Assessment Automation of Complex Student Programming Assignments

Subjects: Computer Science, Information Systems | Education & Educational Research

Contributor: Matija Novak , Dragutin Kermek

Grading student programming assignments is not an easy task. This task is even more challenging when talking about complex programming assignments at university graduate level. By complex assignments, researchers mean assignments where students have to program a complete application from scratch. For example, building a complete web application with a client and server side, whereby the application uses multiple threads that gather data from some external service (like the REST service, IoT sensors, etc.), processes these data and store them in some storage (e.g., a database), implements a custom protocol over a socket or something similar, implements their own REST/SOAP/GraphQL service, then sends or receives JMS/MQTT/WebSocket messages, etc. Such assignments give students an inside view of building real Internet applications. On the other hand, assignments like these take a long time to be tested and graded manually, e.g., up to 1 h per student. To speed up the assessment process, there are different automation possibilities that can check for the correctness of some application parts without endangering the grading quality.

automation grading assessment programming education student

1. Introduction

Assessment is an important part of a teacher’s job that must be done correctly and on time. However, grading student assignments, especially those of a complex nature, presents a formidable challenge for educators. This is even more noticeable when the number of students increases to 50 students or more and teachers want to publish grading results within a reasonable time (i.e., not later than two weeks). Researchers focused on the assessment of student programming assignments, which can be divided into basic (introductory) assignments usually completed by freshmen in introductory programming courses and advanced (complex) assignments completed by more advanced students in the later years of their information technology (IT) study program. The assessment of advanced programming assignments at university undergraduate and graduate levels are focused on.

Advanced assignments require the students to architect entire applications from the ground up. To be more precise, this means to program complete Internet/web applications with client and server components. These applications can include the following: implementations of multi-thread logic, building custom socket-based protocols, reading and storing data within designated repositories, facilitating data collection from diverse external sources, implementing new web services, bidirectional messaging, deployment on different servers, creating and running containers, etc.

Such complex assignments teach the students the process of developing real-world web applications, thereby providing invaluable insights into the application development life cycle. Grading such assignments is a substantial time investment, which is required for manual assessment, for the teacher. It is not unusual that grading such assignments manually takes up to an hour per student submission. In response to this challenge, various possibilities of automation have emerged to speed up the assessment process while keeping the integrity of grading standards. The implementation of this process draws upon methods such as unit testing, bash scripting, using specific software like Apache JMeter, and other such approaches.

2. Assessment Automation of Complex Student Programming Assignments

The automation of programming assignment assessments is not a new concept, as shown in [1]. With the increasing number of massive open online courses (MOOCs), the research into assessment automation has become even more interesting. The research has not been limited to functional testing [2]; it has also focused on providing automated personalized feedback, as in [3]. If readers are interested in feedback automation, a good start would be [4]'s survey.

Researchers have developed many pre-built tools to automate the assessment of programming assignments. For reference, read the survey of [5][6] for an overview of the tools developed between 2006–2010. Another more recent survey [2] was published in 2016 and includes a nice comparison of 30 tools in a detailed feature table. Beyond the tools mentioned in such surveys, there are other articles where a new tool was developed. The most recent systematic review on the topic "Automated Grading and Feedback Tools for Programming Education" can be found in [7].

For example, one study that dealt with the assessment automation of C++ programs with different levels of complexity using static code analysis was presented in [8]. An assessment source-code library was, at the moment of writing the paper, available for free (https://ucase.uca.es/cac/, accessed on 20 September 2023). Another interesting paper was [9], where the authors built a tool called the Flexible Dynamic Analyzer (FDA), which uses semantic analysis techniques to assess results. Similarly, in [10], a tool called AutoGrader was built that relies on program semantics. AutoGrader automatically decides the correctness of student programming assignments according to a reference implementation. Another tool, called DGRADER [11], uses a complex multi file program analysis that can be an advantage to handling complex programming and scaled projects that require more than one file for the program.

Although many tools are mentioned in articles, many of them are not available to the public. Note that finding the tools is not a problem. For example, on GitHub one can find many tools. The problem is that there are similar names for various tools, and it is not always clear which is the one the user is searching for. For example, with AutoGrader, there is one from coursera (https://github.com/coursera/coursera_autograder, accessed on 20 September 2023) and there is one from University of Michigan (https://github.com/eecs-autograder, accessed on

20 September 2023), but it is not clear if the AutoGrader from [10] is a yet another different tool with the same name.

With so many tools available one just has to choose one to use, but there is a problem with pre-built assessment automation tools. As nicely stated in [12] the problem is that: “many of the present assessment tools are developed for a local use and only for a certain type of assignments. Hence, they are often not available for a wider use and would be difficult to adapt to another university, anyway”.

Another issue with tools, which is shown in [2][5], is that they are built for specific programming languages. Although “some of the systems are language independent. Especially if the assessment is based on output comparison” [5], what is most noticeable is that these tools can be divided into two categories, according to [5]: first, automatic assessment systems for programming competitions and second, automatic assessment systems for (introductory) programming education. Here, researchers want to put the emphasis on the word “introductory”. Even though many tools and papers exist [13][14][15] on how to automate the assessment of programming assignments, their focus has been on simple introductory programming assignments.

The researchers focused on more complex (advanced) programming assignments that are usually present in courses in later years of one study program. This leaves us with very few options; basically, tools like AutoGrader or DGRADER would be the available options in this case. One might think that using an existing tool is easier than doing the work manually, but there are some issues to think about before deciding.

First, these tools have a certain logic that needs to be studied and understood. Second, often the tools require a teacher to define tests to be used in grading or build a reference implementation that will be used to compare the solution against and decide if the solution is correct. Building reference implementation is a special issue. When it comes to complex assignments, implementing the whole solution can take up to several days of work. Third, the tool is written in a certain programming language, which the teacher might not be familiar with, so learning the language takes time. Next, these tools have limits, and it might be that they are unable to grade everything the teacher needs the tool for, or the tool might accomplish the task in a different way than the teacher expects or wants. Modifying the tool to fit the teachers’ needs might be difficult or even impossible in existing tools. Considering all these issues, sometimes using existing pre-built tools is more difficult and unnecessary. Especially with complex assignments where things can be done in different ways, it might be easier to build custom assessment scripts.

References

1. Pieterse, V. Automated Assessment of Programming Assignments. In Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research, Heerlen, The Netherlands, 4–5 April 2013; pp. 45–56.

2. Souza, D.M.; Felizardo, K.R.; Barbosa, E.F. A Systematic Literature Review of Assessment Tools for Programming Assignments. In Proceedings of the 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), Dallas, TX, USA, 6–8 April 2016; pp. 147–156.
3. Marin, V.J.; Pereira, T.; Sridharan, S.; Rivero, C.R. Automated Personalized Feedback in Introductory Java Programming MOOCs. In Proceedings of the 2017 IEEE 33rd International Conference on Data Engineering (ICDE), San Diego, CA, USA, 19–22 April 2017; pp. 1259–1270.
4. Keuning, H.; Jeuring, J.; Heeren, B. Towards a Systematic Review of Automated Feedback Generation for Programming Exercises. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, New York, NY, USA, 11–13 July 2016; pp. 41–46.
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, New York, NY, USA, 28–31 October 2010; pp. 86–93.
6. Caiza, J.; Del Alamo, J. Programming assignments automatic grading: Review of tools and implementations. In Proceedings of the 7th International Technology, Education and Development Conference, Valencia, Spain, 4–5 March 2013; pp. 5691–5700.
7. Messer, M.; Brown, N.C.C.; Kölling, M.; Shi, M. Automated Grading and Feedback Tools for Programming Education: A Systematic Review. ACM Trans. Comput. Educ. 2023.
8. Delgado-Pérez, P.; Medina-Bulo, I. Customizable and scalable automated assessment of C/C++ programming assignments. Comput. Appl. Eng. Educ. 2020, 28, 1449–1466.
9. Fonte, D.; da Cruz, D.; Gançarski, A.L.; Henriques, P.R. A Flexible Dynamic System for Automatic Grading of Programming Exercises. In Proceedings of the 2nd Symposium on Languages, Applications and Technologies, Porto, Portugal, 20–21 June 2013; Leal, J.P., Rocha, R., Simões, A., Eds.; OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik: Dagstuhl, Germany, 2013; Volume 29, pp. 129–144.
10. Liu, X.; Wang, S.; Wang, P.; Wu, D. Automatic Grading of Programming Assignments: An Approach Based on Formal Semantics. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), Montreal, QC, Canada, 25–31 May 2019; pp. 126–137.
11. Wang, T.; Santoso, D.B.; Wang, K.; Su, X.; Lv, Z. Automatic Grading for Complex Multifile Programs. Complex 2020, 1–15.
12. Ala-Mutka, K.M. A Survey of Automated Assessment Approaches for Programming Assignments. Comput. Sci. Educ. 2005, 15, 83–102.

13. Singh, R.; Gulwani, S.; Solar-Lezama, A. Automated Feedback Generation for Introductory Programming Assignments. SIGPLAN Not. 2013, 48, 15–26.
14. Staubitz, T.; Klement, H.; Renz, J.; Teusner, R.; Meinel, C. Towards practical programming exercises and automated assessment in Massive Open Online Courses. In Proceedings of the 2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), Zuhai, China, 10–12 December 2015; pp. 23–30.
15. Parihar, S.; Dadachanji, Z.; Singh, P.K.; 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, Bologna, Italy, 3–5 July 2017; pp. 92–97.

Retrieved from https://encyclopedia.pub/entry/history/show/121897