GitSEED: A Git-backed Automated Assessment Tool for Software Engineering and Programming Education

Pedro Orvalho
pmorvalho@tecnico.ulisboa.pt
INESC-ID, IST, U. Lisboa
Lisboa, Portugal

Mikoláš Janota
mikolas.janota@cvut.cz
Czech Technical University in Prague
Prague, Czech Republic

Vasco Manquinho
vasco.manquinho@tecnico.ulisboa.pt
INESC-ID, IST, U. Lisboa
Lisboa, Portugal

Abstract

Due to the substantial number of enrollments in programming courses, a key challenge is delivering personalized feedback to students. The nature of this feedback varies significantly, contingent on the subject and the chosen evaluation method. However, tailoring current Automated Assessment Tools (AATs) to integrate other program analysis tools is not straightforward. Moreover, AATs usually support only specific programming languages, providing feedback exclusively through dedicated websites based on test suites.

This paper introduces GitSEED, a language-agnostic automated assessment tool designed for Programming Education and Software Engineering (SE) and backed by GitLab. The students interact with GitSEED through GitLab. Using GitSEED, students in Computer Science (CS) and SE can master the fundamentals of git while receiving personalized feedback on their programming assignments and projects. Furthermore, faculty members can easily tailor GitSEED's pipeline by integrating various code evaluation tools (e.g., memory leak detection, fault localization, program repair, etc.) to offer personalized feedback that aligns with the needs of each CS/SE course. Our experiments assess GitSEED's efficacy via comprehensive user evaluation, examining the impact of feedback mechanisms and features on student learning outcomes. Findings reveal positive correlations between GitSEED usage and student engagement.

CCS Concepts

• **Applied computing → Computer-assisted instruction**; **Interactive learning environments**; **Computer-managed instruction**; *E-learning*; *Collaborative learning*.

Keywords

Automated Assessment Tools, Programming Education, Software Engineering Education, Computer-aided Education, Git

1 Introduction

The increasing need for programming and Software Engineering (SE) education has led to the emergence of various online courses, including Massive Open Online Courses (MOOCs) [11, 18, 19]. Providing feedback to CS students on their programming assignments and projects demands considerable time and effort from the faculty. Thus, there is a rising demand for systems, such as Automated Assessment Tools (AATs), that can deliver automated, comprehensive, and personalized feedback to students. When compared to non-automated evaluators, such as teaching assistants, AATs can evaluate several assignments or code submissions efficiently and quickly. Hence, AATs facilitate the learning process since students get their feedback much faster. Moreover, AATs offer objectivity and consistency, adhering to some evaluation metric (e.g., a test suite).

The interest and development of AATs dates back to the 1960s [12, 27]. Over the past two decades, there has been a surge in the growth and adoption of AATs [1, 4, 8, 14, 26, 28, 32]. However, despite the remarkable growth in the development and usage of AATs, certain drawbacks have become increasingly apparent. Primarily, a majority of AATs [4, 14] merely display the outcomes of a set of input/output tests used for the student's evaluation and lack other kinds of feedback. Secondly, ATTs tend to be specific to one programming language or a limited set of languages. AATs that are language-agnostic are scarce [8, 28]. Thirdly, AATs typically offer feedback solely through dedicated websites, necessitating students to familiarize themselves with new GUI interfaces. Finally, it is either challenging or impractical to adapt most AATs to integrate other program analysis tools, which might be essential to provide more personalized feedback in some CS/SE courses.

This paper introduces GitSEED, a new tool that overcomes the aforementioned limitations of previous AATs. GitSEED is a novel **Git**-backed AAT for **S**oftware **E**ngineering and Programming **Ed**ucation. Figure 1 presents the overview of GitSEED. As Figure 1 shows, the students interact with GitSEED through GitLab. This way, CS/SE students can learn the fundamentals of git while receiving personalized feedback on their programming assignments and projects. Afterwards, using GitLab's runners, GitSEED is notified whenever there is a new submission from group X. GitSEED evaluates this new submission against a test suite and using program analysis tools defined by the faculty. Finally, the resulting evaluation report is pushed into X's git repository (repo) so that the students have access to personalized feedback right away.

Furthermore, GitSEED is language agnostic, i.e., it can be used for any CS/SE course no matter the programming language(s) used. Moreover, most CS/SE students are familiar or will be familiar throughout their courses, with several git web interfaces (e.g., GitLab, GitHub, Gitea). Therefore, GitSEED eliminates the necessity for students to acquaint themselves with an unfamiliar GUI interface, which happens frequently in several universities where different CS/SE courses use different GUI interfaces for automated assessment of programming tasks [1, 4, 8, 14, 27].

GitSEED has two different categories of assessments: labs and projects. Either one is optional, and it is possible to have an unlimited number of projects depending on the chosen configuration. Faculty can choose which assessment model aligns best with their courses. Moreover, faculty members can easily tailor the pipeline of GitSEED, enhancing the quality of feedback provided to the students aligned with the needs of each course. For example, code evaluation tools can be integrated into GitSEED, such as memory leak detection [17], fault localization [23], program repair [11, 24], plagiarism detection [31], code coverage [9], among others [7, 25].

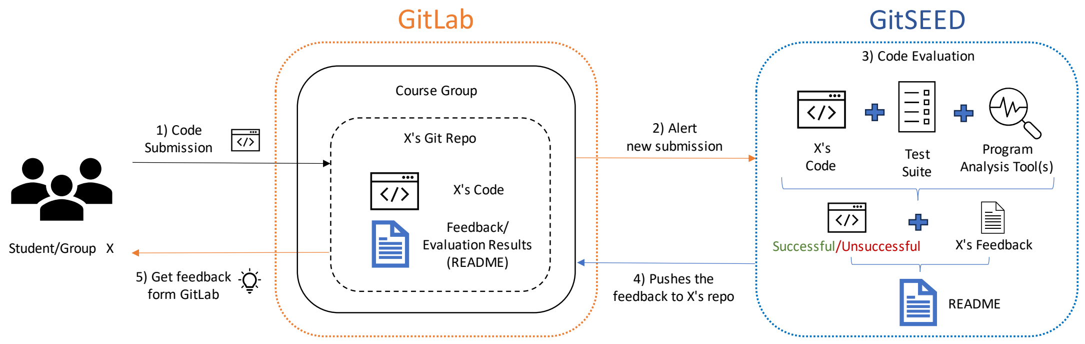

Figure 1: The overview of GitSEED.

The paper is organized as follows. Section 2 presents the implementation and possible configurations of GitSEED in more detail. GitSEED has already proven successful in two separate courses, a first-year programming course and a CS graduate course. Section 3 discusses our experiments and evaluates the effectiveness of GitSEED in enhancing programming education. Through the analysis of feedback from students enrolled in a first-year undergraduate course, we explore the role of GitSEED's features, including dashboards and feedback mechanisms, in facilitating learning and improving student performance. Finally, Section 4 briefly reviews related work, and the paper concludes in Section 5.

To summarize, this paper makes the following contributions:

- We present GitSEED, an open-source language-agnostic automated assessment tool designed for Software Engineering (SE) and Programming Education and backed by GitLab;
- GitSEED is integrated into GitLab's continuous integration (CI) workflow, which adopts educational assessment within a professional version control platform rather than a dedicated website, like so many other AATs;
- Students interact with GitSEED through GitLab, learning this way the fundamentals of git while receiving personalized feedback on their assignments;
- Faculty members can easily adapt GitSEED by integrating other code analysis tools to offer personalized feedback that aligns with the needs of each CS/SE course.
- GitSEED is publicly available on GitLab [20] and on Zenodo [22].

2 GitSEED

This section presents the internals of GitSEED and configuration options. Section 2.1 describes the GitLab features required by GitSEED. Next, Section 2.2 focuses on the back-end of GitSEED and its workflow, and Section 2.3 details the measures taken in order to ensure that all stages of the GitSEED pipeline are safe. Finally, Section 2.4 explains our implementation of GitSEED.

2.1 GitLab

Figure 2 illustrates the complete workflow of GitSEED, where GitLab plays the role of an intermediary between the students and GitSEED. Notice that students only interact with GitLab, and then GitLab triggers the processing of submissions in GitSEED. Furthermore, GitSEED was designed to work with other git web interfaces. Modern, widely-used programming editors/environments (e.g., VS Code) already feature user-friendly interfaces for managing git repos. Hence, the submission process and getting feedback can be easily done within the student's programming environment, eliminating the need to exit their coding workspace. Alternatively, students can also use GitLab's web interface.

2.1.1 GitLab Group, Subgroups and Course Repo. GitSEED expects the following structure of groups in GitLab: a main GitLab group with all the students (e.g., CS101), a subgroup for each distinct evaluation element (e.g., labs, project, etc.), an additional subgroup for feedback, and a git repo for the entire course (CS101, in Figure 2) that contains all the course's information and dashboards for each evaluation element.

GitSEED has different repos for labs and projects. The rationale behind this choice is that while labs remain open throughout the semester, projects have distinct deadlines and may involve different groups of students. Furthermore, the feedback can also be pushed directly to the same git repos used by students for code submissions. However, having two different repos, one for code submissions and one for getting feedback, is the best approach for first-year students. This approach mitigates the chances of merging conflicts or git conflicts between students and GitSEED. Hence, GitSEED uses different repos to simplify the students' repos synchronization. This way, students manage their code development repos, and GitSEED only submits to the feedback repo.

In GitLab, each project member is assigned a role that determines which actions they can take in the git repo [1]. In GitSEED, students assume the role of "Developers" for their own repos (e.g., projects, labs), granting them read and write access. However, students assume the role of "Reporters" for their feedback git repos and in the course global project, granting them viewing but not editing privileges. Note that the group of students can only see their own feedback repo and not those of other groups. Finally, faculty members hold the roles of "Maintainers" or "Owners" for all repos, depending on the chosen configuration.

2.1.2 Continuous Integration (CI). GitSEED takes advantage of the CI pipeline[2] available on GitLab. The CI pipeline, essential

[1]https://docs.gitlab.com/ee/user/permissions.html
[2]https://docs.gitlab.com/ee/ci/

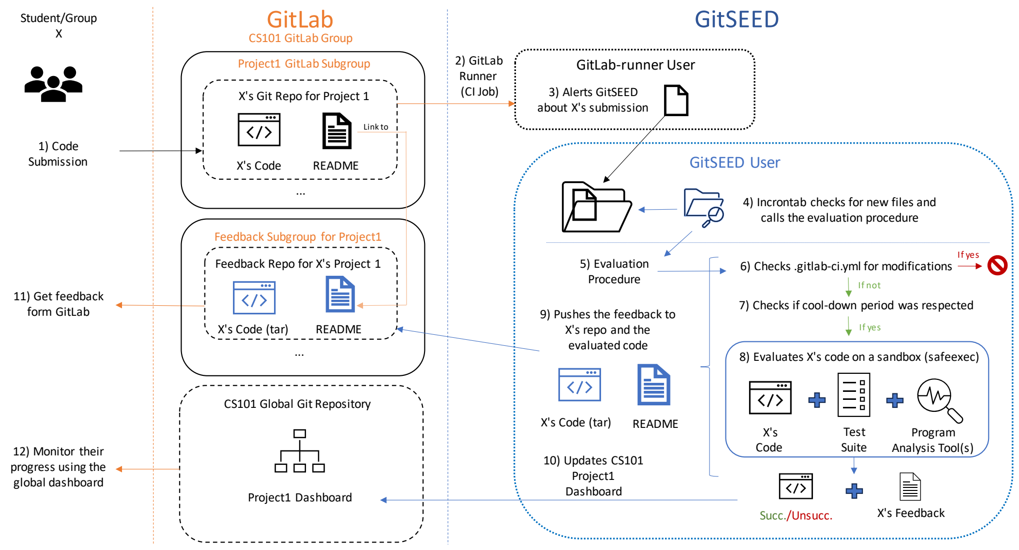
Student/Group X
GitLab
CS101 GitLab Group
Project1 GitLab Subgroup
X's Git Repo for Project 1
X's Code
README
Link to
...
1) Code Submission
Feedback Subgroup for Project1
Feedback Repo for X's Project 1
X's Code (tar)
README
...
11) Get feedback form GitLab
CS101 Global Git Repository
Project1 Dashboard
12) Monitor their progress using the global dashboard
2) GitLab Runner (CI Job)
GitSEED
GitLab-runner User
3) Alerts GitSEED about X's submission
GitSEED User
4) Incrontab checks for new files and calls the evaluation procedure
5) Evaluation Procedure
6) Checks .gitlab-ci.yml for modifications
If yes
If not
7) Checks if cool-down period was respected
If yes
8) Evaluates X's code on a sandbox (safeexec)
X's Code
Test Suite
Program Analysis Tool(s)
Succ./Unsucc.
X's Feedback
9) Pushes the feedback to X's repo and the evaluated code
X's Code (tar)
README
10) Updates CS101 Project1 Dashboard

Figure 2: Workflow of GitSEED for processing a new project submission from group X in CS101.

for testing and deploying software projects, operates through a `.gitlab-ci.yml` script outlining all testing and deployment actions. It utilizes "runners", agents executing these actions, such as tests, as defined in the script.

GitLab Runner. GitSEED requires a GitLab runner to be installed and self-hosted in the machine where GitSEED is running (e.g., a Linux virtual machine (VM)). In this machine, the runner runs the job described in the `.gitlab-ci.yml`. The runner's job is to add information to GitSEED's queue that student X made a new submission to project Y. The faculty needs to adapt the `.gitlab-ci.yml` based on the programming language(s) being evaluated in the course. Distinct students can use different programming languages for the same programming task.

2.2 GitSEED's Back-end

Next, we detail the stages of the GitSEED workflow from Figure 2.

2.2.1 GitLab Manager. The GitLab manager interacts with GitLab and creates/modifies/clones all the necessary subgroups and repos and assigns the students and faculty to their own repos. GitSEED works for single-student groups and for groups with several students. Using the GitLab manager, the faculty can easily manage students' access to their git repos. The students' accesses can be removed, for example, after a project deadline or if the students modified something in the git repo that were not supposed to (e.g., `.gitlab-ci.yml`).

2.2.2 Assessments. GitSEED has two categories of assessments: labs and projects. Either one is optional, and it is possible to have an unlimited number of projects depending on the chosen configuration. Each type of assessment has its own evaluation script.

Labs. GitSEED allows faculty members to publish each different lab's exercises in the student's repos. Each lab corresponds to a practical class. Additionally, with GitSEED, students who do not finish all the lab exercises during the class can still conclude and automatically check their implementations afterwards.

Projects. GitSEED allows the publication of the projects' descriptions and related data in the students' repos. Moreover, after a project's deadline, faculty members can remove the students' access to write into their repos and then reevaluate all the projects one last time. This last reevaluation might be necessary in case there was any submission that was not assessed due to the *cool-down period* (see description of cool-down periods next).

2.2.3 Commits Database. GitSEED maintains a database containing all students' commits timestamps for every evaluation element. The goal is to have a cool-down period for each different evaluation element, ensuring that only the submissions respecting their previous cool-down period are evaluated. This measure is implemented to prevent overloading GitSEED's machine. Note that some CS/SE courses have thousands of students, who tend to submit multiple times, especially near project deadlines. Moreover, a cool-down period forces students to think more thoroughly about their program before making new submissions, as no new feedback will be provided for commits made during this time. The default cool-down period is set at 1 minute for lab exercises and 10 minutes for project assignments. Nevertheless, these periods can be easily modified (see Section 2.2.8). Furthermore, the feedback report (README) lets the students know when their current cool-down period is over.

2.2.4 Dashboard. GitSEED keeps a dashboard/leaderboard for each distinct evaluation element. These dashboards are automatically posted by GitSEED in the course's central git repo so all students can monitor their progress with regard to their colleagues. The dashboards keep track of each student/group's number of successful/unsuccessful tests, their number of submissions, and the number of days since the beginning of the project/lab assignment.

2.2.5 GitLab-Runner. The runner (see Section 2.1.2) needs to be installed in the same machine where GitSEED is running and to have write access to the folder that keeps track of new submissions.

2.2.6 Incrontab. The machine where GitSEED is installed has an incrontab daemon that is triggered by the GitLab runner. The GitLab runner adds the information that a new commit was performed on a given repo, and that triggers the evaluation procedure.

2.2.7 Evaluation. During the evaluation process, GitSEED first checks if the students modified the `.gitlab-ci.yml` script. If this is the case, students lose their rights to push/modify the git repo and are asked to reach out to faculty members. Otherwise, GitSEED checks if the cool-down period was respected. If not, the student's new submission is not evaluated. Otherwise, if the cool-down period was respected, then GitSEED proceeds to the next evaluation step.

`safeexec`. To run the students' code safely, GitSEED uses `safeexec` [30] which is a lightweight sandbox for executing user programs. Alongside `safeexec`, other program analysis tools can be run on the students' code. After completing the evaluation, GitSEED submits the evaluation report (README) and a tar file containing the evaluated code (e.g., a programming assignment's implementation) to the respective feedback repo. Finally, GitSEED updates the respective dashboard in the course's central git repo with the student's performance.

2.2.8 Configurations. GitSEED has several predefined configurations that can be easily modified in the configuration file:

- Cool-down Period (default: 1 min): Amount of time students need to wait between their own submissions;
- Output Visible (default: false): GitSEED shows (or does not) the output of all tests to the students;
- Only First Wrong Output Visible (default: true): GitSEED only shows students their first incorrect output. This option is only used if the previous option is set to true;
- CPU Time Limit (default: 5 sec): GitSEED runs the students' code with this CPU time limit for each test case;
- Memory Limit (default: 8 GB): GitSEED runs the students' code with this memory limit for each test case.

Easily Tailored. GitSEED's current methods for evaluation are fully language agnostic. The evaluation scripts can be easily tailored to evaluate different programming languages. Furthermore, faculty members can quickly adapt the GitSEED pipeline by integrating or replacing various code evaluation tools (e.g., memory leak detection, fault localization, program repair, plagiarism checks, solution checkers) to offer personalized feedback that aligns with the needs of each CS/SE course. Lastly, GitSEED was designed with modularity in mind. On that account, one can easily remove, add, or modify any component of GitSEED without compromising it.

2.3 Safety Measures

Several measures must be ensured for GitSEED to operate safely. Firstly, the GitLab runner user needs to be granted write access to GitSEED's folder for new submissions. However, this user should not have access to any other folders, as the GitLab runner executes code from the `.gitlab-ci.yml` script, which may be tampered with by students. By limiting access to only that folder, the GitLab runner cannot alter or read anything else from GitSEED. Furthermore, GitSEED runs the students' code using `safeexec`, which simulates a sandbox controlling read/write accesses. Note that GitSEED is not dependent on `safeexec`. Due to GitSEED modularity, `safeexec` can be quickly replaced with some other sandbox application. Lastly, given the crucial role of `.gitlab-ci.yml` in GitSEED's functionality, this script is added to the `gitignore` file and students' READMEs explicitly instruct them not to edit this `yml` script. Nevertheless, GitSEED checks for any tampering with this script before evaluating the student's code. If detected, GitSEED restricts the student's access to the repo until the faculty checks the situation.

2.4 Implementation

For this paper, we used a GitLab instance self-hosted at Instituto Superior Técnico. The GitSEED system was deployed on a dedicated virtual machine running Linux (Debian 4.19) on a AMD Opteron(TM) Processor 6276 with 16GB of RAM. Additionally, the virtual machine hosted the gitlab-runner package, version 15.9.1. GitSEED is implemented using `bash` and python3 (version 3.9.16). GitSEED uses `bash` to execute and evaluate the students' code. On the other hand, GitSEED relies on python3 and `curl` to communicate with GitLab, through its API (v3.15.0). Furthermore, GitSEED utilizes python3 and `sqlite3` for the maintenance of the course's dashboards and the database containing the commit history.

3 Impact Discussion

This section discusses our experiments with GitSEED, between Spring 2023 and Spring 2024, in two distinct academic courses at Instituto Superior Técnico, a first-year undergraduate and a graduate course. GitSEED offers both formative and summative assessments. Formative assignments, such as lab classes, remain accessible throughout the semester, allowing students to revise until correct. Summative assignments, such as projects, also permit unlimited attempts but come with strict deadlines. Students were briefed that formative assignments serve as learning aids, encouraging exploration without fear of repercussions for errors. The aim is for students to utilize GitSEED until mastery is achieved. Conversely, summative assignments serve as assessments of acquired knowledge and skill, showcasing proficiency in the subject. Since these summative assignments require more computation time and memory, higher cool-down periods were established between each group's submissions, to prevent overloading GitSEED's machine.

3.1 Courses Setup

*3.1.1 **Undergraduate Course (Spring 2023)**.* GitSEED was initially used in Spring 2023 in a first-year undergraduate course, Introduction to programming in C, with a total of 528 enrolled students. GitSEED was used to assess this course's labs (formative assignments) and projects (summative assignments).

Assignments. For formative assignments, GitSEED created individual git repos for the lab classes. Configuration settings included a one-minute cool-down period, a five-second CPU time limit, and an 8GB memory limit for each programming assignment across eight labs. Throughout these eight labs, students made a total of 10338 code contributions to their repos. Regarding the summative assignments, this course had two different programming projects, each configured for single-student groups. Configuration settings included a five-second CPU time limit, a 16GB memory limit, and a 10-minute cool-down period for project evaluations. Students made a total of 15061 code contributions to their repos, 7916 to the first project and 7145 to the second one.

Program Analysis Tools. For projects evaluation, GitSEED was tailored to: (1) identify forbidden libraries in student projects, and (2) run valgrind [17] to detect memory leaks in their code. Providing feedback on memory leaks proved beneficial, particularly for first-year students unfamiliar with these tools.

Opportunities for improvement. Throughout the course, we noticed that some students forgot to pull the feedback from GitLab to their local repos before pushing new modifications, resulting in git-merge issues. Consequently, GitLab's CI would ignore these commits. This happened because students use GitLab web interface to get their feedback while coding in their local git repos. To address this issue, GitSEED now publishes feedback in separate repositories, with students' READMEs containing links for easy synchronization, especially for first-year students.

3.1.2 ***Graduate Course (Fall 2023)***.

In Fall 2023, GitSEED was also used in a graduate course on Automated Reasoning with 38 students. The project consisted of solving an NP-Hard optimization problem through a Boolean logic solver using Python. There were 21 groups, each consisting of a maximum of two students. Configuration settings included a one-minute CPU time limit, a 16GB memory limit, and a 20-minute cool-down period for project evaluations. Throughout the project, students made a total of 269 contributions to their repos. Once again, we customized GitSEED, in this case, to incorporate both private and public test cases for evaluating students' code. Moreover, we also inserted additional software into the GitSEED pipeline that gave students feedback about the satisfiability and optimality of their projects' solutions.

3.1.3 ***Undergraduate Course (Spring 2024)***.

In Spring 2024, GitSEED once again played a pivotal role by supporting the first-year undergraduate course, Introduction to Programming in C, which had a total enrollment of 509 students. GitSEED served as the platform for assessing labs and projects in the course, employing configurations similar to those outlined in Section 3.1.1. However, notable adjustments were made for this course iteration.

Feedback. Feedback was provided in separate repositories based on the insights gained from the previous year.

Program Analysis Tools. A significant enhancement was the integration of four additional program analysis tools into GitSEED: CFaults, cppcheck, clang-tidy, and `Lizard`. `Lizard` [16] is a cyclomatic complexity analyzer for various programming languages, aiding in evaluating code length and complexity. The fault localization tool pinpointed faulty statements within the programs based on a test suite. Additionally, cppcheck [6] and clang-tidy [2] are static analyzers used to detect uninitialized variables and various errors, such as division by zero. Finally, CFaults [21] is a formula-based fault localization tool that pinpoints bug locations within the programs. The insights generated by these tools were compiled into feedback reports and appended alongside test-suite evaluation outcomes in the students' feedback repositories. The results from both the fault localization tool and the static analyzers were presented to students as "Hints", strategically guiding them towards potential problematic statements within their programs. This approach aimed to provide students with targeted assistance in identifying and rectifying programming errors. Moreover, GitSEED was configured to display only the first incorrect output to students, fostering a focused learning environment.

3.2 User Study

In Spring 2024, we conducted a comprehensive user study with students to gather valuable feedback on their experience with GitSEED, particularly focusing on its dashboards and the various types of feedback provided by analysis tools, namely `valgrind`, `lizard`, and "hints" (generated by fault localization and static analyzers). Throughout the course, we noticed that incorporating motivational elements, such as the dashboards available within GitSEED, effectively encouraged student engagement and facilitated their progress. Approximately 20% of the students who were enrolled in the course for the entire semester took part in the questionnaire. They were asked anonymously to evaluate the usefulness of the different feedback mechanisms and features of GitSEED they encountered during the semester (see Appendix A). The findings revealed that students perceived the following aspects as particularly beneficial:

GitSEED: 91.8% of students found GitSEED to be a valuable resource. Its role in providing a centralized platform for assignment submission, feedback reception, and revision evidently streamlined the learning process and enhanced overall comprehension. **Dashboards**: 82.2% of students acknowledged the significance of dashboards in tracking their progress and monitoring their performance relative to course objectives. **Hints**: Despite being less prevalent than other feedback mechanisms, 68.5% of students recognized the utility of hints generated by fault localization and static analyzers. These hints acted as invaluable pointers, directing students towards potential errors in their code and fostering a deeper understanding of programming concepts through self-correction. **`Valgrind`**: 90.4% of students found the feedback from `valgrind` to be beneficial. This tool's ability to detect memory management issues and provide detailed diagnostics undoubtedly aided students in debugging their programs and writing more robust code. **`Lizard`**: 75.3% of students appreciated the insights offered by `lizard`, particularly its analysis of code complexity and length. By highlighting areas of code that might require simplification or restructuring, `lizard` contributed to the optimization of students' coding practices and the cultivation of clearer, more efficient programming habits. In addition to evaluating the specific components of GitSEED, students were given the opportunity to provide general feedback through short-answer responses. These open-ended questions allowed students to express their thoughts, suggestions, and concerns regarding their overall experience with GitSEED. Overall, the user study underscored the positive impact of GitSEED's features and feedback mechanisms on students' learning experiences, reaffirming its value as a comprehensive educational tool for programming courses.

4 Related Work

Automated Assessment Tools (AATs). Over the past decades, there has been a growing interest in the automated evaluation of Software Engineering (SE) and Computer Science (CS) students [27]. Typically, AATs assess programming tasks using input/output (IO) tests predefined by the course's faculty. There exists a substantial number of AATs that function as web-based Integrated Development Environments (IDEs) for evaluating students' code using IO tests. Examples include CodeOcean [33], Mooshak [14], and Web-Cat [8]. Enki [26] is also a web-based IDE although it offers other kinds of pedagogical skills to the students, e.g., gamification features. Furthermore, Autolab [1] and Submitty [28] are open-source web-based course management platforms that automatically grades students' code. Autolab maintains scoreboards for each evaluation element in order to motivate the students, while Submitty provides an interface for Teaching Assistants (TAs) to manually grade assignments. Additionally, Codeboard.io [4] is a web-based IDE to teach programming tasks. Faculty members can share programming exercises with the students. These exercises are assessed using a result string or a set of predefined unit tests. The set of programming languages available is limited and Codeboard.io is difficult to tailor in order to get more personalized feedback for the students. GradeStyle [13] serves as a code style marker tool that provides feedback for assignments in Java by opening a GitHub issue in each student's repository. Moreover, gradescope [10] is another online tool to administer and grade programming assignments as well as other kinds of assessments (e.g., exams). However, gradescope only has paid licenses for education.

GitHub Classroom [3] is an AAT tool available on GitHub that allows faculty to create and manage digital classrooms and assignments. GitHub Classroom uses the same mechanism of a CI runner (GitHub Actions) to process student code and report on quality aspects. GitHub Classroom shares several benefits with GitSEED, is language agnostic, and enables students to learn the fundamentals of git while receiving feedback on their assignments. However, GitHub Classroom operates on a third-party platform. There may be regulatory or institutional policies that restrict the use of cloud-based services for certain types of data. On the other hand, GitLab is open-source and can be self-hosted by educational institutes. Additionally, GitHub Classroom may not seamlessly integrate with existing learning management systems (LMS) used by educational institutions. This lack of integration can result in administrative challenges, such as maintaining separate platforms for course materials, grades, and communication. While GitSEED can be easily integrated with this kind of systems. Finally, while GitHub Classroom is free to use, some advanced features or integrations may require paid GitHub plans. For example, the number of minutes available for GitHub Actions (CI) is limited per month. Instructors may need to consider the cost of providing GitHub accounts or repositories for students, especially in cases where institutional resources are limited. In contrast, educational institutions can use the premium version of GitLab for free, and both GitLab and GitSEED are open-source projects. Finally, it is worth noting that there is no equivalent to GitHub Classroom on GitLab, highlighting an opportunity for GitSEED to fill this gap.

Competitive Programming Contests (CPCs). CPCs are online platforms that host programming contests. In these websites, students and CS/SE professionals, engage in solving computational problems under time/memory constraints. These contests serve as platforms to assess and enhance problem-solving skills, algorithmic efficiency, and programming proficiency. Typically, CPCs assess contestants' code using an IO test suite. LeetCode [15], topcoder [34], Codeforces [5], and replit [29] are among the most famous CPCs. Both Codeforces [5] and replit [29] offer features for programming education. However, the evaluation process relies solely on IO tests.

5 Conclusion

This paper presents GitSEED, an open-source, language-agnostic automated assessment tool (AAT) seamlessly integrated with GitLab. Students benefit from personalized feedback on programming assignments and projects, mastering Git fundamentals simultaneously. Notably, GitSEED eliminates the need for students to navigate new GUI interfaces. Integrated into GitLab's continuous integration (CI) workflow, GitSEED brings educational assessment into a professional version control platform rather than a dedicated web-based platform. Furthermore, faculty can easily customize GitSEED's pipeline with various code evaluation tools. Our experiments showcased GitSEED's success in both undergraduate and graduate courses, affirming its efficacy in programming education. It enhances student engagement and learning outcomes. Positive student feedback highlights GitSEED contribution to active learning and a supportive educational environment.

Acknowledgments

This work was partially supported by Portuguese national funds through FCT, under projects UIDB/50021/2020 (DOI: 10.54499/-UIDB/50021/2020), PTDC/CCI-COM/2156/2021 (DOI: 10.54499/-PTDC/CCI-COM/2156/2021) and 2022.03537.PTDC (DOI: 10.54499/-2022.03537.PTDC) and grant SFRH/BD/07724/2020 (DOI: 10.54499/-2020.07724.BD). PO acknowledges travel support from the EU's Horizon 2020 research and innovation programme under ELISE Grant Agreement No 951847. This work was also supported by the MEYS within the program ERC CZ under the project POSTMAN no. LL1902 and co-funded by the EU under the project *ROBOPROX* (reg. no. CZ.02.01.01/00/22_008/0004590). This article is part of the RICAIP project funded by the EU's Horizon 2020 research and innovation program under grant agreement No 857306.

References

[1] autolab. 2023. https://autolabproject.com. Accessed: 2024-05-01.
[2] clang tidy. 2024. . https://clang.llvm.org/extra/clang-tidy/. Accessed: 2024-05-01.
[3] GitHub Classroom. 2024. . https://classroom.github.com. Accessed: 2024-05-01.
[4] Codeboard.io. 2023. https://codeboard.io. Accessed: 2024-05-01.
[5] Codeforces. 2023. https://codeforces.com. Accessed: 2024-05-01.
[6] cppcheck. 2024. . https://cppcheck.sourceforge.io. Accessed: 2024-05-01.
[7] Pedro da Silva. 2019. *SQUARES : A SQL Synthesizer Using Query Reverse Engineering*. Master's thesis. Instituto Superior Técnico, Lisboa, Portugal.
[8] Stephen H. Edwards and Manuel A. Pérez-Quiñones. 2008. Web-CAT: automatically grading programming assignments. In *Proceedings of the 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2008, Madrid, Spain, June 30 - July 2, 2008*, June Amillo, Cary Laxer, Ernestina Menasalvas Ruiz, and Alison Young (Eds.). ACM, "", 328. https://doi.org/10.1145/1384271.1384371
[9] gcov. 2024. . https://gcc.gnu.org/onlinedocs/gcc/Gcov.html. Accessed: 2024-05-01.
[10] GradeScope. 2024. . https://www.gradescope.com. Accessed: 2024-05-01.

[11] Sumit Gulwani, Ivan Radicek, and Florian Zuleger. 2018. Automated clustering and program repair for introductory programming assignments. In *PLDI 2018*. ACM, "", 465–480.

[12] Jack Hollingsworth. 1960. Automatic graders for programming classes. *Commun. ACM* 3, 10 (1960), 528–529. https://doi.org/10.1145/367415.367422

[13] Callum Iddon, Nasser Giacaman, and Valerio Terragni. 2023. GRADESTYLE: GitHub-Integrated and Automated Assessment of Java Code Style. In *45th IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training, SEET@ICSE 2023, Melbourne, Australia, May 14-20, 2023.* IEEE, "", 192–197. https://doi.org/10.1109/ICSE-SEET58685.2023.00024

[14] José Paulo Leal and Fernando M. A. Silva. 2003. Mooshak: a Web-based multi-site programming contest system. *Softw. Pract. Exp.* 33, 6 (2003), 567–581. https://doi.org/10.1002/spe.522

[15] Leetcode. 2023. https://leetcode.com. Accessed: 2024-05-01.

[16] Lizard. 2024. Lizard: A simple code complexity analyser. https://github.com/terryyin/lizard. Accessed: 2024-05-01.

[17] Nicholas Nethercote and Julian Seward. 2003. Valgrind: A program supervision framework. *Electronic notes in theoretical computer science* 89, 2 (2003), 44–66.

[18] Pedro Orvalho, Mikoláš Janota, and Vasco Manquinho. 2022. C-Pack of IPAs: A C90 Program Benchmark of Introductory Programming Assignments. https://doi.org/10.48550/arXiv.2206.08768

[19] Pedro Orvalho, Mikolás Janota, and Vasco Manquinho. 2022. InvAASTCluster: On Applying Invariant-Based Program Clustering to Introductory Programming Assignments. *CoRR* abs/2206.14175 (2022). https://doi.org/10.48550/ARXIV.2206.14175 arXiv:2206.14175

[20] Pedro Orvalho, Mikoláš Janota, and Vasco Manquinho. 2024. . https://gitlab.inesc-id.pt/u020557/GitSEED. Accessed: 2024-09-10.

[21] Pedro Orvalho, Mikolás Janota, and Vasco Manquinho. 2024. CFaults: Model-Based Diagnosis for Fault Localization in C Programs with Multiple Test Cases. In *Formal Methods - 26th International Symposium, FM 2024, Milan, Italy, 2024, Proceedings (Lecture Notes in Computer Science, Vol. 14933)*. Springer, Cham, 463–481. https://doi.org/10.1007/978-3-031-71162-6_24

[22] Pedro Orvalho, Mikoláš Janota, and Vasco Manquinho. 2024. *GitSEED: A Git-backed Automated Assessment Tool for Software Engineering Education.* https://doi.org/10.5281/zenodo.13741158

[23] Pedro Orvalho, Mikolás Janota, and Vasco M. Manquinho. 2024. CFaults: Model-Based Diagnosis for Fault Localization in C Programs with Multiple Test Cases. *CoRR* abs/2407.09337 (2024). https://doi.org/10.48550/ARXIV.2407.09337 arXiv:2407.09337

[24] Pedro Orvalho, Jelle Piepenbrock, Mikolás Janota, and Vasco M. Manquinho. 2023. Graph Neural Networks for Mapping Variables Between Programs. In *ECAI 2023 - 26th European Conference on Artificial Intelligence (Frontiers in Artificial Intelligence and Applications, Vol. 372)*. IOS Press, Poland, 1811–1818. https://doi.org/10.3233/FAIA230468

[25] Pedro Orvalho, Miguel Terra-Neves, Miguel Ventura, Ruben Martins, and Vasco M. Manquinho. 2019. Encodings for Enumeration-Based Program Synthesis. In *Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11802)*, Thomas Schiex and Simon de Givry (Eds.). Springer, ., 583–599. https://doi.org/10.1007/978-3-030-30048-7_34

[26] José Carlos Paiva, José Paulo Leal, and Ricardo Alexandre Peixoto de Queirós. 2016. Enki: A Pedagogical Services Aggregator for Learning Programming Languages. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2016, Arequipa, Peru, July 9-13, 2016*, Alison Clear, Ernesto Cuadros-Vargas, Janet Carter, and Yván Túpac (Eds.). ACM, "", 332–337. https://doi.org/10.1145/2899415.2899441

[27] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. 2022. Automated Assessment in Computer Science Education: A State-of-the-Art Review. *ACM Trans. Comput. Educ.* 22, 3 (2022), 34:1–34:40. https://doi.org/10.1145/3513140

[28] Matthew Peveler, Jeramey Tyler, Samuel Breese, Barbara Cutler, and Ana L. Milanova. 2017. Submitty: An Open Source, Highly-Configurable Platform for Grading of Programming Assignments (Abstract Only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2017, Seattle, WA, USA, March 8-11, 2017*, Michael E. Caspersen, Stephen H. Edwards, Tiffany Barnes, and Daniel D. Garcia (Eds.). ACM, "", 641. https://doi.org/10.1145/3017680.3022384

[29] replit. 2023. https://replit.com. Accessed: 2024-05-01.

[30] safeexec. 2024. . https://github.com/ochko/safeexec. Accessed: 2024-05-01.

[31] Saul Schleimer, Daniel Shawcross Wilkerson, and Alexander Aiken. 2003. Winnowing: Local Algorithms for Document Fingerprinting. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, Alon Y. Halevy, Zachary G. Ives, and AnHai Doan (Eds.). ACM, "", 76–85.

[32] Chad Sharp, Jelle van Assema, Brian Yu, Kareem Zidane, and David J. Malan. 2020. An Open-Source, API-Based Framework for Assessing the Correctness of Code in CS50. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2020, Trondheim, Norway, June 15-19, 2020*, Michail N. Giannakos, Guttorm Sindre, Andrew Luxton-Reilly, and Monica Divitini (Eds.). ACM, "", 487–492. https://doi.org/10.1145/3341525.3387417

[33] Thomas Staubitz, Hauke Klement, Ralf Teusner, Jan Renz, and Christoph Meinel. 2016. CodeOcean - A versatile platform for practical programming excercises in online environments. In *2016 IEEE Global Engineering Education Conference, EDUCON 2016, Abu Dhabi, United Arab Emirates, April 10-13, 2016*. IEEE, "", 314–323. https://doi.org/10.1109/EDUCON.2016.7474573

[34] topcoder. 2023. https://www.topcoder.com. Accessed: 2024-05-01.

A User Study Questions

- **Q1.** In your opinion, was the use of the code submission/evaluation system in this course helpful?
- **Q2.** In your opinion, were the dashboards for the labs and the project helpful?
- **Q3.** In your opinion, were the hints provided in the feedback for Lab 2 submissions helpful?
- **Q4.** In your opinion, was the Valgrind report in the feedback for the labs and project helpful?
- **Q5.** In your opinion, was the Lizard report in the project feedback helpful?
- **Q6.** If you would like to leave any comments or suggestions, please enter them here.