

Article

Bridging the Gap between Project-Oriented and Exercise-Oriented Automatic Assessment Tools

Bruno Pereira Cipriano ^{1,*} , Bernardo Baltazar ¹ , Nuno Fachada ^{1,2} , Athanasios Vourvopoulos ³ 
and Pedro Alves ¹ 

¹ ECATI, Lusófona University, Campo Grande, 376, 1749-024 Lisboa, Portugal; a21704025@alunos.ulht.pt (B.B.); nuno.fachada@ulusofona.pt (N.F.); pedro.alves@ulusofona.pt (P.A.)

² COPELABS, Lusófona University, Campo Grande, 376, 1749-024 Lisboa, Portugal

³ Institute for Systems and Robotics—Lisboa, Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais, 1049-001 Lisboa, Portugal; athanasios.vourvopoulos@tecnico.ulisboa.pt

* Correspondence: bcipriano@ulusofona.pt

Abstract: In this study, we present the DP Plugin for IntelliJ IDEA, designed to extend the Drop Project (DP) Automatic Assessment Tool (AAT) by making it more suitable for handling small exercises in exercise-based learning environments. Our aim was to address the limitations of DP in supporting small assignments while retaining its strengths in project-based learning. The plugin leverages DP's REST API to streamline the submission process, integrating assignment instructions and feedback directly within the IDE. A student survey conducted during the 2022/23 academic year revealed a positive reception, highlighting benefits such as time efficiency and ease of use. Students also provided valuable feedback, leading to various improvements that have since been integrated into the plugin. Despite these promising results, the study is limited by the relatively small percentage of survey respondents. Our findings suggest that an IDE plugin can significantly improve the usability of project-oriented AATs for small exercises, informing the development of future educational tools suitable for mixed project-based and exercise-based learning environments.

Keywords: automated assessment; computer science education; programming education; hci; IntelliJ IDEA



Citation: Cipriano, B.P.; Baltazar, B.; Fachada, N.; Vourvopoulos, A.; Alves, P. Bridging the Gap between Project-Oriented and Exercise-Oriented Automatic Assessment Tools.

Computers **2024**, *13*, 162. <https://doi.org/10.3390/computers13070162>

Academic Editor: Stelios Xinogalos

Received: 30 May 2024

Revised: 25 June 2024

Accepted: 27 June 2024

Published: 30 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Automatic assessment tools (AATs) are software systems that automatically grade programming assignments. These tools are common in computer science and engineering courses, providing recognized benefits for students, as automatic feedback improves their autonomy. Additionally, they are beneficial for educators, as they reduce assessment time and promote grade consistency [1–4]. However, many of these tools are course- or institution-specific and are not easily adaptable to other scenarios due to a lack of documentation or unavailability of the tool's binary or source code, among other issues [5,6]. Furthermore, existing commercial tools (e.g., repl.it and HackerRank) are usually web-based systems, which do not align well with real-world scenarios where integrated development environments (IDEs), such as NetBeans and IntelliJ IDEA, are prevalent [7,8].

Some courses use an exercise-based approach, requiring students to solve a number of small assignments (e.g., weekly) [9,10], while others use a project-based methodology, where students develop larger projects that can take multiple weeks or even months to complete [10,11]. This duality is also observable in many AATs available in the market or described in the literature. For example, most web-based AATs are better suited for small exercises, since larger projects usually require multiple files. Although some web editors support multiple files, we argue that this is not ideal from a usability and tool perspective—not to mention their inadequacy in preparing students for the job market, as they create artificial environments that do not compare with the software development

workflows commonly found in professional settings. On the other hand, AATs designed for project assessment often require impractical submission legwork for the grading of small exercises.

In turn, several courses follow a mixed approach, using both small exercises and larger projects as teaching and assessment methods [10,12,13]. This is the case for several courses taught by the authors, where automatic assessment is performed with Drop Project (DP), an AAT originally designed to support the development and grading of larger projects. Furthermore, educators using this teaching approach sometimes have to use multiple AATs within the same course [13], requiring both students and teachers to interact with multiple systems.

DP supports submissions via web upload or using a Git repository, requiring the inclusion of an AUTHORS.txt file with the name of the student or students. Additionally, in the web upload method, code and supporting files must be submitted in a ZIP file. Therefore, DP is not ideal from a usability perspective in scenarios involving smaller exercises. For example, one of our courses employs a ‘mini-test’ in which students are asked to implement several functions in one or—at most—two files. These ‘mini-tests’ are taken during laboratory classes, and students have up to one hour to solve them. Students submit their solutions through the web upload method, since version control with Git is yet to be introduced. Also, given the pressure of being assessed, students sometimes make small, time-wasting mistakes, potentially resulting in grades that do not reflect their actual level of knowledge. Examples of such mistakes include creating the AUTHORS.txt file with an incorrect format or problems creating the ZIP file, which easily lead to precious minutes being wasted.

To streamline this procedure and bridge the gap between a project-oriented AAT and its exercise-oriented utilization, we propose the DP Plugin for the IntelliJ IDEA IDE. This plugin automates submission-related perfunctory tasks, such as creating the required documentation files and compressing and uploading the solution, while displaying assignment instructions and submission reports directly within the IDE. The plugin reduces the cost of context switching, allowing students to focus on the exercises without leaving the IDE. The proposed plugin was developed using an agile methodology, with students directly involved in the coding and testing process. This work aims to address whether it is possible to improve the usability of a project-oriented AAT via an IDE plugin, enhancing its capability in assisting students in solving small exercises.

This paper is organized as follows. In Section 2, we review existing literature, highlighting how several popular AATs manage project- and exercise-based assessments. Section 3 describes the materials used in this work. Specifically, Section 3.1 examines the DP project-oriented AAT and its limitations regarding small exercises, while Section 3.2 introduces the DP Plugin for IntelliJ IDEA, providing technical details and illustrating how it aims to address the aforementioned issues with DP by keeping students engaged within the IDE and minimizing cursory submission tasks. Section 4 describes the research methodology, which consists of a student survey to assess the plugin’s impact on students’ assignment submission experiences. The results of this survey are presented in Section 5, while Section 6 offers a discussion of these results, considering their implications and potential limitations. Finally, Section 7 summarizes our findings and offers directions for future research.

2. Related Work

Project-based (PB) learning describes a course structure in which students work towards finding solutions to real-world problems [10,14]. These problems are represented as projects that often span the entire course duration. On the other hand, in exercise-based (EB) courses, students solve questions with predetermined answers (i.e., small coding questions) [10]. In EB courses, individual assignments have shorter durations and are typically less creative than those used in PB learning [10]. PB learning is not a new idea, and research has shown that engineering students tend to be more motivated when working with real-life case studies and challenging open-ended questions [15,16], which are more easily

implemented with projects rather than small exercises. A more recent study evaluated PB learning practices in several engineering degrees and noted that while students who participate in PB learning are generally motivated and demonstrate better teamwork and communication skills, they also display a less rigorous understanding of engineering fundamentals [17]. The same study concluded that a mixed approach of traditionally lectured courses with both PB and EB components might be the best way to satisfy industry needs without sacrificing knowledge of engineering fundamentals.

The use of AATs is common in computer science courses for the automatic assessment of PB and/or EB assignments [18], with a variety of available commercial and academic systems. However, many of these are proprietary and/or not packaged in a production-ready fashion, making their adoption costly or impractical. Previous research has shown that some of these AATs were created as part of a thesis or are sufficiently developed to study a specific research question or support the needs of a particular course but are generally not suitable for widespread deployment [3,5]. However, the situation appears to be improving, with some of the more recent developments being open-source [3], which could lead to more widespread deployment of these tools.

Despite these issues, some relevant tools should be considered when adopting an AAT. For instance, repl.it is an online tool where teachers can create assignments that students solve directly in its web-based IDE [19]. HackerRank [?] is a similar platform, originally developed for competitive programming and programming interviews. While such services might be practical for small exercises, they are not ideal for mid- to large-sized, multi-file projects, as they offer environments far removed from common professional development workflows [7]. Another relevant system is Mooshak [21,22], which supports programming contests and allows submissions via the upload of a single file, making it more appropriate for validating small exercises. However, Raccode, Mooshak's companion plugin, allows students to submit a code file directly from the NetBeans IDE [7]. Web-CAT [23] is an AAT where solutions can be uploaded via a browser or submitted through a NetBeans IDE plugin, making it apparently appropriate for handling both small exercises and projects. However, Web-CAT's code is spread over various repositories, and the most recent binary was last updated in November 2008. Although Web-CAT's documentation seems comprehensive, at the time of writing, multiple links on its GitHub-hosted website were broken, making it hard to recommend. A more recent AAT, check50 [24], is a command-line tool that uses a domain-specific language to write tests for student code and does not enforce any specific structure (e.g., mandatory functions). This makes it inappropriate for large projects, which need to be well-structured following software engineering best practices. The Canary Framework (CF) AAT follows a continuous integration approach, mimicking real professional settings [8], and allows for interaction via commits to a Git repository. This approach is more suitable for PB assessments. However, the need to teach Git skills might not be feasible in certain academic contexts, such as introductory-level courses. Additionally, CF is not a single integrated piece of software but a set of six independent tools that can be combined to achieve the intended goals, and the source code for several of these tools does not seem to be available. Finally, DP is an AAT originally developed to support the development of projects [2], offering submissions through ZIP file upload or via a Git repository, making it less than ideal for smaller exercises, although the plugin discussed in this paper aims to address that issue. DP is fully open-source and can be readily deployed using a Docker image, for example [25].

The discussed tools and their main characteristics, including DP with the proposed plugin, are summarized in Table 1.

Table 1. Summary and main characteristics of the discussed AATs, including Drop Project featuring the proposed plugin. “Open?” indicates whether the tool is open-source. “EB?” and “PB?” refer to suitability for exercise-based and project-based approaches, respectively. The term “web-IDE” is used as a shorthand for “online code editors”.

Tool	Open?	Installation	EB?	PB?	Beginner-Friendly?
repl.it	No	N/A	Yes	No	Yes (web-IDE)
HackerRank	No	N/A	Yes	No	Yes (web-IDE)
Mooshak	Yes	Pre-compiled JAR file	Yes	No	Yes (submissions via upload)
Web-CAT	Yes	Pre-compiled WAR file (from 2008); source install requires compilation of multiple sub-systems	Yes ^a	Yes	Yes (submissions via upload and IDE plugin)
check50	Yes	Requires installation of multiple tools	Yes	No	Yes
Canary framework	Partially ^b	Requires installation of multiple tools	No	Yes	No (only Git submissions)
VPL	Yes	Must be installed by Moodle’s administrator	Yes	No	Yes (submissions via upload and web-IDE)
Drop Project	Yes	WAR file compiled from latest sources/one-click deploy to Heroku (single tool)	Yes ^c	Yes	Yes (submissions via upload and Git)

^a With EB plugin for NetBeans IDE. ^b Since the code for half the repositories is not available. ^c With the proposed plugin for IntelliJ IDEA.

From our investigation, the AAT that comes closest to being suitable for both small exercises and projects is Web-CAT. However, it does not appear to be easy to install, deploy, and/or use due to a lack of updated binaries and documentation. In summary, all of the AATs discussed so far have one or more types of limitations, which can be summarized as follows:

- They are difficult to install, deploy, and/or use;
- They are inappropriate for beginner students, e.g., by only allowing submissions using Git;
- They are unsuitable for assessing and evaluating multi-file projects (i.e., PB learning);
- They are unsuitable for assessing and evaluating small exercises (i.e., EB learning).

DP suffers from the last of these issues, making it an imperfect choice for small exercises. The proposed DP Plugin for IntelliJ IDEA aims to solve this problem, fostering the use of a single assessment tool in mixed courses (EB and PB), minimizing tool switching, and allowing teachers and students to save time and avoid problems with cursory tasks.

3. Materials

3.1. Drop Project

DP is an open-source AAT designed and implemented to support the evaluation of academic software projects [2]. It is a mature tool, used in multiple courses for over five years. The following list, adapted from [2], highlights a subset of DP’s features.

- Supports individual and group submissions;
- JVM and Maven-based, supporting Java and Kotlin out of the box, with easy integration of other JVM-based languages such as Clojure and Scala;
- Features a waiting time between submissions, termed “Cool-off period”, which can be set to limit students’ trial-and-error attempts [26];
- Optional assignment-specific access-control lists to ensure only authorized students can access certain assignments;
- Supports web upload and Git submissions, allowing teachers to adapt the submission procedure to the students’ experience level;
- Teachers can define assignments and tests using all strategies available in JUnit (e.g., unit tests, input/output matching, and so on). Tests can be public (students receive feedback) or private/hidden (students receive no feedback);
- Can run and assess student-defined unit tests, as well as measure the respective coverage;

- Besides performing functional validations, DP can check the quality of students' code using teacher-defined style rules. It can also restrict keywords, like forbidding loops in exercises requiring recursive implementations;
- Teachers can download students' submissions in Maven format and easily import them into an IDE, facilitating the debugging process;
- Students' results can be exported to a CSV file.

These and other functionalities make DP a good option when choosing an AAT for PB learning. However, as mentioned earlier, it is not ideal for supporting students with small exercises due to the legwork required by both submission types.

On a more technical note, DP also provides a REST API that can be used to integrate external tools (<https://deisi.ulusofona.pt/drop-project/swagger-ui/>, accessed on 26 June 2024). This API provides functionalities such as assignment listing, assignment presentation, and solution submission. The API implements the same security and access control measures as DP itself. For example, the 'List of public assignments' endpoint only returns assignments to which all students have access. To access a private assignment, the student must supply the respective assignment ID, and the API will check if the given student has access to that assignment before returning its contents. If an assignment has a defined "cool-off" period, the API will check if the time for a certain student has passed before allowing that student to perform new submissions.

A video demonstration of DP's workflow from both the teacher's and the student's points of view is available [27].

3.2. DP Plugin for IntelliJ IDEA

The DP Plugin for IntelliJ IDEA presented in this paper leverages DP's REST API, providing an abstraction layer on top of DP's web upload submission method, simplifying parts of this process. Figure 1 presents an overview of how the various components integrate and communicate with each other.

The plugin is integrated into IntelliJ IDEA's main window (Figure 2), allowing students to simultaneously view their code, the assignment instructions (Figure 3), and the assessment results of their latest submission (Figure 4). The plugin uses assignments created for DP [2]. Therefore, there is no additional assignment creation overhead for teachers already familiar with DP, while students can still use DP's standard submission methods if they so desire.

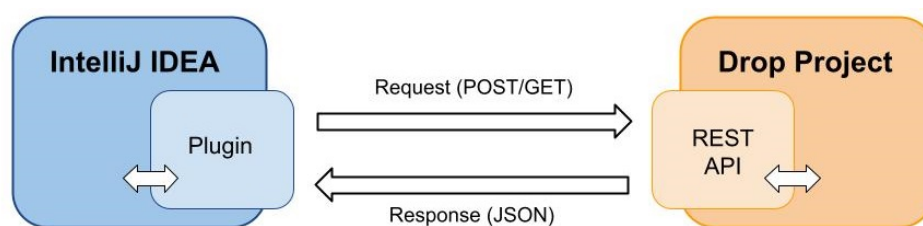


Figure 1. Overview of the solution's data flow. The plugin integrates with IntelliJ IDEA and connects to DP via its REST API.

The plugin also comes with its own security measures. For instance, the feature that updates the assignment list will only make an API request if a specific number of seconds has passed since the last update request.

The plugin was developed during the 2022/23 academic year. In both semesters, development versions of the plugin were deployed and distributed among students during selected classes of specific courses to test and obtain feedback. These efforts yielded a production-ready version of the plugin, which is available for download at IntelliJ's marketplace (<https://plugins.jetbrains.com/plugin/21870-drop-project>, accessed on 26 June 2024). The tool's source code is also available on GitHub (<https://github.com/drop-project>

edu/Drop-Project-for-Intellij-Idea, accessed on 26 June 2024). A video demonstration of the proposed plugin is available [28].

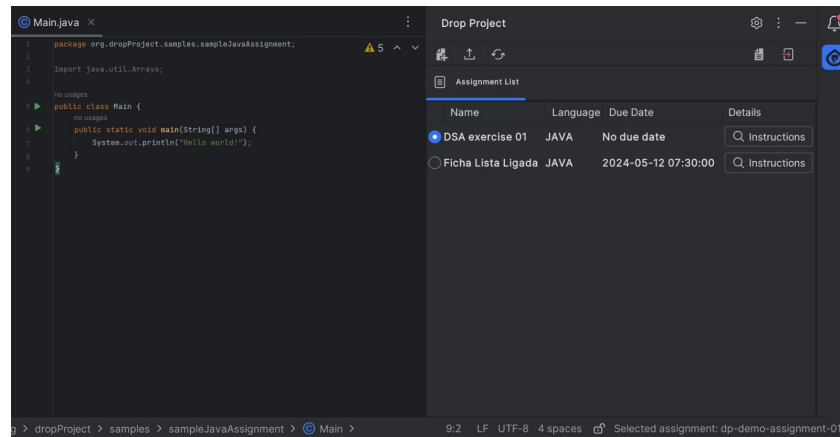


Figure 2. The plugin integrated into IntelliJ IDEA's main window. The plugin lists the assignments available to the student. The 'Instructions' button allows students to access the instructions for each available assignment (see Figure 3).

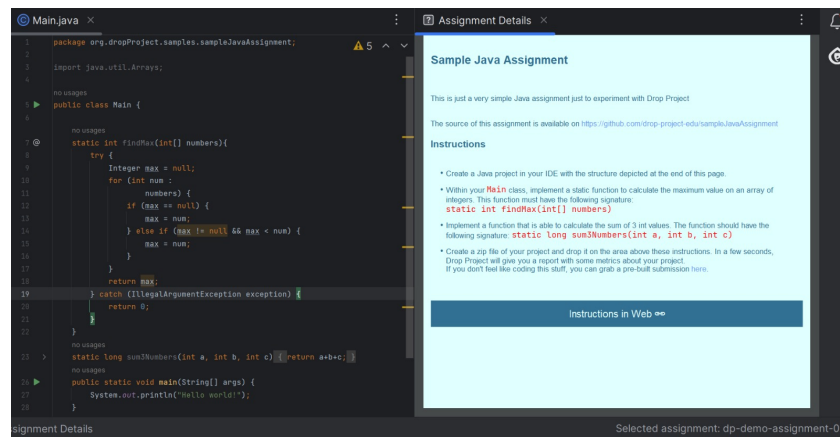


Figure 3. The assignment instructions inform students of the tasks they need to perform. Students can see their code and the assignment instructions in the same window, reducing the need for context switching.

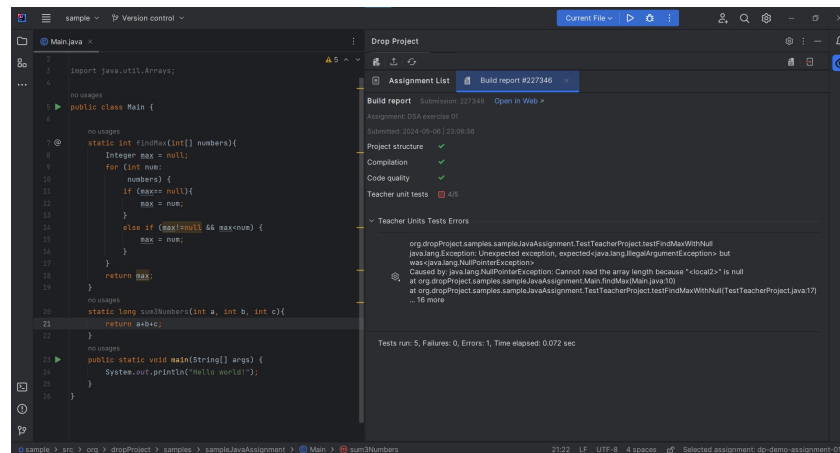


Figure 4. The submission report informs students of their progress and presents them with any unit test failures. Students can see their code and its issues in the same window.

4. Research Methods

A student survey was conducted to evaluate the plugin in the context of a Data Structures and Algorithms course during the second semester of the 2022/23 academic year, i.e., during the final stages of the plugin's development. The survey comprised both qualitative and quantitative questions. The questions aimed to assess students' perceptions regarding various aspects of the plugin, including its impact on reducing submission time, students' confidence in the plugin, the value of having specific reporting functionalities integrated into the IDE, and the psychological impact of using the plugin in an evaluation environment. The research context is further detailed in Section 4.1, while the survey questions are presented in Section 4.2.

4.1. Research Context

The Data Structures and Algorithms course follows a mixed approach, utilizing both PB and EB learning methodologies, with DP used for automatic assessment. All students in this course have interacted with DP, using its web interface, as DP is also used in the previous programming course of the curriculum (i.e., Programming Fundamentals).

This course employs multiple evaluation mechanisms. One of them—weekly assignments—aims to promote student participation and engagement, contributing 5% to the final course grade. The course also includes 'mini-tests', which are one-hour assessments where students implement various related data structures and functions under the supervision of a teacher. Each 'mini-test' has a weight of 6.75% in the final course grade.

4.2. Survey Questions

Students were divided into the following two main groups, with specific questions for each group:

Adopters: Students who identified as regular plugin users;

Non-adopters: Students who reported not having used the plugin.

Students who self-identified as regular plugin users (i.e., adopters) were asked multiple plugin-specific questions, with answers given on a 5-level Likert scale, as shown in Table 2. Considering that students who self-identified as regular plugin users may not have actually used the plugin during the 'mini-test' assessment, questions 7 and 8 in Table 2 were only available for students who did so.

Table 2. Survey questions for students who adopted the plugin. Questions 1–6 were answered by all adopter students, while questions 7–8 were only answered by students who used the plugin during the second 'mini-test'. Questions were answered on a 5-level Likert scale.

ID	Question
1	Rate the plugin's effectiveness in simplifying the DP submission process.
2	Rate the plugin in terms of saving time and/or effort when compared to the traditional method (web upload).
3	Rate the relevance of the automatic creation of the AUTHORS.txt file and the automatic submission of the ZIP file with the '/src' folder and AUTHORS.txt.
4	Rate your confidence in submitting exercises through the plugin.
5	Rate the positive effects on your productivity from the ability to see assignment instructions directly in the IDE.
6	Rate the positive effects on your productivity from the ability to view a submission's build report directly in the IDE.
7	Rate the contribution of using the Plugin in reducing anxiety and/or nervousness during the 'mini-test'.
8	Rate the contribution of using the Plugin in improving your performance during the 'mini-test'.

To gain additional insights into plugin usage (or lack thereof) during the semester, three open-ended questions (OE1–OE3), as shown in Table 3, were posed to specific student groups. First, to understand why some students who self-identified as plugin users (i.e., adopters) did not use the plugin during the 'mini-test', we asked for their reasons (open-ended question OE1, Table 3). Second, all adopters answered a qualitative question to leave any relevant comments (open-ended question OE2, Table 3). Finally, we asked the

students who had not tried the plugin (i.e., non-adopters) for their reasons for not doing so (open-ended question OE3, Table 3).

Table 3. Open-ended questions posed to students in the specified target groups.

ID	Question	Target Student Group
OE1	If you did not use the DP Student Plugin in the ‘mini-test’: explain why.	Adopters that did not use the plugin during the ‘mini-test’.
OE2	Leave any comment that you find relevant.	Adopters.
OE3	If you do not use the plugin, explain why.	Non-adopters.

5. Results

The survey was conducted during the last two weeks of the second semester of the 2022/23 academic year. Out of a total of 154 students, 35 participated in the survey (22.7%). Among these respondents, 20 were identified as adopters of the plugin, while the remaining 15 were non-adopters. Furthermore, of the 20 adopters, half (10 students) used the plugin during the ‘mini-test’, whereas the other half did not. Figure 5 summarizes this breakdown.

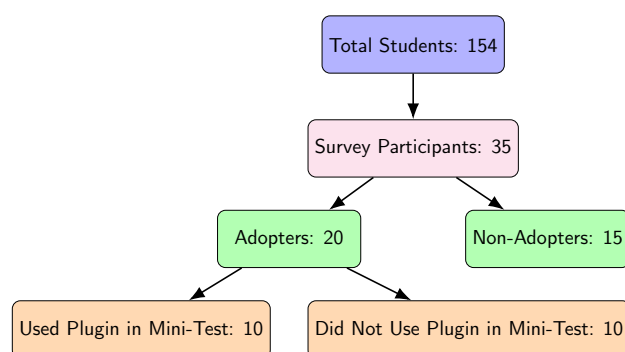


Figure 5. Hierarchical breakdown of student participation and plugin adoption based on the survey responses.









5.1. Impact of the Plugin on Adopters

Table 4 shows the distribution and summary statistics for the replies given by adopters of the plugin. Replies to questions 1–6, which tend toward the highest possible evaluation, indicate that students found the plugin useful. Questions 1–4 have mean scores of at least 4.5, all with modes equal to 5. However, questions 5 and 6—regarding the assignment instructions and submission report displayed within the IDE—had slightly lower scores, with averages of 3.8 (mode 4) and 4.1 (mode 5), respectively. Students do not seem to greatly value having this information within the IDE, particularly the assignment instructions. We believe this is due to the richer formatting of these instructions in the web environment, which displays information such as required function prototypes in a colored format.

Question 7 asked if the plugin helps reduce students’ anxiety and/or nervousness during the ‘mini-test’. The mean score given by the participants was 3.6, indicating a generally positive response towards the plugin’s efficacy. The standard deviation of 1.173 and the histogram distribution denote some variability in the participants’ experiences and perceptions. Interestingly, the data were bimodal, with modes at both 3 and 5. This indicates that while a significant number of participants felt the plugin had a moderate effect (rating of 3), there was also a substantial group that found it highly effective (rating of 5) in reducing anxiety and nervousness. As for question 8, which focused on the plugin’s effect on the students’ performance, the mean response was 3.7, suggesting that many participants felt the plugin had a positive impact. The longer-tailed distribution (standard deviation of 1.159) indicates a moderate spread in the responses, meaning that while many participants agreed with the statement, there was still some variability in opinions. Notably, both options 4 and 5 were selected with the same frequency above all other options. This

indicates that a significant number of participants either agreed or strongly agreed with the statement, highlighting the perceived effectiveness of the plugin during the 'mini-test'.

Table 4. Histogram and summary statistics for the questions posed to plugin adopters. Questions 1–6 were answered by the 20 students who reported using the plugin, while questions 7–8 were only answered by the 10 students who reported using the plugin during the second 'mini-test'. All questions were answered using a 5-level Likert scale. Histograms are normalized to the maximum count value for visualization purposes.

ID	Histogram	Count	Mean	Std. Dev.	Mode
1		20	4.6	0.680	5
2		20	4.5	0.827	5
3		20	4.3	0.978	5
4		20	4.6	0.598	5
5		20	3.8	1.105	4
6		20	4.1	1.160	5
7		10	3.6	1.173	3; 5
8		10	3.7	1.159	4; 5

5.2. Reasons for Not Using the Plugin during the 'Mini-Test'

Nine out of ten adopter students who did not use the plugin during the 'mini-test' replied to question OE1 (Table 3). These replies were analyzed qualitatively to find common reasons for not using the plugin during the assessment.

The predominant topic found in the responses centered around issues arising from the compatibility between the version of IntelliJ IDEA that certain students were using and the plugin itself. This concern was raised by three students, leading them to opt against utilizing the plugin during the assessment. The remaining six students reported different issues that could not be grouped into any specific topic. For example, one student mentioned, "I didn't think it was necessary and I wanted to be confident that I had sent it correctly and be able to see the results and errors immediately after the page loaded". Another student commented, "I had to log in, and the instructions in the IDE are not as optimized and easy to read", while another student indicated, "I changed computers and forgot to install it". Notably, one student commented on the instructions' readability, which is consistent with the lower mean result of the respective qualitative question (i.e., question 5, mean = 3.8).

5.3. Open-Ended Comments about the Plugin

A total of 10 students (out of 20 adopters) answered question OE2 (Table 3). Again, students' responses were analyzed qualitatively to find common topics. The following three recurring topics were found: (1) positive comments about the plugin (e.g., "useful", "simplifies", "saves time") (four students), (2) suggestions for improving the formatting of the assignment instructions (two students), and (3) suggestions to avoid the need to log in again when changing assignments and/or when re-opening the IDE (two students). It should be noted that the suggestion about saving login data between uses has since been implemented.

5.4. Reasons for Not Adopting the Plugin

Thirteen out of 15 students replied to question OE3 (Table 3). A qualitative analysis highlighted the following two main topics: (1) "I'm not interested in it"/"I don't feel the need" (six students) and (2) "I don't know how to" (four students). An example reply that falls under the first topic is, "I never thought it was necessary, although I give credit and think it is useful", while an example response for the second topic is, "I don't understand very well how to use it. I think a video explanation would be useful".

6. Discussion and Limitations

As the evaluation survey shows, the tool was well received by our students, who identified a number of positive effects, such as time savings. Students also provided interesting suggestions, some of which have already been implemented, such as the plugin remembering login details. Our findings suggest that an IDE plugin does improve the usability of a project-oriented AAT for small exercises, thus informing the design of future AATs suitable for mixed learning environments combining PB and EB assessment. The positive reception of the plugin highlights its potential to enhance the learning experience by streamlining workflows and reducing the time required for administrative tasks (e.g., ZIP file creation), allowing students to focus more on learning the material.

Results also show that there are two main types of non-adopters, namely (1) students who are happy and/or comfortable using DP's web interface to the point that they are not open to trying new interaction formats and (2) students who need some help in order to be able to use the plugin. For the former, we believe they are acting out of habit, since their initial exposure to DP was through the web interface. This will likely change in the future, as students also have the plugin available in the preceding Programming Fundamentals course. Regarding the latter group, who need more help, a video explaining the plugin's installation and usage has since been made available [28].

Moreover, the integration of student suggestions into the plugin development process demonstrates the value of student feedback in creating more effective educational tools. The implementation of simple features such as remembering login details shows how even minor enhancements can improve the overall user experience, potentially leading to a more user-friendly and efficient learning environment. Additionally, student feedback raises awareness of the need for documentation (e.g., demonstration videos) to help learners use these educational tools effectively.

Given these results, we believe the proposed IDE plugin effectively extends DP to be a more practical AAT for small exercises, bridging the gap between its PB roots and its EB usage. As discussed in Section 2, other AATs have proposed IDE plugins. However, Mooshak's IDE plugin—Raccode—simply enhances the existing EB capabilities of Mooshak, itself an EB AAT. Web-CAT, a PB-oriented AAT, does provide an IDE plugin for improving its EB capabilities. However, the issues with Web-CAT—poorly organized code, out-of-date binaries, and missing documentation—make it a difficult choice as a modern and versatile AAT.

Nonetheless, our work presents several limitations. First, while the survey indicates that the majority of students find the plugin useful and believe it improves their workflows, the number of students who answered the survey was relatively small, which presents some threats to validity. Also, since participation was optional, there might have been some self-selection bias, as students who opted to participate in the survey might have been the ones more interested or motivated to discuss the plugin. However, we do not believe this to be the case, since approximately 42.5% of the responses were from students who did not use the plugin.

Furthermore, a student survey may not be sufficient to accurately measure the pedagogical impact of the proposed plugin. To gain a more comprehensive understanding and increase confidence in the plugin's pedagogical value, it would be beneficial to complement the current survey data with other research methods, such as direct classroom observations and analysis of students' academic results over time.

7. Conclusions

In this paper, we proposed the DP Plugin for IntelliJ IDEA, a companion tool for the DP AAT, with the goal of making it more apt to handle small exercises used in EB learning. As the evaluation survey shows, the tool was well received by our students, who identified a number of positive effects, such as time savings. Students also provided interesting suggestions, some of which have already been implemented, such as the plugin remembering login details. Our findings suggest that an IDE plugin does improve the

usability of a project-oriented AAT for small exercises, thus informing the design of future AATs suitable for mixed learning environments combining PB and EB assessment.

Despite the promising feedback on the plugin's effectiveness, our study has some limitations. The small number of survey participants raises concerns about validity, and the optional nature of the survey could lead to self-selection bias. However, the significant number of non-adopters among respondents helps mitigate this concern. Additionally, a student survey alone may not fully capture the plugin's educational impact. Future research should include direct classroom observations and long-term academic performance analysis to provide a more thorough assessment of the plugin's pedagogical value.

We are currently integrating the plugin with OpenAI's GPT [29] to allow students to promptly get help with their assignments without leaving their IDE [30]. The plugin will also collect usage interaction data, with student consent. We plan to harvest these data with the goal of preparing, curating, and publishing a dataset that will allow the computer science education community to study these interactions between students and large language models.

Author Contributions: Conceptualization, B.P.C. and P.A.; funding acquisition, A.V. and P.A.; methodology, B.P.C. and P.A.; software, B.P.C., B.B. and P.A.; validation, B.P.C., N.F. and P.A.; formal analysis, B.P.C., N.F. and P.A.; investigation, B.P.C.; data curation, B.P.C.; writing—original draft preparation, B.P.C. and N.F.; writing—review and editing, B.P.C., N.F. and P.A.; visualization, B.P.C. and N.F.; supervision, N.F., A.V. and P.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundação para a Ciência e a Tecnologia (FCT) through the LARSyS—FCT Project (DOI: 10.54499/LA/P/0083/2020, 10.54499/UIBP/50009/2020, and 10.54499/UIDB/50009/2020) and COPELABS—FCT Project (DOI: 10.54499/UIDB/04111/2020). Author Nuno Fachada acknowledges his FCT grant (DOI: 10.54499/CEECINST/00002/2021/CP2788/CT0001).

Informed Consent Statement: Informed consent was obtained from all students involved in the study.

Data Availability Statement: A CSV file with the survey replies is available [31].

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AAT	Automatic Assessment Tool
API	Application Programming Interface
CSV	Comma-separated Values
DP	Drop Project
EB	Exercise-based
IDE	Integrated Development Environment
JVM	Java Virtual Machine
PB	Project-based
REST	Representational State Transfer

References

1. Enström, E.; Kreitz, G.; Niemelä, F.; Söderman, P.; Kann, V. Five years with kattis—Using an automated assessment system in teaching. In Proceedings of the 2011 Frontiers in Education conference (FIE), Rapid City, SD, USA, 12–15 October 2011; p. T3J-1.
2. Cipriano, B.P.; Fachada, N.; Alves, P. Drop Project: An automatic assessment tool for programming assignments. *SoftwareX* **2022**, *18*, 101079. [[CrossRef](#)]
3. Paiva, J.C.; Leal, J.P.; Figueira, Á. Automated Assessment in Computer Science Education: A State-of-the-Art review. *ACM Trans. Comput. Educ. (TOCE)* **2022**, *22*, 34. [[CrossRef](#)]
4. Luxton-Reilly, A.; Tempero, E.; Arachchilage, N.; Chang, A.; Denny, P.; Fowler, A.; Giacaman, N.; Kontorovich, I.; Lottridge, D.; Manoharan, S.; et al. Automated Assessment: Experiences From the Trenches. In Proceedings of the 25th Australasian Computing Education Conference, Melbourne, VIC, Australia, 30 January–3 February 2023; pp. 1–10.

5. Ihantola, P.; Ahoniemi, T.; Karavirta, V.; Seppälä, O. Review of recent systems for automatic assessment of programming assignments. In Proceedings of the 10th Koli Calling International Conference on Computing Education Research, Koli, Finland, 28–31 October 2010; pp. 86–93.
6. Keuning, H.; Jeuring, J.; Heeren, B. A systematic literature review of automated feedback generation for programming exercises. *ACM Trans. Comput. Educ. (TOCE)* **2018**, *19*, 3. [[CrossRef](#)]
7. Silva, A.; Leal, J.P.; Paiva, J.C. Raccode: An Eclipse Plugin for Assessment of Programming Exercises (Short Paper). In Proceedings of the 7th Symposium on Languages, Applications and Technologies (SLATE 2018), Guimaraes, Portugal, 21–22 June 2018; Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
8. Heckman, S.; King, J. Developing Software Engineering Skills using Real Tools for Automated Grading. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, MD, USA, 21–24 February 2018; pp. 794–799.
9. Offutt, J.; Ammann, P.; Dobolyi, K.; Kauffmann, C.; Lester, J.; Praphamontriphong, U.; Rangwala, H.; Setia, S.; Wang, P.; White, L. A novel self-paced model for teaching programming. In Proceedings of the Fourth (2017) ACM Conference on Learning@Scale, Cambridge, MA, USA, 20–21 April 2017; pp. 177–180.
10. Lenfant, R.; Wanner, A.; Hott, J.R.; Pettit, R. Project-Based and Assignment-Based Courses: A Study of Piazza Engagement and Gender in Online Courses. In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, Turku, Finland, 7–12 July 2023; pp. 138–144.
11. Lukefahr, H.; Watterson, C.; Roberts, A.; Carnegie, D.; Atkins, M. Project-Based Learning to Improve Core First-Year Engineering Courses. In Proceedings of the 2019 IEEE Frontiers in Education Conference (FIE), Covington, KY, USA, 16–19 October 2019; pp. 1–7.
12. Bridson, K.; Atkinson, J.; Fleming, S.D. Delivering Round-the-Clock Help to Software Engineering Students Using Discord: An Experience Report. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1, Providence, RI, USA, 3–5 March 2022; pp. 759–765.
13. Edwards, S.H.; Murali, K.P. CodeWorkout: Short Programming Exercises with Built-in Data Collection. In Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, Bologna, Italy, 3–5 July 2017; pp. 188–193.
14. Schwalm, J.; Tylek, K.S. Systemwide Implementation of Project-Based Learning: The Philadelphia Approach. *Afterschool Matters* **2012**, *15*, 1–8.
15. Baillie, C.; Fitzgerald, G. Motivation and attrition in engineering students. *Eur. J. Eng. Educ.* **2000**, *25*, 145–155. [[CrossRef](#)]
16. Grotta, A.; Prado, E.P.V. Benefits of the project-based learning to cope with computer programming education: A systematic literature review. In Proceedings of the PBL2019 Immersive Virtual International Conference, Virtual, 12–13 July 2019; pp. 12–13.
17. Mills, J.; Treagust, D. Engineering Education—Is Problem-based Or Project-Based Learning The Answer? *Australas. J. Eng. Educ.* **2003**, *3*, 2–16.
18. Insa, D.; Silva, J. Semi-Automatic Assessment of Unrestrained Java Code: A Library, a DSL, and a Workbench to Assess Exams and Exercises. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, Vilnius, Lithuania, 4–8 July 2015; pp. 39–44.
19. Škorić, I.; Orehovački, T.; Ivašić-Kos, M. Exploring the acceptance of the web-based coding tool in an introductory programming course: A pilot study. In Proceedings of the International Conference on Human Interaction and Emerging Technologies, Paris, France, 27–29 August 2020; pp. 42–48.
20. Maguire, P.; Maguire, R.; Kelly, R. Using automatic machine assessment to teach computer programming. *Comput. Sci. Educ.* **2017**, *27*, 197–214. [[CrossRef](#)]
21. Leal, J.P.; Silva, F. Mooshak: A Web-based Multi-site Programming Contest System. *Softw. Pract. Exp.* **2003**, *33*, 567–581. [[CrossRef](#)]
22. Guerreiro, P.; Georgouli, K. Combating Anonymously in Populous CS1 and CS2 Courses. *ACM SIGCSE Bull.* **2006**, *38*, 5. [[CrossRef](#)]
23. Edwards, S.H.; Perez-Quinones, M.A. Web-CAT: Automatically Grading Programming Assignments. *SIGCSE Bull.* **2008**, *40*, 328. [[CrossRef](#)]
24. Sharp, C.; van Assema, J.; Yu, B.; Zidane, K.; Malan, D.J. 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, Trondheim, Norway, 15–19 June 2020; ITiCSE '20, pp. 487–492. [[CrossRef](#)]
25. Boettiger, C. An introduction to Docker for reproducible research. *ACM SIGOPS Oper. Syst. Rev.* **2015**, *49*, 71–79. [[CrossRef](#)]
26. Baniassad, E.; Zamprogno, L.; Hall, B.; Holmes, R. Stop the (autograder) insanity: Regression penalties to deter autograder overreliance. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, Virtual, 13–20 March 2021; pp. 1062–1068.
27. Cipriano, B.P. How to Use Drop Project to Create Programming Assignments. 2021. Available online: <https://www.youtube.com/watch?v=65IwIBuMDIE> (accessed on 26 June 2024).
28. Baltazar, B. Drop Project Student Plugin. 2023. Available online: <https://www.youtube.com/watch?v=N2YUvU3Lxm4> (accessed on 26 June 2024).
29. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual, 6–12 December 2020; pp. 1877–1901.

30. Cipriano, B.P. Towards the Integration of Large Language Models in an Object-Oriented Programming Course. In Proceedings of the 2024 Innovation and Technology in Computer Science Education, Milan, Italy, 8–10 July 2024. [[CrossRef](#)]
31. Cipriano, B.P.; Baltazar, B.; Alves, P.; Fachada, N. Drop Project Student Plugin for IntelliJ IDEA—Evaluation Survey. 2023. Available online: <https://zenodo.org/records/8432997> (accessed on 26 June 2024).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.