Automated grading assignments in programming – advantages, problems and effects on learning

1st Tatjana Stojanović
Department of Software Engineering
University of Belgrade
Serbia
tatjana.stojanovic@fon.bg.ac.rs

2nd Saša D. Lazarević
Department of Software Engineering
University of Belgrade
Serbia
sasa.lazarevic@fon.bg.ac.rs

***Abstract*— E-education is having a great growth in the last few years. It is obvious that the need for e-learning software is growing. In the process of learning, grading assignments is one of the most important activities. Grading, programming languages and environments are not standard across different educational facilities, that's why many tools for grading are made especially for one or a set of courses. While there are several tools that are made for universal use, often they don't support all features that are needed for different courses. In this paper, reasons for automatization of grading activity will be presented, but also problems and considerations which must be resolved while developing such a system. Also, a model for grading complex programming assignments will be presented, which should provide the necessary flexibility for most of the courses.**

Keywords*— *e-education, e-learning, automated grading, programming assignments

I. Introduction

E-learning is one of the fastest-growing industries. The global e-learning market was projected at USD 144 Billion in 2019 and is estimated to reach USD 374.3 Billion by 2026 [1]. Demand for online courses is growing as many schools and universities had to adjust their courses to be online, since the start of the pandemic. Numerous online courses and e-learning platforms have been used for gaining knowledge not only for beginners but also for developers learning new technologies and expanding their knowledge. Tools for automatization of processes in learning are needed.

One of the most important activities of learning is the evaluation of knowledge. This encompasses defining tests, their assessments, grading and providing feedback. The idea of automatization of some of these activities has existed since the 60s. Since then many tools have been implemented and widely used – tools for creating forms, questionnaires, collaboration, online lectures, grading assignments etc.

For grading system used for grading assignments in programing two approaches are common: dynamic and static analysis of the submitted code.

Dynamic analysis approach involves code execution and comparing output of the tested program with expected output through many test cases. Test data can be manually provided or automaticaly generated. While executing code correctness and efficiency can be tested.[8]

Static analysis involves analyzing code without the need to execute it. With static analysis it is possible to grade programming style analysis, detect syntax or semantic errors, analyze software metric and structural similarity.[7] While grading an assignment with syntax errors and being able to provide a more detailed feedback, these tools are usually suitable for small programs. Also, since there are many possible solutions for a program, grader should provide them all to the systems so that assignments can be fairly graded. [8]

In this paper, a dynamic analysis tool has been presented. This tool enables a grader to create assignments combining multiple functions and determining as many as needed differents tests for each assignment and function. Problems which must be resolved while implementing this kind of system will be presented, along with advantages and disadvantages we have observed.

II. Reasons and advantages

The most important reasons for automatization of grading are [5]:

1. *Time*: grading implemented programs is time-consuming task and represent big workload on graders since each program must be tested with many tests' inputs.
2. *Error-prone*: usually, while grading an assignment teacher is, instead of testing inputs, analyzing code line by line, which can lead to more errors.
3. *Subjectivity*: while grading code inspecting code line by line, teachers usually have a model solution and can be biased when students have different solutions.
4. *Plagiarism*: software can be used to compare assignments and detect plagiarism, while when several people are grading assignments, plagiarism can remain undetected.

These problems can be overcome using tools for automated grading. Some tools are created for universal use but often they don't provide all the needed features or suitable for every type of grading assignment. That's why there are many implementations of these systems. [2]

One system for grading programing assignments should be flexible in term of defining a test, evaluating partially correct answers, generating feedback for students, and being easy to use and adapt to the specificity of a course.

Grading metrics for a course are not standard. Every institution and even every grader has its own model for grading. When grading programing assignments, these metrics are usuallly tested:[2]

1. Execution – compilation and execution,
2. Functional testing – functionality,
3. Non-functional testing
    a. Specific requirements

b. Maintainbillity – design, style, complexity

c. Efficiency – use of physical resources, execution times, processes number, file or code size.

Most of the grading tools are using dynamic analyzing test-based, meaning that usually execution and functionality of the program is being tested. Of course, for evaluating other non-functional characteristics of the program, some other student activities must be used. Also, there are some automated tools are being developed for that purpose.

Usability of test-based grading depends on topics covered by course. Teachers must be aware of possible pitfalls of using only automated grading. Although test-based tools cannot grade a programs' maintainability, their can improve learning quality. Advantages and disadvantages observed using this kind of automated grading will be discussed later.

III. Problems and considerations

Implementing a system for automated grading can be a complex task. Implementation involves predicting possible student errors, possible attempts to trick the software or submitting code, while being able to provide a complete report for grades and detailed feedback for students.

Here are identified problems that must be resolved when designing the system for grading programming assignments using dynamic analysis:

1. Syntax – beginners can often have problems in syntax while programming. When this error should occur these assignments cannot be tested, but feedback about this error can be provided.
2. Trying to trick a software – in a lack of knowledge or solution students may try to implement a solution that is true for only a subset of inputs or for given example. This can be resolved by using a large set of test inputs. Also, sometimes students are requested to implement a function that is available in the standard library. Use of those functions must be prevented.
3. Endless loops – often while learning, students may make an endless loop. This kind of error can cause a problem in a grading program. The time or number of loop iteration must be limited. Also any blocking functions, like reading from streams, need to be ended or input must be provided.
4. Runtime errors – while testing a program runtime errors must be expected. This kind of error must be expected and graded. The grading of an assignment or a test that created such an error should not impact the system. It is very important to deal with fatal errors, which can occur. It must be secured that data is not corrupted and logging of program events.
5. Memory leaks – in an environment that has a garbage collector this is not an issue, but in an environment where a student must free memory when necessary, this also must be put into consideration. When testing these systems, memory leaks, should not have a big impact on the system, but in implementation mocking and determining if there is a memory leak should be provided.
6. Potentially malicious code – program must be tested in a safe environment because a student can intentionally or unintentionally submit a malicious code.
7. Feedback – Providing useful feedback is necessary. Students need to be able to know what they did wrong. This, also, can lower number of complaints. [6] While static analyzers are able to give more detailed feedback about syntax, semantic errors and even hints on how to fix them, dynamic analyzers give informations about the passed and failed tests. Students will need to discover a reason for test failure themselves.
8. Defining a test – when using automated grading, when defining assignments they must be clear, precise and not ambiguous. Test implementation must be correct. If assignment is not clear and precise, then graders must create tests for every understanding of an assignement, in order to grade fairly.
9. Working with input and output streams – as mentioned, reading from input streams may block a function. Although, problems of blocking functions must be resolved, another issue appears. Formats of inputs and outputs may be a problem. Grader must explicitly determine and give examples for input functions. If order of inputs or format is changed, function will not provide desired solution. Same is true for output, testing printed output values in text file or standard output streams can be challenging, testing possible deviations from the desired format, and also having in mind the order of the printed values.

Listed problems are some of problems which must be taken in consideration. These are the most important considerations to have in mind when developing a grading system for programming assignments, while depending of the course needs, new ones may appear. Most of these considerations need to be resolved for any kind of grading and test running, others can be resolved in test itself. Proposition for some problems are given.

IV. Model

In this section, a model for designing this system will be described. It is shown in Figure 1.

Usually, a exam test is created with an idea of connecting several topics and assignments which are part of the exam test. An exam test is consisted of implementing different functions and being able to use those functions and combining them to resolve an assignment. Sometimes function implemented for one assignment can be reused in resolving another one. This is done to let students recognize where they can reuse a function and to connect different topics. Hence, a separation between a function and an assignment has been made. This also enables a program to grade a partially implemented assignment. Also, the presented system is flexible and can be customized to suit different assignment types.

When defining a test, a student must be given a function prototype based on which it will implement necessary functions.

Student data can be obtained in several ways. We used a named source code file for defining students personal data.

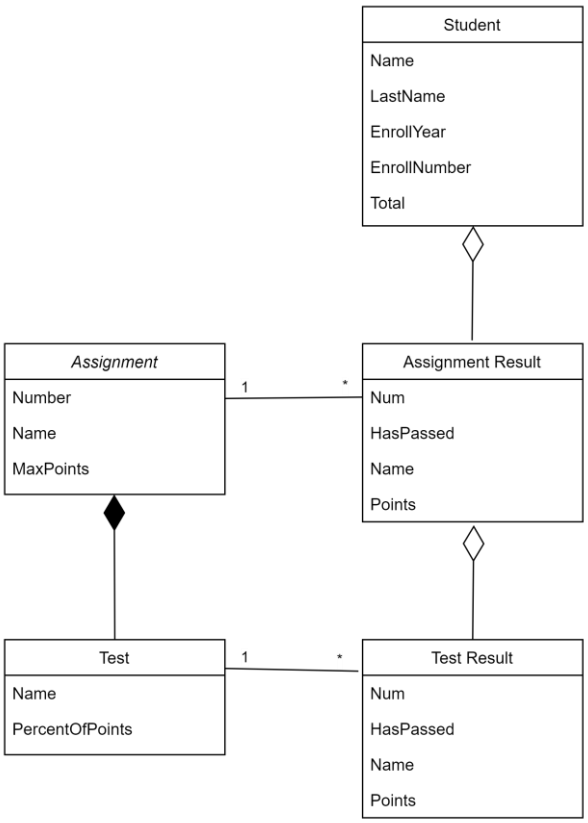

Fig. 1 Model of automated grading system

V. GRADING WORKFLOW

Software system for grading assignments in C is implemented using mentioned model. Using this model, it's easy to define functions suitable for a course need.

First step in grading process is creating assignments. When assignments are created, a code generator for setting up tests is used. For each assignment it is required to provide a total number of points, a description. Next, for each assignment it is needed to declare at least one test which should be run. For each test a percentage of total points should be declared along with a description. This description provides a feedback for students, so it needs to explain what is actually being tested. Grader should have in mind that any kind of error in one test will result in zero points.

```
TYPES
typedef int MATRIX [10][10];
typedef struct node NODE;
typedef struct node* PNODE;
struct node{
int info;
PNODE next;
};
ETYPES
FNS
int columnMaximum(MATRIX m, int rowCount, int column-
Index);
double arithmeticMean(int* givenArray, int n);
void insertAtBeginning(int* givenArray, int* n, int
value);
void removeFromIndex(int* givenArray, int* n, int giv-
enIndex);
int isBinary(char* text);
int isDecimal(char* text);
int isCorrectTime(char* time);
struct time duration(struct time start, struct time
end);
void insertAtEnd(PNODE* head, int value);
int mostFrequent(PNODE head);
EFNS
START task1 | 20 | Task1
columnMaximum | 0.25 | Is returning the column maximum
arithmeticMean | 0.25 | Is returning the arithmetic
mean
insertAtBeginning | 0.25 | Is inserting an element at
the beginning of the array
removeFromIndex | 0.25 | Is removing an element from
given index
END
```

The first part (block from TYPES to ENDTYPES) lets user define types that will be needed in order to implement functions. Second part from FNS to ENDFNS defines all functions which students should implement and use in other functions.

Grader only needs to implement tests. It is possible to use any kind of library needed for the course. When all tests are implemented, program needs to be compiled and grading exams can be run. Program is designed in a way that a single or a batch of programs can be tested.

After student has taken exam their source code can be graded. One limitation is that student can have only one file with source code. Each exam (their source code) will be compiled in *dll* library to be tested.

Their program is compiled against a header which declares all export functions and an additional header which helps the system to monitor their work. Result of grading is a *csv* file, but the format can be easily changed.

```
Name, Last name, Year, Number, Total, Total for
assignment 1, Description for test 1.1, Descrip-
tion for test 1.2, Total for assignment 2, De-
scription for test 2.1, Description for test 2.2
Aleksadra,Vukovic,2018,2872,5,5.000000,0,1,0,0,
Ana,Mirkovic,2019,2185,5,5.000000,0,1,0,0,
Marija,Bisevac,2018,3327,20,20.000000,1,1,1,1,
Anja,Nikolic,2018,845,10,10.000000,0,1,0,1,
Petar,Andjelic,2019,2543,5,5.000000,0,0,0,1,
Djordje,Kojanic,2020,2822,15,15.000000,1,1,0,1,
Jovan,Radosevic,2019,818,10,10.000000,0,1,1,0,
Andjela,Vukasinovic,2019,3808,0,0.000000,0,0,0,0,
……
```

Report contains information about scores for students, and feedback about every assignment and test for each student with their descriptions. Each assignment for each student is run as a separate process in order to execute tests safely.

VI. ADVANTAGES AND DISADVANTAGES

Using an automated grading system always require for teachers and graders to adjust the way that they create assignments. Here is a list of the most important advantages and disadvantages of using such system.

Advantages are:

1. Fast – in couple of seconds hundereds of programs can be graded, which is usually done by multiple graders and it would last for hours. Lowering workload for teachers involving grading activites, can give them time to devote time to other activites which improves whole learning process, such as

teaching preparation, more assignment prepartion and other.

2. Objective – all students are graded the same way. Beside time, objectivity is one of the most important advantages.
3. Students are guided to a desired solution – declaring types and prototypes enables a grader to determine in which way a function should be implementeted. In that way graders have better control over topics which they want to cover and grade.
4. More assignments can be given – it is shown that frequently giving assignments during semester makes students engaged in lectures and helps them to learn more easily. Because of high workload for teachers and graders, students are usually not given enough assignments. When using automatized grading one can easily give as much as assignments as possible.
5. More testing – given the fact that even a small mistake can disable a program to be able to grade them. This encourages students to test their solutions thoroughly before submitting it.

Disadvantages are:

1. Code style cannot be graded – with test-based grading solution only behavior of a function is tested. For grading code style, one must grade it manually or use a software for grading code style.
2. One error fails the test – The slightest error can result in a not-working function. With test-based software we cannot partially grade those functions.
3. Less creative solutions – with giving types and prototypes student don't have to create their own solutions for these problems, and determine types which should be used.
4. More time must be spent creating assignments and tests – because assignments must be precise and clear it takes more time to double check meaning of each assignment and function prototypes. If a mistake occurs, it can be hard to fix later.

Using a test-based grading system for grading programming assignments can improve quality of grading process. Using automatization for grading assignment in programming will decrease a workload for graders, so they can dedicate their time to more creative and meaningful tasks. Also, analysis of provided feedback can also be useful for a learning process, for example, determining which topics need to be covered with more exercises and lectures.

Feedback generated by test-based system can only provide information about passing the tests. Students will know, based on provided description, where did their function fail. But they will not know where they made a mistake.

When using automated grading it is important to acknowledge the downsides. For example, a teacher should have in mind that creativity of students is not stimulated, because need for problem solving skills is somewhat reduced by giving types and prototypes to students. Also, quality of code cannot be tested. Those skills are important for students and programmers. Teachers should encourage students to adopt those skills, by using different assignment and grading technique. Of course, it is important to make reasonable assignments and tests, in order to achieve fair grading.

VII. Conclusion

As the online learning business is growing, the interest in programming education is also rising. While the interests are rising quickly, the teaching staff has been facing a large number of assignments for grading. Luckily, this activity can be automated. In this paper, problems which must be considered when designing a system for grading programming assignments were discussed.

Designing software for automated grading of progamming assignments must allow quick and correct behaviour in a safe environment, without a risk of failure or corrupting data. It is important to enable grading of partially correct solutions while providing meaningful feedback on error. A proposition for model which is general for most test-based solutions has been given, providing needed flexibility for determining a grade. This means that the flexibility of software can be adjusted to different types of exams is needed.

While determining what system or how to implement a grading system, not only implementation but also psychological and pedagogical consequences on the learning process must be considered. A teacher has to make an exam that is convenient for automated grading in chosen or implemented software, hence more time is needed for defining exams. Creating tests also must provide enough testing inputs for the grading system to work properly and must be written carefully.

In an example which have been considered in this paper, test-based grading system proved to be an useful tool for grading large group of students. Although it has benefits like possiblity of giving a large number of assignments, there is still need for other activites for aquiring other important programming skills.

Future work for this system involves web assesment of students program and also including static analysis mainly for improving feedback.

References

[1] E-learning Market By Provider (Content Provider and Service Provider), Deployment Mode (Cloud and On-premise), by Course (Primary and Secondary Education, Higher Education, Online Certification and Professional Course, Test Preparation), by Enterprise (SMEs and Large Enterprises) and By Region Global Industry Outlook, Market Size, Business Intelligence, Consumer Preferences, Statistical Surveys, Comprehensive Analysis, Historical Developments, Current Trends, and Forecasts, 2020–2026 (url: https://www.fnfresearch.com/e-learning-market)

[2] J. C. Caiza, J. M. Del Alamo, Programming assignments automatic grading: review of tools and implementations (url: https://www.researchgate.net/publication/262042207_Programming_Assignments_Automatic_Grading_Review_of_Tools_and_Implementations)

[3] M. Vujošević-Janičić, M. Nikolić, D. Tošić, V. Kuncak, Software verification and graph similarity for automated evaluation of students' assignments, Information and Software Technology, Volume 55, Issue 6, 2013, Pages 1004-1016, ISSN 0950-5849, DOI:https://doi.org/10.1016/j.infsof.2012.12.005. (url: https://www.sciencedirect.com/science/article/pii/S0950584912002406)

[4] Ihantola, P., Ahoniemi, T., Karavirta, V., & Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. Proceedings of the 10th Koli Calling Interna-

tional Conference on Computing Education Research - Koli Calling *'10.* DOI:10.1145/1930464.1930480

[5] B. Cheang, A. Kurnia, A. Lim, Wee-Chong Oon, On automated grading of programming assignments in an academic institution, Computers & Education, Volume 41, Issue 2, 2003, Pages 121-131, ISSN 0360-1315, DOI: https://doi.org/10.1016/S0360-1315(03)00030-7. (url: https://www.sciencedirect.com/science/article/pii/S0360131503000307)

[6] H. H. Lee. 2021. Effectiveness of Real-time Feedback and Instructive Hints in Graduate CS Courses via Automated Grading System. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21). Association for Computing Machinery, New York, NY, USA, 101–107. DOI: https://doi.org/10.1145/3408877.3432463

[7] K. M Ala-Mutka (2005) A Survey of Automated Assessment Approaches for Programming Assignments, Computer Science Education, 15:2, 83-102, DOI: 10.1080/08993400500150747

[8] Rahman, Khirulnizam & Nordin, Md Jan. (2007). A Review on the Static Analysis Approach in the Automated Pro-gramming Assessment Systems.