SoftwareX 18 (2022) 101079

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

Drop Project: An automatic assessment tool for programming assignments

Bruno Pereira Cipriano *, Nuno Fachada, Pedro Alves

Lusófona University, COPELABS, Campo Grande, 376, Lisbon, Portugal

ARTICLE INFO

Article history:
Received 24 July 2021
Received in revised form 10 March 2022
Accepted 1 April 2022

Keywords:
Automated assessment
Computer science education
Programming education
Unit testing

ABSTRACT

Automated assessment tools (AATs) are software systems used in teaching environments to automate the evaluation of computer programs implemented by students. These tools can be used to stimulate the interest of computer science students in programming courses by providing quick feedback on their work and highlighting their mistakes. Despite the abundance of such tools, most of them are developed for a specific course and are not production-ready. Others lack advanced features that are required for certain pedagogical goals (e.g. Git integration) and/or are not flexible enough to be used with students having different computer literacy levels, such as first year and second year students.

In this paper we present Drop Project (DP), an automated assessment tool built on top of the Maven build automation software. We have been using DP in our teaching activity since 2018, having received more than fifty thousand submissions between projects, classroom exercises, tests and homework assignments. The tool's automated feedback has allowed us to raise the difficulty level of the course's projects, while the grading process has become more efficient and consistent between different teachers. DP is an extensively tested, production-ready tool. The software's code and documentation are available in GitHub under an open-source software license.

© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Code metadata

Current code version	v0.9.4
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00136
Legal Code License	Apache License 2.0
Code versioning system used	Git
Software code languages, tools, and services used	Java, Kotlin, Maven, Spring, MySQL
Compilation requirements, operating environments & dependencies	JDK 11 or newer
If available Link to developer documentation/manual	https://drop-project-edu.github.io/drop-project/
Support email for questions	dropproject@ulusofona.pt

Software metadata

Current software version	v0.9.4
Permanent link to executables of this version	https://github.com/drop-project-edu/drop-project/releases/tag/v0.9.4
Legal Software License	Apache License 2.0
Computing platforms/Operating Systems	Linux, Microsoft Windows and macOS
Installation requirements & dependencies	JDK 11 or newer
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://drop-project-edu.github.io/drop-project/wiki/
Support email for questions	dropproject@ulusofona.pt

1. Motivation and significance

In this paper we propose Drop Project (DP), an Automated assessment tool (AAT) which addresses a number of issues found

* Corresponding author.
E-mail address: bcipriano@ulusofona.pt (Bruno Pereira Cipriano).

https://doi.org/10.1016/j.softx.2022.101079
2352-7110/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

in existing systems, and in which students can submit projects and get detailed feedback on their quality. We start by reviewing existing tools in Section 1.1, and based on this discussion, elaborate our motivation for proposing DP as a valid alternative in the AAT landscape in Section 1.2. The software is thoroughly described in Section 2, a number of examples are presented in Section 3, and the tool's deployment alternatives are highlighted in Section 4. The impact DP has had on our teaching practice is discussed in Section 5. Section 6 closes the paper, offering some conclusions.

1.1. Background

AATs are educational software systems that automate the assessment (or evaluation) of student-developed computer programs. AATs have been around since at least 1960 [1]. These tools allow students to check if their code solutions satisfy the requirements of the assignment with full autonomy, without assistance from the teacher. This is usually done by automating the compilation and execution of the student's code in a restricted environment and validating its correctness through output matching, unit testing, or other techniques [2,3]. Depending on the tool, students may or may not receive rich feedback about their submissions' failures. By providing quick feedback and highlighting mistakes, these tools can be used to foster computer science students' engagement in programming courses and increase their motivation [4]. Finally, most of these tools automatically grade student solutions.

The use of AATs is common in Computer Science courses [5], and there are a variety of commercial and academic systems. However, many of them are closed-source and/or not packaged in a production-ready fashion, making their adoption in other institutions impractical. This is partially confirmed by previous research, which has shown that some of these AATs are created as part of a thesis or for a particular course [3]. They are finished enough to study a certain research question or to support the needs of a particular course, but are generally not suitable for widespread deployment. However, there are some tools that partially overcome these issues and should be considered when trying to adopt an AAT.

A first example is repl.it, a proprietary online tool where teachers create assignments that can be evaluated with unit testing or with input/output matching, depending on the programming language [6]. Students interact with this platform using a web-based IDE.

Some universities use open-source systems originally developed for setting up programming contests. An example of this is Mooshak [7,8]. Students interact with Mooshak by uploading code files, which are then tested using input/output matching. Mooshak is geared towards the validation of assignments that can be solved with a single source file, and thus better suited for simple exercises.

Web-CAT [9] is an open-source automated grading system in which solutions can be uploaded via browser or submitted using a plugin for the NetBeans IDE. It is able to evaluate the code coverage of the student's own tests. Assignment creation implies the manual upload of the unit tests using web forms. Web-CAT's recommended installation method is a pre-built WAR file[1] that can be downloaded from a Sourceforge page.[2] However, it should be noted that the WAR file was last updated in 2008. The GitHub repositories[3] contain commits after that date, thus the WAR file does not seem to have the latest changes.

Another relevant tool is check50 [10], an open-source command line tool for assessing the correctness of students' code using tests written in a custom language. The same authors developed a set of separate companion tools for checking adherence to code style guidelines, performing the actual submissions, and plagiarism detection.

The Canary Framework is a set of six independent programs based on real workplace tools, such as GitHub, Jenkins, Checkstyle and Jacoco [11]. Like Web-CAT, Canary is able to measure the coverage of the student's test code. The Canary Framework only supports Git submissions, which may be a limiting factor for first-year students. Although Canary Framework's tools are announced as open-source, only half of these seem to have their source code available.[4]

Finally, some AATs are integrated out-of-the-box with learning management systems such as Moodle [12]. An example is CodeRunner, which allows students to write and submit code directly from the browser [13]. A distinguishable characteristic of this tool is that it is able to assess OpenGL code [14].

1.2. Motivation

DP is in production use since 2018 at the Lusófona University's Bachelor in Computing Engineering[5] (BSc), and its development was motivated by a number of factors. The first was the increase of enrolled students year over year, which was resulting in the grading process getting progressively harder and more time-consuming. It was particularly hard to ensure that all of the students' projects were subject to fair and equivalent assessment by the different teachers. Thus, we decided to use, adapt or implement such a system. Submission flexibility was an important software selection criteria, i.e., the tool should ideally accept both beginner-friendly web upload submissions and submissions via a Git repository, thus being useful for students and courses of all levels. Unfortunately, the AAT landscape did not offer a free and open-source, production-ready solution with this flexibility at the time, which prompted the initial development of DP.

DP is based on the principles and technology underlying most continuous integration servers. It relies on Maven [15], the most popular build automation tool for the languages that run on the Java Virtual Machine (JVM), such as Java, Kotlin or Scala. In DP, assignments must conform to the Maven standard format. Thus, they are easily shared and reused across several courses, eliminating a common complaint in AATs: the lack of a uniform format or structure that facilitates the portability of assignments and submissions (e.g., for sharing of assignments between teachers and institutions) [16].

In DP, feedback is based on multiple types of validations. For example:

- Unit tests, to validate functional requirements.
- Code quality, e.g., by checking if all variables are using the expected casing and if the code is respecting the indentation standards.
- Execution time and memory usage, useful for courses where computing efficiency is relevant.

With this tool, students can check with full autonomy if their work satisfies the requirements that have been defined for the assignment. This gives them the opportunity to improve their work before the deadline, as opposed to the more traditional method where students would only receive feedback after the work had been delivered and usually in the form of a grade.

[1] WAR files – Web Application Resource or Web application ARchive – are used to distribute a collection of Java classes, HTML pages and other resources that together constitute a web-application.

[2] https://Web-CAT.org/projects/Web-CAT/InstallWebCat.html.

[3] https://github.com/Web-CAT.

[4] https://sos-cer.github.io/projects/auto-grading.

[5] https://www.ulusofona.pt/en/undergraduate/computing-engineering.

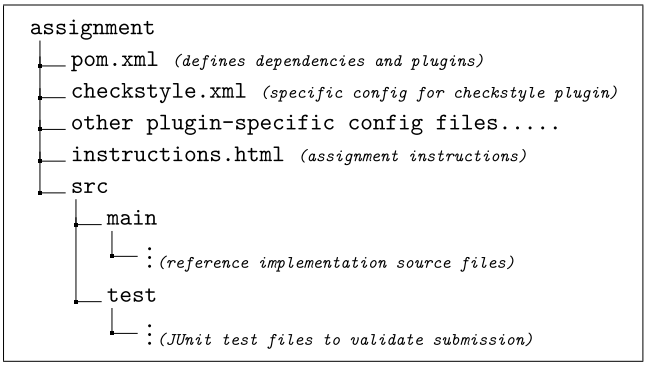


Fig. 1. Assignment folder structure.

2. Software description

2.1. Assignments and submissions

The two most important concepts in DP are assignments and submissions. Assignments are created by teachers and represent a problem (e.g., exercise, project, etc.) to be solved by the students. The assignment contains a set of instructions that the students must follow, as well as any code or configuration that is needed in order to evaluate the student's work, such as unit tests. Submissions are performed by students, according to the assignment instructions, with the primary goal of passing all the unit tests.

2.2. Workflow

Teachers create assignments as standard Maven projects, as illustrated in Fig. 1. The `pom.xml` file defines the Project Object Model, which contains information about the project, as well as any configuration details used by Maven during the build process. This includes dependencies from third-party libraries and the project's build life cycle. In its most simple form, the build life cycle fetches dependencies, compiles and packages the project. Nonetheless, depending on the plugins configured in the `pom.xml` file, the build cycle can perform dozens of other tasks, such as checking compliance with coding guidelines, unit testing, integration testing, calculating quality metrics, determining test coverage, deploying to the cloud, and more. Some plugins require additional configuration files. For example, the Checkstyle [17] plugin – which can, for example, check if all variable names are at least 3 characters long – requires the `checkstyle.xml` file. The assignment must include at least a JUnit [18] test class in the `/src/test` folder and a `Main` class with the reference implementation, that must pass all the tests. This is important since it reduces the possibility of errors in the test cases, which happen more frequently than desired [5].

Assignments are stored in a private Git repository to which DP must be connected using the standard access keys made available by the repository provider. Any changes made to the assignment are easily updated in DP through a simple refresh button.

A video demonstration of the workflow from both the teacher's and the student's point view is available at https://www.youtube.com/watch?v=65IwIBuMDlE.

2.3. Software architecture

The application is developed in Kotlin and uses the Spring Boot framework [19]. This framework handles several facets of an application such as request dispatching, authentication, form validation, HTML rendering, error handling, asynchronous execution and integration tests.

DP's architecture, presented in Fig. 2, is based on the MVC (Model View Controller) design pattern and integrated with Spring Boot. Students and teachers interact with the system via a web application. HTTP(S) requests are handled by a set of controller classes and respective methods. For example, the `AssignmentController` manages the creation and edition of assignments. Services use automatically injected Repository classes that handle all database operations and model classes that mimic the database tables (i.e., for each table there is a corresponding model class). Relevant examples of services are the `AssignmentValidator`, which checks assignments for consistency and correction (e.g., is the reference implementation passing the assignment's tests?), as well as the `GitClient` and the `ZipService`, which support the Git and Upload submission methods, respectively. The `JacocoResultsParser` handles the parsing of students test coverage results, and the `JUnitResultsParser` manages the parsing of JUnit test results for each submission. Some services interact directly with the file system, such as the `JacocoResultsParser` and `JUnitResultsParser`, which read the files produced by the `MavenInvoker` class. The latter interacts with the Maven Invoker open-source library [20], allowing programmatic execution of Maven builds. This library runs as a service, as shown in Fig. 2, and is a core component of DP's workflow, discussed in the next paragraph. As mentioned above, the model component mimics the database and represents domain concepts such as `Assignment`, `Submission` and `Author` (i.e. `Student` or `Teacher`). Data persistence is achieved through the Java Persistence API (JPA).

Fig. 3 presents the evaluation pipeline followed by DP. Maven executes in its own JVM, properly sandboxed by a custom Security Manager that acts at the Java API level (e.g., checking unauthorized file access or system calls). This sandbox has the advantage of isolating student's code from the teacher's code, preventing security problems that could arise if the former could invoke the latter [21].

Additionally, DP provides a web interface where (1) students can submit their projects by uploading a ZIP file or connecting to a Git repository, (2) students can see the corresponding build reports, as shown in Fig. 4(a), and, (3) teachers can create and configure assignments by connecting to a Git repository containing the Maven "master" project. Teachers also have access to several reports such as submissions for a given assignment and the submission history of a given group.

DP's code is unit tested and, as of July 2021, the test suite is achieving a code coverage above 75%.

2.4. Software functionalities

DP provides teachers and students with the following features:

Standard assignment format Assignments are Git repositories with a Maven project, following a standard (i.e., not specific to DP) configuration.

Individual and group submissions Besides standard individual submissions, DP supports submissions performed by groups.

Programming Languages Java and Kotlin are supported out-of-the-box. Since Maven runs on the JVM, integration of other JVM languages, such as Clojure and Scala, is trivial.

Cool-off period Teachers can define an optional waiting period between submissions in order to minimize "trial and error" approaches by the students [22].

Access-control lists DP supports (optional) per assignment access-control lists in order to ensure that only authorized students have access to specific assignments.

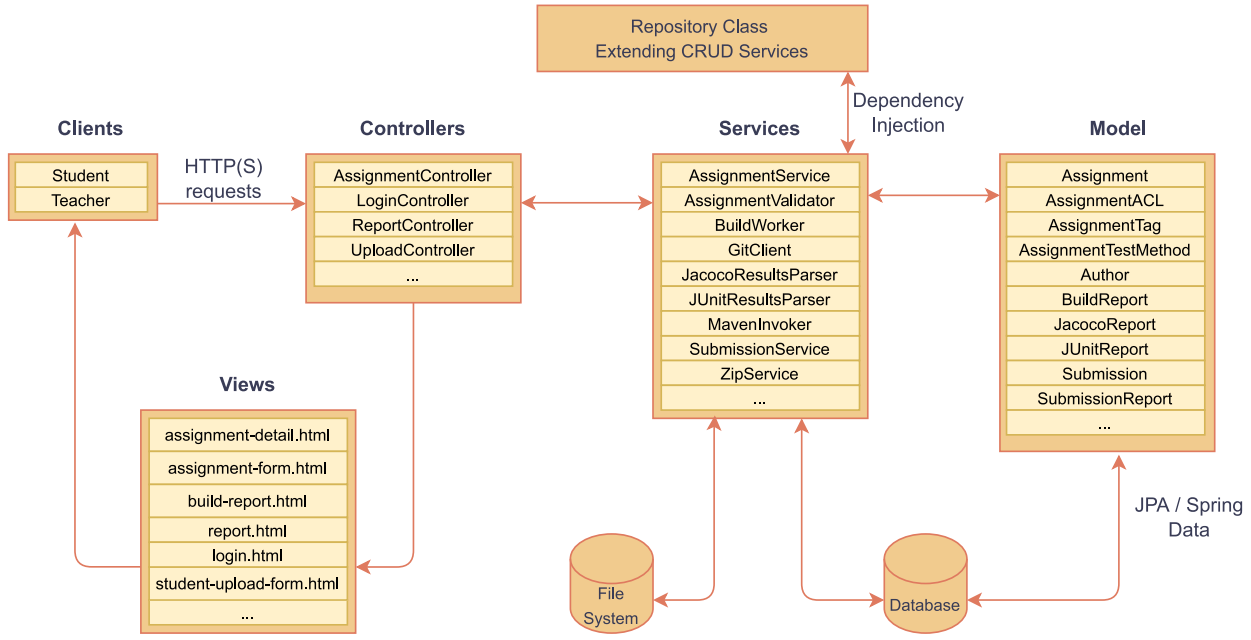


Fig. 2. DP's high level architecture, based on the MVC pattern and integrated with Spring Boot. Only the most relevant classes are presented.

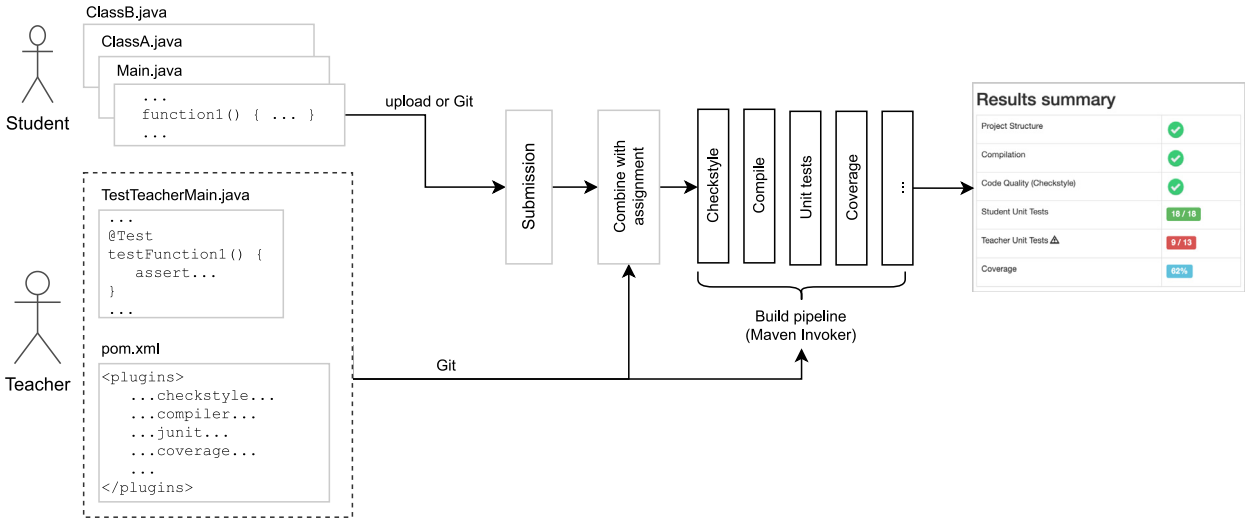


Fig. 3. DP's evaluation pipeline. The student's submitted files are combined with the assignment's files and then processed by the build pipeline. In the end, the evaluation report is generated.

Multiple submission methods DP supports Git (e.g., GitHub) and ZIP file submissions, allowing teachers to adapt the assignment submission procedure to the student's experience level.

Public tests Unit tests for evaluating submissions that give complete feedback to students. Note that only the feedback is public, i.e., the test code itself, is not public.

Private/hidden tests Unit tests for evaluating submissions that offer no feedback or partial feedback to students.

Student's own tests DP is able to run and evaluate unit tests defined by the students.

Submission timeout detection DP is able to detect and abort the evaluation process when a student's submission is taking too long to execute, which allows interrupting solutions that have infinite loops or solutions that are not optimized.

Multiple testing approaches Teachers can define assignment's tests using all strategies available in JUnit, such as unit tests, input/output matching tests, and so on. Since the tests are implemented by code, the teacher has full control on the level of feedback given to students.

Code style evaluation DP is able to check the student's code using teacher-defined style rules. Style validation can also be configured to forbid certain keywords and apply restrictions to the student's code, such as disallowing loop-related keywords in exercises that require recursive implementations.

Code coverage evaluation DP can evaluate student's own tests in terms of coverage of the respective code, thus promoting unit testing.

Execution time DP measures the amount of time a submission takes to run, thus allowing for code efficiency evaluation.

Submission signaling The capability of signaling "interesting" submissions. For example, reporting groups that fail the same tests or groups that pass the majority of tests with a relatively low number of submissions, facilitating plagiarism detection.

Leaderboard Teachers can configure a leaderboard where the students' projects and respective results are listed. DP supports three types of leaderboard, which differ in the way projects are sorted: number of public tests passed, test coverage percentage and execution time.

Test matrix DP displays a matrix/table where teachers can check which students are passing each unit test, either public or private. This is useful, for example, to detect tests that are too hard or tests that should be reviewed because they might be incorrect.

Submission download Teachers can download student's submissions in standard Maven format. The submission can then be imported with a single click into any IDE, thus allowing easy debugging of the student's failing tests.

Data exporter DP can export student's results to a CSV file,[6] simplifying the grading process and allowing for statistical analysis.

Assignment exportation Assignments and submissions can be exported, allowing them to be ported to other DP installations or to the teachers' own computers to be subject to further analysis. This is also useful for DP's developers to test their additions to the tool.

Mark submission as final Teachers can mark which of a student's submission should be exported. This gives some extra flexibility in deciding which submission will be graded, since the last submission might not be the best one.

Rebuild submission DP is able to rebuild submissions considering the latest test code. This is useful to rebuild students' submission when bugs in the assignment are found and fixed after the students' last submissions.

3. Illustrative examples

Several DP reports are displayed in Fig. 4. Fig. 4(a) presents a submission report from the student's point of view. In this example, the submission is validated by Checkstyle, all of the student's own tests are passing, but only 20 out of 24 teacher tests do so. Note that these 24 tests are the public ones. Fig. 4(b) also shows a submission report, this time from the teacher's perspective. It contains more information than the student's version, indicating how many hidden tests the submission is passing and how much code coverage the student's own tests have reached. Students and teachers also have access to the detailed JUnit report, which contains the reasons for any potential failures (Fig. 4(c)). The displayed information is fully controlled by the teachers when defining the assignment's tests, taking advantage of JUnit's API. The feedback can range from the standard JUnit output, e.g. "Expected 10 but was 20", to more textual hints and/or tips, such as "Function getMovies() returned too many elements. Beware of duplicated entries in the input file.". Listing 1 shows an example JUnit test similar to the ones we define to validate student's submissions.

Sample assignments with introductory programming exercises for both Java[7] and Kotlin[8] are available from DP's GitHub account.

4. Deployment

DP's GitHub repository contains a "Deploy to Heroku" button which builds a WAR file based on the master branch and deploys it to Heroku, a cloud provider.[9] This process results in an executable demo containing an in-memory database with fake/test data (teachers, students and submissions), thus allowing the experimentation of the system with minimal time investment. Heroku provides free accounts that can be used to perform this experimentation. The repository also contains a Docker [23] image which can be used to run the demo locally or to install a production-ready version of DP on a server.

5. Impact

In the 2018/19 edition of Algorithms and Data Structures, an anonymous student survey was conducted in order to understand the students' perception of using the tool, considering aspects such as usability, motivation and fairness. The survey was optional and was completed by 22 students after the main project had been graded. Students were asked to specify their level of agreement with the statements listed in Table 1 using a 5-point Likert scale (1=Strongly disagree, 5=Strongly agree). In general, the majority of students agree with the statements, indicating that DP has a positive impact on their learning experience. This is particularly clear in the final question, where most students strongly agree with the prospect of using Drop Project again.

DP is currently being used in four course units of the aforementioned BSc: Programming Fundamentals (1st year, 1st semester), Algorithms and Data Structures (1st year, 2nd semester), Programming Languages II (2nd year, 1st semester), and Mobile Computing (3rd year, 2nd semester). DP is a core pedagogical tool in the first three of these courses, which are heavily built on top of it. It has been successfully used to support multiple types of pedagogical activities, such as classroom exercises, homework assignments, short class assessments, projects (larger evaluations that usually span weeks or months), and project defenses (small evaluations where the students have to show mastery of their own project's code). As an example, in the 2020/21 edition of Algorithms and Data Structures, a total of 40 assignments were devised. Table 2 presents statistical information about submissions performed in the context of the courses' main assignments/projects in the last two academic years. These numbers clearly demonstrate the impact that DP had on these courses, and consequently, on our teaching and assessment practices.

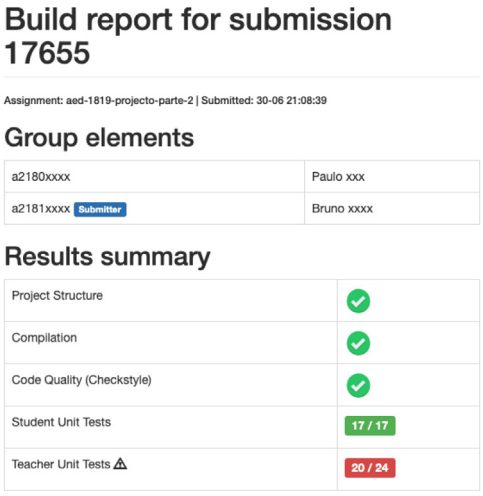
The tool's automated and immediate rich feedback helps and guides the students – as demonstrated by the survey results presented in Table 1 – while also allowing teachers to understand where students are struggling, facilitating teacher intervention and student support, while reducing the evaluation workload. This has allowed us to progressively increase the difficulty levels of the courses' projects. Also, the grading process has become more efficient and consistent between different teachers. Furthermore, every interaction between the students and DP is fully logged. Thus, DP's submission database contains data that can be mined to study the way students use the system. This data has the potential to be used in multiple research avenues, such as identification of errors in the teacher's tests, plagiarism detection, authorship attribution and/or identification of struggling students, in a similar fashion to what was done in Refs. [24–26], for example.

[6] CSV (comma-separated values) files are plain text files that contain information separated by commas and are easily imported in most spreadsheet applications.

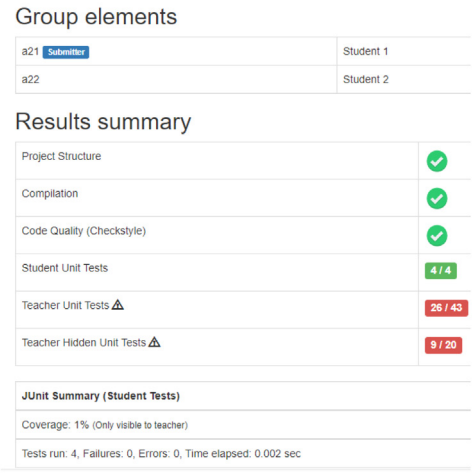
[7] https://github.com/drop-project-edu/sampleJavaAssignment.

[8] https://github.com/drop-project-edu/sampleKotlinAssignment.

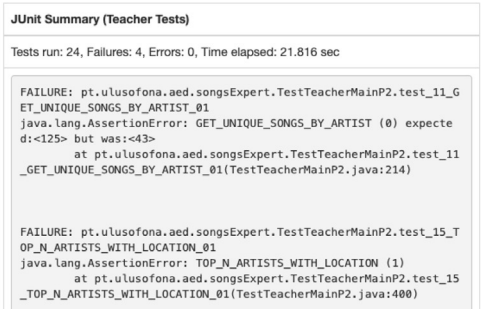
[9] https://www.heroku.com.



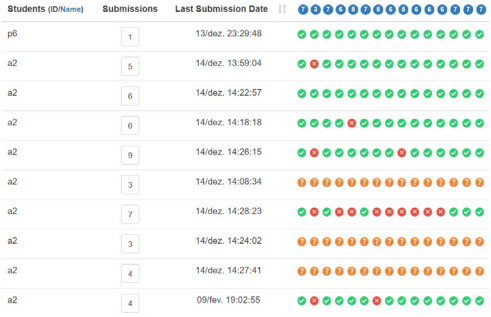
(a) A submission report from the student's perspective.



(b) A submission report from the teacher's perspective.



(c) Excerpt of JUnit report (student's perspective).



(d) Test matrix for a recent assignment (teacher's perspective).

Fig. 4. Several example reports from DP. On the left side are two reports from the student's perspective, while the right side shows two reports from the point of view of the teachers.

Listing 1: An example JUnit test for a project where students implement a board game. The messages are only displayed when the respective assertion fails.

```
@Test
public static void testMovePlayer001() {
    GameManager manager = new GameManager("game-001.txt");
    assertEquals("The function getCreatures() returned a list of incorrect size", 2, manager.getCreatures().size());
    assertFalse("Cannot move creatures to the outside of the board!", manager.movePlayer(0, 0, -1, 0));
    assertTrue("A Turtle should be able to move from 0,0 to 0,1", manager.movePlayer(0, 0, 0, 1));
}
```

Table 1
Survey results evaluating the students' experience with Drop Project using a Likert scale (1-Strongly disagree, 5-Strongly agree).

Statements evaluated	Mean response
1. I felt more confident in my programming abilities	3.36
2. I felt the grading was fair	3.64
3. I didn't need much teacher support to complete the project	3.77
4. It increased my fondness of programming	4.09
5. I am looking forward to use this tool again	4.82

6. Conclusions

In this paper we presented DP, an open-source AAT with a wide range of features that can be used to support programming courses of varying complexity and students with different backgrounds. The usage of DP has allowed us to reduce evaluation workload, improve grading consistency and augment the project's difficulty levels.

In the future we aim to increase the range of projects that can be assessed by this tool, by supporting other build systems besides Maven, thus extending the number of languages that can be used with DP.

Table 2
Assignment and submission metrics for the projects of each course designed around DP that is described in the 'Impact' section, considering the latest two academic years. The presented courses are Programming Fundamentals (PF), Algorithms and Data-Structures (ADS), and Programming Languages II (PL2). Only the submissions for the course's main project/assignment are presented.

Academic year	Metric	Course units		
		PF	ADS	PL2
2019/20	Students	115	72	71
	Submissions	4188	1901	1544
	Tests	28	13	24
	Cool-off (min.)	5	45	60
2020/21	Students	112	108	88
	Submissions	3810	2935	2705
	Tests	28	24	25
	Cool-off (min.)	0	30	30

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was funded by the Fundação para a Ciência e a Tecnologia, Portugal under Grant No. : UIDB/04111/2020 (COPELABS).

References

[1] Douce C, Livingstone D, Orwell J. Automatic test-based assessment of programming: A review. J Educ Resour Comput 2005;5(3):4–es. http://dx.doi.org/10.1145/1163405.1163409, URL http://portal.acm.org/citation.cfm?doid=1163405.1163409.

[2] Amelung M, Forbrig P, Rösner D. Towards generic and flexible web services for E-assessment. ACM SIGCSE Bull 2008;40:5.

[3] 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 calling '10. Berlin, Germany: ACM Press; 2010, p. 86–93. http://dx.doi.org/10.1145/1930464.1930480, URL http://portal.acm.org/citation.cfm?doid=1930464.1930480.

[4] Lehtonen T. Javala–addictive e-learning of the Java programming language. In: Proceedings of Kolin Kolistelut/Koli calling–Fifth annual baltic conference on computer science education. Finland: Koli; 2005, p. 41–8.

[5] 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. 2015, p. 39–44.

[6] 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: International conference on human interaction and emerging technologies. Springer; 2020, p. 42–8.

[7] Leal JP, Silva F. Mooshak: A web-based multi-site programming contest system. Softw - Pract Exp 2003;33(6):567–81. http://dx.doi.org/10.1002/spe.522.

[8] Guerreiro P, Georgouli K. Combating anonymousness in populous CS1 and CS2 courses. ACM SIGCSE Bull 2006;38(3):5.

[9] Edwards SH, Perez-Quinones MA. Web-CAT: Automatically grading programming assignments. SIGCSE Bull 2008;40(3):328. http://dx.doi.org/10.1145/1597849.1384371.

[10] Sharp C, van Assema J, Yu B, Zidane K, Malan DJ. 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. New York, NY, USA: Association for Computing Machinery; 2020, p. 487–92. http://dx.doi.org/10.1145/3341525.3387417.

[11] 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, Maryland, USA: ACM Press; 2018, p. 794–9. http://dx.doi.org/10.1145/3159450.3159595, URL http://dl.acm.org/citation.cfm?doid=3159450.3159595.

[12] Martín-Blas T, Serrano-Fernández A. The role of new technologies in the learning process: Moodle as a teaching tool in Physics. Comput Educ 2009;52(1):35–44.

[13] Lobb R, Harlow J. Coderunner: A tool for assessing computer programming skills. ACM Inroads 2016;7(1):47–51.

[14] Wünsche BC, Chen Z, Shaw L, Suselo T, Leung K-C, Dimalen D, Mark Wvd, Luxton-Reilly A, Lobb R. Automatic assessment of OpenGL computer graphics assignments. In: Proceedings of the 23rd annual ACM conference on innovation and technology in computer science education. 2018, p. 81–6.

[15] Apache Software Foundation. Apache Maven project. 2021, https://maven.apache.org/. [Online; last accessed 17 February 2021].

[16] Edwards SH, Börstler J, Cassel LN, Hall MS, Hollingsworth J. Developing a common format for sharing programming assignments. ACM SIGCSE Bull 2008;40(4):167–82.

[17] Ivanov R, et al. Checkstyle. 2021, https://checkstyle.org/. [Online; last accessed 17 February 2021].

[18] Garcia B. Mastering software testing with JUnit 5: Comprehensive guide to develop high quality Java applications. Packt Publishing Ltd; 2017.

[19] The Spring Team. Sprint boot. 2021, https://spring.io/projects/spring-boot. [Online; last accessed 17 February 2021].

[20] Scholte R, et al. Apache Maven Invoker. 2022, https://maven.apache.org/shared/maven-invoker/. [Online; last accessed 09-March-2022].

[21] Kratzke N. Smart like a fox: How clever students trick dumb automated programming assignment assessment systems. In: Lane HC, Zvacek S, Uhomoibhi J, editors. Proceedings of the 11th international conference on computer supported education, Volume 2. SciTePress; 2019, p. 15–26. http://dx.doi.org/10.5220/0007424800150026.

[22] 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. 2021, p. 1062–8.

[23] Boettiger C. An introduction to Docker for reproducible research. Oper Syst Rev 2015;49(1):71–9.

[24] Ljubovic V, Pajic E. Plagiarism detection in computer programming using feature extraction from ultra-fine-grained repositories. IEEE Access 2020;8:96505–14.

[25] Spacco J, Denny P, Richards B, Babcock D, Hovemeyer D, Moscola J, Duvall R. Analyzing student work patterns using programming exercise data. In: Proceedings of the 46th ACM technical symposium on computer science education. New York, NY, USA: Association for Computing Machinery; 2015, p. 18–23. http://dx.doi.org/10.1145/2676723.2677297.

[26] Diana N, Eagle M, Stamper J, Grover S, Bienkowski M, Basu S. An instructor dashboard for real-time analytics in interactive programming assignments. In: Proceedings of the seventh international learning analytics & knowledge conference. 2017, p. 272–9.