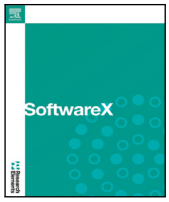
SoftwareX 24 (2023) 101578

Contents lists available at ScienceDirect

SoftwareX

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



Original software publication

Dodona: Learn to code with a virtual co-teacher that supports active learning

Charlotte Van Petegem [a,*], Rien Maertens [a], Niko Strijbol [a], Jorg Van Renterghem [a], Felix Van der Jeugt [a], Bram De Wever [b], Peter Dawyndt [a], Bart Mesuere [a]

[a] Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium
[b] Department of Educational Studies, Ghent University, Ghent, Belgium

ARTICLE INFO

Keywords:
Computer-assisted instruction
Interactive learning environments
Education

ABSTRACT

Dodona () is an intelligent tutoring system for computer programming. It provides real-time data and feedback to help students learn better and teachers teach better.

Dodona is free to use and has more than 61 thousand registered users across many educational and research institutes, including 20 thousand new users in the last year. The source code of Dodona is available on GitHub under the permissive MIT open-source license.

This paper presents Dodona and its design and look-and-feel. We highlight some of the features built into Dodona that make it possible to shorten feedback loops, and discuss an example of how these features can be used in practice. We also highlight some of the research opportunities that Dodona has opened up and present some future developments.

Code metadata

Current code version	2023.08
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00106
Code Ocean compute capsule	n/a
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	Ruby 3.1 (Ruby-on-Rails 7.0), JavaScript, TypeScript
Compilation requirements, operating environments & dependencies	Ruby 3.1, yarn 1.22, Unix-like (e.g. Ubuntu 22.04), Memcached 1.6.14, MySQL 8
If available Link to developer documentation/manual	https://docs.dodona.be
Support email for questions	dodona@ugent.be

1. Motivation and significance

Learning how to solve problems with computer programs requires practice, and programming assignments are the main way in which such practice is generated [1]. Because of its potential to provide feedback loops that are scalable and responsive enough for an active learning environment, automated source code assessment has become a driving force in programming courses. This has resulted in a proliferation of educational programming platforms [2–5]. Automated assessment was introduced into programming education in the early 1960s [6] and allows students to receive immediate and personalized feedback on each submitted solution without the need for human intervention. [7] identified the labor-intensive nature of assessing programming assignments as the main reason why students are given few such assignments when ideally they should be given many more. While almost all platforms support automated assessment of code submitted by students, contemporary platforms usually offer additional features such as gamification in the FPGE platform [8], integration of full-fledged editors in iWeb-TD [9], exercise recommendations in PLearn [10], automatic grading with JavAssess [11], assessment of test suites using test coverage measures in Web-CAT [12] and automatic hint generation in GradeIT [13].

This paper presents Dodona () as an online learning environment that recognizes the importance of active learning and just-in-time feedback in courses involving programming assignments. After presenting some of its key features for computer-assisted learning and teaching (Section 2) and giving an example on how they can be used (Section 3),

* Correspondence to: Krijgslaan 281 S9, 9000 Ghent, Belgium.
E-mail address: dodona@ugent.be (Charlotte Van Petegem).

https://doi.org/10.1016/j.softx.2023.101578
Received 14 February 2023; Received in revised form 19 October 2023; Accepted 2 November 2023
Available online 10 November 2023

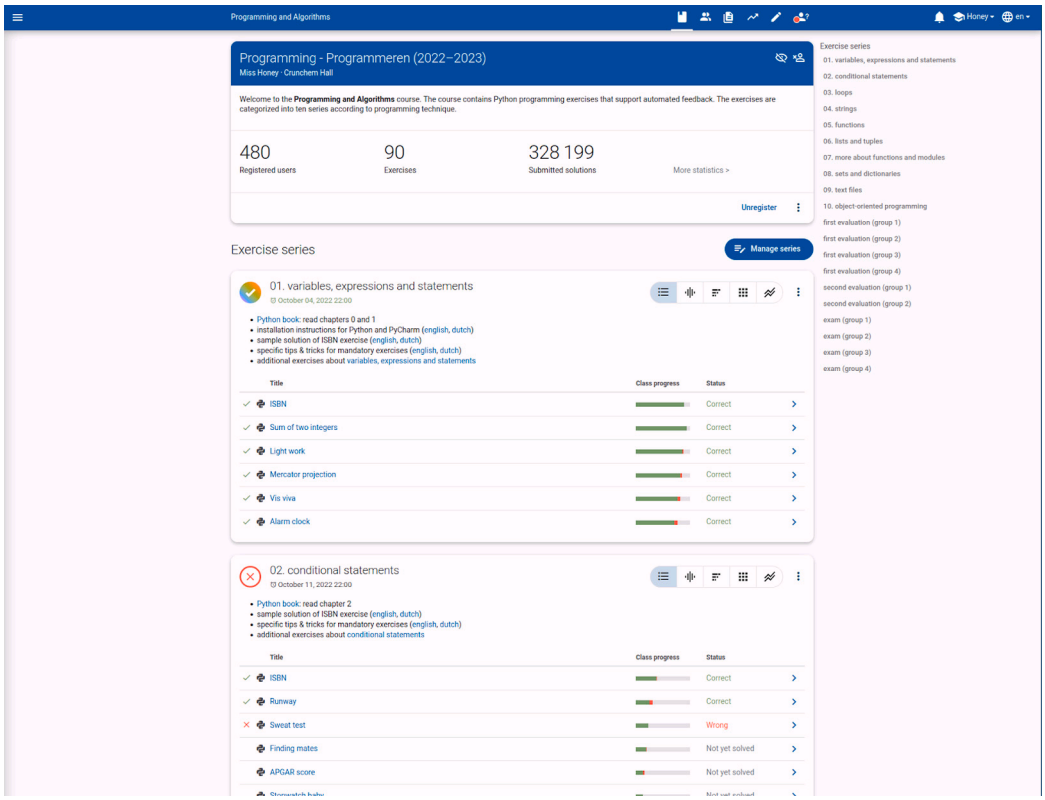


Fig. 1. Main course page (administrator view) showing some series with deadlines, reading activities and programming assignments in its learning path. The red dot on top shows that there are pending questions.

we discuss how Dodona succeeds in being a general tool for programming education that has grown beyond the institution where it was originally created, and how it has enabled EdTech research (Section 4).

2. Software description

Dodona is an intelligent tutoring system for computer programming based on a generic infrastructure for automatic assessment. It uses a distributed model for the development and publication of learning materials. Dodona lowers the barriers to wider adoption of the tool by following best practices for authentication, content management, and assessment. In addition, we adopted a holistic view on computer-assisted learning and teaching that spans all aspects of managing courses, including learning analytics and educational data mining. Each of these features is described in more detail below. A full explanation of all features can be found in Section 1 of the supplementary material.

2.1. Classroom management

In Dodona, a **course** is where teachers and instructors effectively manage a learning environment. A course can be thought of as a learning path consisting of course units called **series**, each containing a sequence of **learning activities** (Fig. 1). Among the learning activities we distinguish between **reading activities**, which can be marked as read, and **programming assignments** with support for automated assessment of submitted solutions. Teachers can make content in the course invisible/inaccessible to students. This can be used, for example, when assignments are being prepared for an assessment.

Students can **self-register** for courses to avoid unnecessary user management. However, teachers can restrict which students can register and whether registrations need to be approved before students can access the course. Teachers can also add labels to students to manage subgroups, which can play a role in learning analytics and reporting.

2.2. Student submissions

Reading activities can be marked as read only once, but there is no limit to the number of solutions students can submit to programming assignments. Series can have a deadline which does not prevent students from submitting solutions, but learning analytics and other reports/exports will usually only take into account submissions before the deadline. Submitted solutions are automatically assessed and feedback is displayed as soon as it is ready. In addition to automated assessment, student submissions can be further assessed and graded manually.

As all submissions are stored along with their metadata, we use this data to show reports and learning analytics on the course page [14]. The data can also be exported to support teachers wanting to do their own analysis on the data [15,16].

2.3. Trusted identities

To ensure that teachers can trust the identities of their students, which is particularly important for assessments, Dodona uses decentralized authentication. It does this by delegating authentication to external identity providers via SAML [17], OAuth 2 [18,19] and OpenID Connect [20].

2.4. Automated assessment

There is a wide variety of approaches to software testing (static /dynamic, black-box/white-box). In addition, there are many criteria against which students' source code can be validated (e.g. functional correctness, speed, memory usage, security, readability). To cope with this diversity, Dodona uses a generic infrastructure for automated assessment, consisting of three loosely coupled components: a container, a judge, and an assignment-specific assessment configuration.

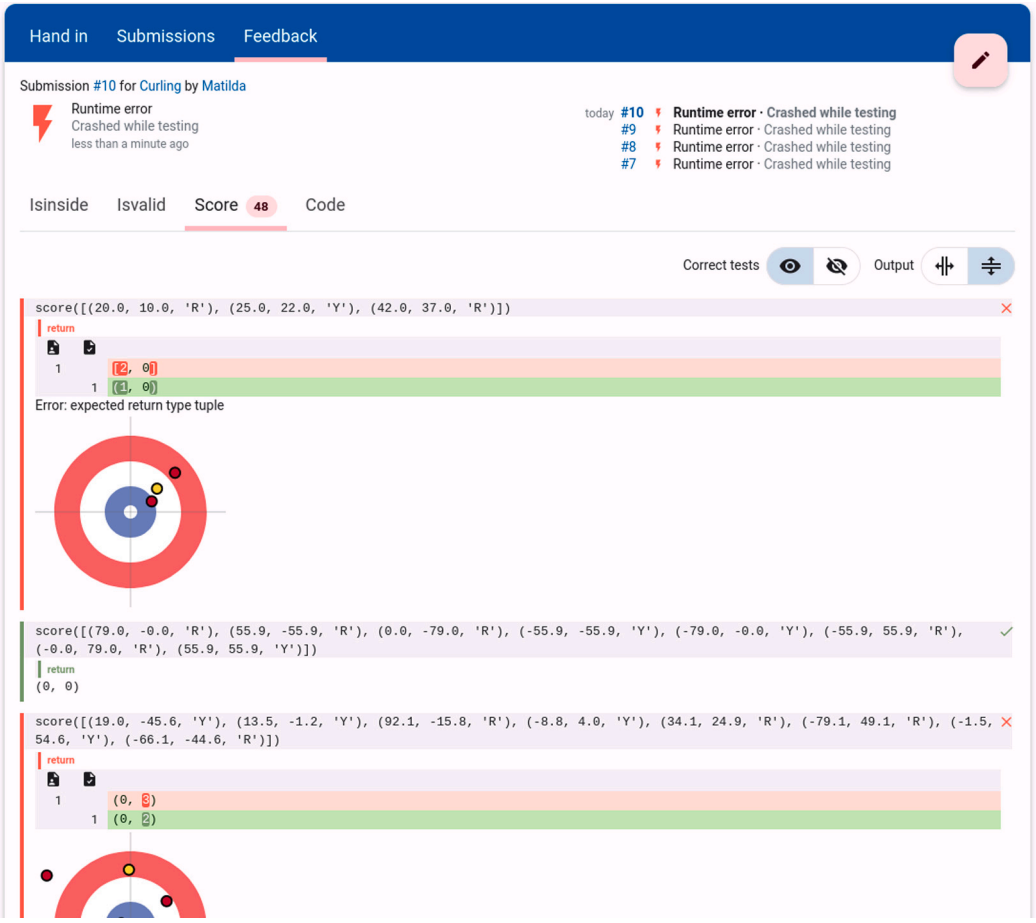


Fig. 2. Dodona rendering of feedback generated by the judge that assessed a submission of the Python programming assignment "Curling". The judge split its feedback across three tabs, one for each function that needs to be implemented for this assignment: `isinside`, `isvalid` and `score`. All tests under the `isinside` and `isvalid` tabs passed, but 48 tests under the score tab failed as can be seen immediately from the badge in the tab header. Dodona also added a fourth tab "Code" that displays the source code of the submission with annotations added during automatic and/or manual assessment.

A Docker container [21] is used for proper virtualization. It defines the runtime environment in which all data and executables are provided. Before starting an assessment, the container is extended with the submission, judge, and assessment configuration and its resources.

The actual assessment is performed by a judge [22] which can be used to assess submissions for multiple assignments. The judge uses the assignment configuration to know how to test the submission. Rather than providing a fixed set of judges, Dodona uses a minimalistic interface that allows third parties to create new judges.[1] A judge reads the assessment configuration and generates the feedback using a JSON schema.[2] What the assessment configuration should look like is defined by the judge.

Dodona takes responsibility for rendering the feedback (Fig. 2). This frees judge developers from the effort of rendering the feedback. It also gives a consistent look and feel to students solving programming assignments, even if their submissions were assessed by different judges.

2.5. Questions, answers, and manual assessment

Dodona allows students to ask teachers questions directly about their submitted code, either on a specific line or as a general question about the submission. Teachers are notified when there are pending questions (see the red dot on top of Fig. 1), which they can manage from a dashboard. Teachers can reply directly to students' annotations. Teachers can also add these annotations without there first being a question. This functionality is used as a building block for manual assessment.

Teachers can create an **evaluation** to manually grade submissions in a series. The evaluation aggregates the submissions that need to be assessed and allows teachers to navigate through them systematically. A scoring rubric can also be added per exercise. While marking, assessors can leave feedback in the same way that they annotate student code (Fig. 3). When reviewing a student's submission, assessors have direct access to the feedback that was previously generated during automated assessment. Manual feedback and grades are shared with the students at the touch of a button. Grades can also be exported.

2.6. Reliability and robustness

To ensure that the system is robust to sudden increases in workload and when serving hundreds of concurrent users, Dodona has a multi-tier service architecture that delegates different parts of the application to different servers running Ubuntu 22.04 LTS. More specifically, the web server, database (MySQL 8), caching system (Memcached 1.6.14) and Python Tutor each run on their own machine. In addition, a scalable pool of interchangeable worker servers are available to automatically assess incoming student submissions. The web server is the only public-facing part of Dodona, running a Ruby on Rails web application (Ruby 3.1, Rails 7.0) that is available on GitHub under the permissive MIT open-source license.

[1] https://docs.dodona.be/en/guides/creating-a-judge/
[2] https://github.com/dodona-edu/dodona/tree/develop/public/schemas

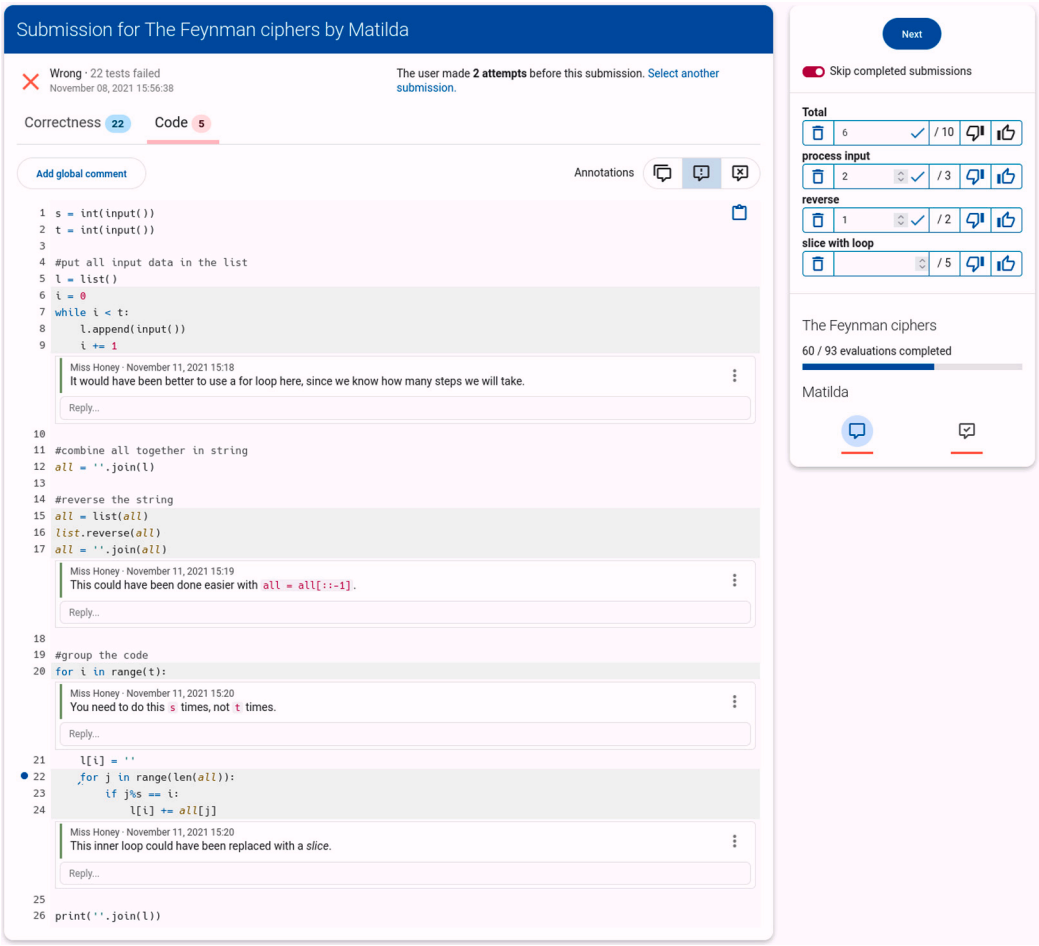


Fig. 3. Manual assessment of a submission: a teacher gave feedback on the code by adding inline annotations and is grading the submission by filling up the scoring rubric.

Dodona needs to operate in a challenging environment where students simultaneously submit untrusted code to be executed on its servers ("remote code execution by design") and expect automatically generated feedback, ideally within a few seconds. Many design decisions are therefore aimed at maintaining and improving the reliability and security of its systems.

3. Illustrative example

Dodona was originally created to support our own introductory programming course. This course has 10 series of programming assignments, divided into topics. There are also two midterms during the semester and an exam at the end of the semester. Hidden series are used for these midterms and exams. They are graded using Dodona's built-in grading functionality, which allows feedback to be given to the students with minimal overhead. Several features in Dodona have been implemented to meet the needs of this course. A full case study of how we use Dodona to run this course can be found in Section 2 of the supplementary material.

4. Use and impact

Dodona's design decisions have allowed it to spread to more than 1000 schools, colleges and universities, mainly in Flanders (Belgium) and the Netherlands. The renewed interest in embedding computational thinking in formal education has undoubtedly been an important stimulus for such a wide uptake [23]. All other educational institutions use the version of Dodona hosted at Ghent University, which is free to use for educational purposes.

Dodona currently hosts a collection of 15 thousand learning activities that are freely available to all teachers, allowing them to create their own learning paths tailored to their teaching practice. In total, 61 thousand students have submitted more than 15 million solutions to Dodona in the seven years that it has been running (Fig. 4).

A qualitative user experience study of Dodona was performed in 2018. 271 higher education students responded to a questionnaire that contained the following three questions: 1. What are the things you value while working with Dodona? 2. What are the things that bother you while working with Dodona? 3. What are your suggestions for improvements to Dodona? Students praised its user-friendliness, beautiful interface, immediate feedback with visualized differences between expected and generated output, integration of the Python Tutor, linting feedback and large number of test cases. Negative points were related to differences between the students' local execution environments and the environment in which Dodona runs the tests, and the strictness with which the tests are evaluated. Other negative feedback was mostly related to individual courses the students were taking instead of the platform itself.

4.1. Research possibilities

The large amount of educational data generated by Dodona opens up new research opportunities to better understand students' behavior, progress and knowledge. This in turn can lead to better informed decisions about courses and their pedagogy, and to identify students at risk. To perform such investigations, researchers can either export data from their courses or set up their own Dodona instance. Rich metadata is available, allowing for a broad spectrum of research opportunities.

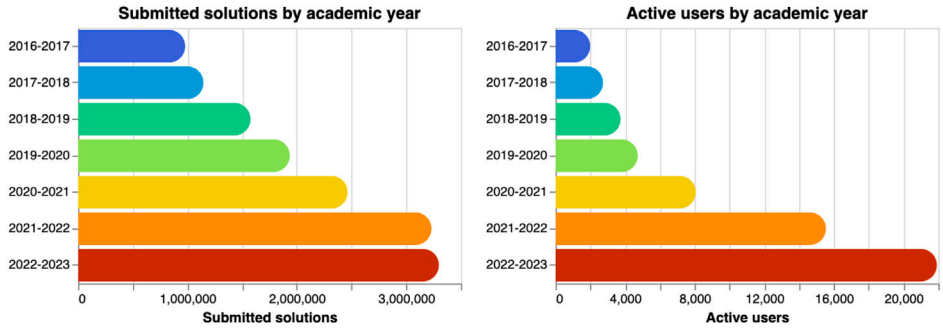


Fig. 4. Overview of the number of submitted solutions and active users by academic year. Users were active when they submitted at least one solution for a programming assignment during the academic year.

This includes pass/fail prediction [24], plagiarism detection [25], recommendations of manual feedback, exercise recommendations [10], and user knowledge modeling [26,27].

Due to the open-source nature of Dodona, the platform itself can also serve as a base for further development, such as advanced learning analytics [28], generic educational software testing frameworks [29], automatic hint generation [30], static analysis of code modifications made by students [31], and integration of external tools.

5. Conclusions

Nicol and Macfarlane-Dick's model of formative assessment and their seven principles of good practice feedback [32] fit very well with how Dodona facilitates students' self-directed learning as they practice their programming skills, and how it provides teachers with information that helps to shape their teaching. For students, active learning promises to reinforce learning by creating shorter feedback loops that help them make adjustments early on in the learning process. However, scaling up the provision of feedback throughout the entire learning process might become a real bottleneck for teachers and instructors.

Dodona therefore aims to free up valuable teacher time to maintain a collaborative and responsive dialogue with students based on high-quality and timely feedback. However, while it may be the ultimate ideal for some, current educational technology does not yet allow the entire feedback loop to be fully automated. Supporting the human aspect of learning and teaching is therefore an important focus in the design of the Dodona user experience. Students can track and fix potential errors in their code with a built-in graphical debugger, ask online questions directly about their submitted solutions with the integrated Q&A module, and monitor their own progress with learning analytics dashboards. Teachers can customize learning paths with their own learning materials and interactive assignments, share materials with their colleagues, monitor student progress (individually or in groups) using learning analytics dashboards, organize high-stakes tests and exams with automated feedback, assess students with rich feedback using a grading module with support for code reviews, and detect and prevent plagiarism with dedicated and interactive tools. Pushing the boundaries of Dodona as a virtual co-teacher that progressively becomes smarter at supporting or automating pedagogical tasks is an active area of our research.

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.

Data availability

No data was used for the research described in the article.

Acknowledgments

We are grateful for the financial support of Ghent University (Belgium) and the Flemish Government (Belgium, Voorsprongfonds) through numerous innovation in education grants. Part of this work was also supported by the Research Foundation — Flanders (FWO) for ELIXIR Belgium (I002819N). Thanks to Ghent University (Belgium) for granting us the 2018 Minerva Award for our contributions to active learning and innovation in education through the development of Dodona. Thanks to the Flemish Government (Belgium) for granting us with the 2022 Flanders Digital Award for providing each student high quality education through Dodona. Thanks to Johan Van Camp and his team at the Ghent University Data Center for hosting Dodona software services. Thanks to Hanne Elsen (UGent Data Protection Office) for assisting us with GDPR and privacy-related issues. Thanks to the computer science students and instructors who helped in developing the Dodona platform as interns, while working on their master's thesis or while running courses: Winnie De Ridder, Tibo D'hondt, Lucianos Lionakis, Felix Van der Jeugt, Mathieu Coussens, Pieter De Clercq, Timon De Backer, Ilion Beyst, Dieter Mourisse, Brecht Willems, Robbert Gurdeep Singh, Louise Deconick, Tim Ramlot, Bram Devlaminck, Freya Van Speybroeck, Toon Baeyens, Jeroen Tiebout and Anton Kindt. Thanks to all judge developers: Peter Dawyndt, Dieter Mourisse, Niels Neirynck, Felix Van der Jeugt, Charlotte Van Petegem, Bart Mesuere (Python); Peter Dawyndt, Dieter Mourisse, Bart Mesuere, Rien Maertens, Charlotte Van Petegem (JavaScript); Felix Van der Jeugt, Christophe Scholliers, Charlotte Van Petegem, Rien Maertens (Haskell); Niels Neirinck, Pieter Verschaffelt, Charlotte Van Petegem (Bash); Felix Van der Jeugt, Pieter De Clercq, Bart Mesuere, J. Steegmans (Java); Robbert Gurdeep Singh, Charlotte Van Petegem, Rien Maertens (Prolog); Dieter Mourisse (C#); Charlotte Van Petegem, Viktor Verstraelen, Gust Bogaert, Koen Plevoets, Bart Mesuere (R); Maarten Vandercammen, Elisa Gonzalez Boix (C/C++); Boris Sels, Niko Strijbol, Charlotte Van Petegem, Peter Dawyndt, Bart Mesuere (TESTed); Mathijs Saey (Scheme); Stijn De Clercq, Quinten Vervynck, Brecht Willems (HTML & CSS); Brecht Willems, Tim Ramlot, Pieter De Clercq (SQL); Thanks to Pieter De Clercq and Tobiah Lissens for developing the JetBrains IDE plugin, to Stijn De Clercq and Pieter De Clercq for developing the Visual Studio Code plugin and to Mathijs Saey at the Vrije Universiteit Brussel (Brussels, Belgium) for developing the DrRacket plugin. Thanks to all teachers and instructors who developed learning activities for Dodona and shared them on the platform as open educational resources. Thanks to all users who reported issues and provided feedback.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2023.101578.

References

[1] Gibbs G, Simpson C. Conditions under which assessment supports students' learning. Learn Teach Higher Educ 2005;(1):3–31, URL https://eprints.glos.ac.uk/3609/. Number: 1 Publisher: University of Gloucestershire.

[2] Ala-Mutka KM. A survey of automated assessment approaches for programming assignments. Comput Sci Educ 2005;15(2):83–102, Publisher: Routledge _eprint: http://dx.doi.org/10.1080/08993400500150747.

[3] Douce C, Livingstone D, Orwell J. Automatic test-based assessment of programming: A review. J Educ Resourc Comput 2005;5(3):4–es. http://dx.doi.org/10.1145/1163405.1163409.

[4] 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. New York, NY, USA: Association for Computing Machinery; 2010, p. 86–93. http://dx.doi.org/10.1145/1930464.1930480.

[5] Paiva JC, Leal JP, Figueira A. Automated assessment in computer science education: A state-of-the-art review. ACM Trans Comput Educ 2022;22(3):34:1–40. http://dx.doi.org/10.1145/3513140.

[6] Hollingsworth J. Automatic graders for programming classes. Commun ACM 1960;3(10):528–9. http://dx.doi.org/10.1145/367415.367422.

[7] Cheang B, Kurnia A, Lim A, Oon W-C. On automated grading of programming assignments in an academic institution. Comput Educ 2003;41(2):121–31. http://dx.doi.org/10.1016/S0360-1315(03)00030-7, URL https://www.sciencedirect.com/science/article/pii/S0360131503000307.

[8] Paiva JC, Queirós R, Leal JP, Swacha J, Miernik F. Managing gamified programming courses with the FGPE platform. Information 2022;13(2):45. http://dx.doi.org/10.3390/info13020045, URL https://www.mdpi.com/2078-2489/13/2/45. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[9] Fonseca I, Martins NC, Lopes F. A web-based platform and a methodology to teach programming languages in electrical engineering education – evolution and student feedback. In: 2023 32nd annual conference of the European association for education in electrical and information engineering. 2023, p. 1–3. http://dx.doi.org/10.23919/EAEEIE55804.2023.10181316, URL https://ieeexplore.ieee.org/abstract/document/10181316. ISSN: 2472-7687.

[10] Vasyliuk A, Lytvyn TBV. Design and implementation of a Ukrainian-Language educational platform for learning programming languages. 1613. 2023, p. 0073, Proceedings http://ceur-ws.orgISSN. URL https://ceur-ws.org/Vol-3426/paper32.pdf.

[11] Insa D, Silva J. Automatic assessment of Java code. Comput Lang, Syst Struct 2018;53:59–72. http://dx.doi.org/10.1016/j.cl.2018.01.004, URL https://www.sciencedirect.com/science/article/pii/S1477842417301045.

[12] Edwards SH, Perez-Quinones MA. Web-CAT: Automatically grading programming assignments. In: Proceedings of the 13th annual conference on innovation and technology in computer science education. New York, NY, USA: Association for Computing Machinery; 2008, p. 328. http://dx.doi.org/10.1145/1384271.1384371, URL https://dl.acm.org/doi/10.1145/1384271.1384371.

[13] Parihar S, Dadachanji Z, Singh PK, Das R, Karkare A, Bhattacharya A. Automatic grading and feedback using program repair for introductory programming courses. In: Proceedings of the 2017 ACM conference on innovation and technology in computer science education. New York, NY, USA: Association for Computing Machinery; 2017, p. 92–7. http://dx.doi.org/10.1145/3059009.3059026, URL https://dl.acm.org/doi/10.1145/3059009.3059026.

[14] Ferguson R. Learning analytics: Drivers, developments and challenges. Int J Technol Enhanced Learn 2012;4(5/6):304–17, URL http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijtel. Number: 5/6.

[15] Baker RSJd, Yacef K. The state of educational data mining in 2009: A review and future visions. J Educ Data Min 2009;1(1):3–17. http://dx.doi.org/10.5281/zenodo.3554657, URL https://jedm.educationaldatamining.org. Number: 1.

[16] Romero C, Ventura S. Educational data mining: A review of the state of the art. IEEE Trans Syst, Man, Cybern, C (Appl Rev) 2010;40(6):601–18. http://dx.doi.org/10.1109/TSMCC.2010.2053532, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).

[17] Farrell S, Reid I, Orchard D, Sankar K, Moses T, Edwards EN, et al. Assertions and protocol for the OASIS security assertion markup language (SAML). 2002, Organization for the Advancement of Structured Information Standards (OASIS) Standard (November 2002), http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf.

[18] Hardt D. The OAuth 2.0 authorization framework. Request for Comments. RFC 6749, Internet Engineering Task Force; 2012, http://dx.doi.org/10.17487/RFC6749, URL https://datatracker.ietf.org/doc/rfc6749. Num Pages: 76.

[19] Leiba B. OAuth web authorization protocol. IEEE Internet Comput 2012;16(1):74–7. http://dx.doi.org/10.1109/MIC.2012.11, Conference Name: IEEE Internet Computing.

[20] Sakimura N, Bradley J, Jones M, De Medeiros B, Mortimore C. Openid connect core 1.0. 2014, p. S3, The OpenID Foundation.

[21] Peveler M, Maicus E, Cutler B. Comparing jailed sandboxes vs containers within an autograding system. In: Proceedings of the 50th ACM technical symposium on computer science education. New York, NY, USA: Association for Computing Machinery; 2019, p. 139–45. http://dx.doi.org/10.1145/3287324.3287507.

[22] Wasik S, Antczak M, Badura J, Laskowski A, Sternal T. A survey on online judge systems and their applications. ACM Comput Surv 2018;51(1):3:1–34. http://dx.doi.org/10.1145/3143560.

[23] Wing JM. Computational thinking. Commun ACM 2006;49(3):33–5, Publisher: ACM New York, NY, USA.

[24] Van Petegem C, Deconinck L, Mourisse D, Maertens R, Strijbol N, Dhoedt B, et al. Pass/fail prediction in programming courses. J Educ Comput Res 2022;07356331221085595. http://dx.doi.org/10.1177/07356331221085595, Publisher: SAGE Publications Inc.

[25] Maertens R, Van Petegem C, Strijbol N, Baeyens T, Jacobs AC, Dawyndt P, et al. Dolos: Language-agnostic plagiarism detection in source code. J Comput Assist Learn 2022. http://dx.doi.org/10.1111/jcal.12662, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/jcal.12662. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jcal.12662.

[26] Blikstein P, Worsley M, Piech C, Sahami M, Cooper S, Koller D. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. J Learn Sci 2014;23(4):561–99, Publisher: Routledge _eprint: http://dx.doi.org/10.1080/10508406.2014.954750.

[27] Costa EB, Fonseca B, Santana MA, de Araújo FF, Rego J. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. Comput Hum Behav 2017;73:247–56. http://dx.doi.org/10.1016/j.chb.2017.01.047, URL https://www.sciencedirect.com/science/article/pii/S0747563217300596.

[28] Chen H-M, Nguyen B-A, Yan Y-X, Dow C-R. Analysis of learning behavior in an automated programming assessment environment: A code quality perspective. IEEE Access 2020;8:167341–54. http://dx.doi.org/10.1109/ACCESS.2020.3024102, URL https://ieeexplore.ieee.org/document/9195825. Conference Name: IEEE Access.

[29] Strijbol N, Van Petegem C, Maertens R, Sels B, Scholliers C, Dawyndt P, et al. TESTed—An educational testing framework with language-agnostic test suites for programming exercises. SoftwareX 2023;22:101404. http://dx.doi.org/10.1016/j.softx.2023.101404, URL https://www.sciencedirect.com/science/article/pii/S2352711023001000.

[30] Chow S, Yacef K, Koprinska I, Curran J. Automated data-driven hints for computer programming students. In: Adjunct publication of the 25th conference on user modeling, adaptation and personalization. New York, NY, USA: Association for Computing Machinery; 2017, p. 5–10. http://dx.doi.org/10.1145/3099023.3099065, URL https://dl.acm.org/doi/10.1145/3099023.3099065.

[31] Hamer S, Quesada-López C, Jenkins M. Students projects' source code changes impact on software quality through static analysis. In: Paiva ACR, Cavalli AR, Ventura Martins P, Pérez-Castillo R, editors. Quality of information and communications technology. Communications in computer and information science, Cham: Springer International Publishing; 2021, p. 553–64. http://dx.doi.org/10.1007/978-3-030-85347-1_39.

[32] Nicol DJ, Macfarlane-Dick D. Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. Stud High Educ 2006;31(2):199–218, Publisher: Routledge _eprint: http://dx.doi.org/10.1080/03075070600572090.