice: Practical Use-cases on Project Management Methods including Agile, Kanban and Scrum*. BPB Publications.

Malikowski, S. R., Thompson, M. E., and Theis, J. G. (2007). A model for research into course management systems: Bridging technology and learning theory. *J. Educ. Comput. Res.*, 36(2):149–173.

Nelson, M. A. and Ponciano, L. (2021). Experiences and insights from using github classroom to support project-based courses. *arXiv preprint arXiv:2103.07242*.

Rahman, A., Partho, A., Morrison, P., and Williams, L. (2018). What questions do programmers ask about configuration as code? In *Proc. 4th RCoSE*, pages 16–22.

Seidl, M., Scholz, M., Huemer, C., and Kappel, G. (2015). *UML@ classroom: An introduction to object-oriented modeling*. Springer.

Smart, J. F. (2011). *Jenkins: The Definitive Guide*. O'Reilly Media, Inc.

Sol., R., Santos., E., Reis., M., and Pereira., L. (2021). Computer supported collaborative learning for programming: A systematic review. In *Proc. 13th CSEDU*, pages 184–191.

Subra, J.-P. and Vannieuwenhuyse, A. (2018). *Scrum: un método ágil para sus proyectos*. Ediciones ENI.

Tushev, M., Williams, G., and Mahmoud, A. (2020). Using github in large software engineering classes. an exploratory case study. *Computer Science Education*, 30(2):155–186.

Zagalsky, A., Feliciano, J., Storey, M.-A., Zhao, Y., and Wang, W. (2015). The emergence of github as a collaborative platform for education. In *Proc. 18th ACM CSCW*, pages 1906–1917.