Automatic Marking System

An investigation into the automatic marking of Java code for students using a web application.

Robert Guest – 1607625

R.Guest@wlv.ac.uk

Contents Page

Abstract

Introduction

*Academic Question*

*Aims*

*Objectives*

*The need for automatic marking*

*Main introduction*

Literature Review

Analysis of findings

Development of the Artefact

Testing

Conclusion

Critical Evaluation

Future Recommendations

References

Bibliography

Introduction

Academic Question

What are the main considerations for using an automatic marking system for assessing students code?

Aims

1. To evaluate current uses of automatic marking systems.
2. To build a web application for students to upload their code for assessment and receive a score.
3. To conduct further research to see how the artefact can be used as a tool to teach programming.

Objectives

Aim 1:

1. Use the internet to find relevant academic papers on existing systems.
2. Assess the pros and cons of the existing systems to see how they could be improved.
3. Use these points for improvement as requirements for the development of the artefact and put them into practice.

Aim 2:

1. Find the most appropriate tools for the development. Including IDE’s and frameworks.
2. Setup a server for the web application to be placed upon, while having a separate application for compiling the code.
3. Set up a development environment and version control for the code, so that any older version can be used if need be.

Aim 3:

1. Conduct research into the idea of static analysis of code using academic papers.
2. Evaluate static analysis to discuss if this is the most appropriate method for the artefact.
3. Research automatic marking from an educational perspective to see how such an artefact could be used as a learning tool.

The need for automatic marking

When it comes to automatic assessment, this has proven to be very useful for schools, especially at a university level. This is because of the sheer number of students that are taking part in computer science courses and will all be sitting programming assessments. For a tutor to go through each students code and give feedback, this would take up a lot of time. This would be appropriate for the marking of written work, but for code, this is something that can be better dealt with. It is clear through research, that many schools, especially at a university level, have already been putting this idea into practice.

This project looks to address the many different ways an automatic marking system for code could be implemented, to look for the best option. Many schools have software in place for students to run during their assessments, where they will fill in answers boxes (for their code) and receive instant feedback once they submit the code. Having such systems in place reduces stress on tutors and saves them a huge amount of time.

An example of an existing system is ‘CodeAssessor’ implemented by the University of Tennessee. This is an example of students being able to work through computer science problems from basic to advanced while getting instant feedback. ‘Hence students receive immediate feedback as they progress through problems, and the feedback also ensures that they do not get stuck for prolonged periods of time’ Brad Vander Zanden, Michael W. Berry (2013).

When an automatic marking system is in place, the benefits serve both the tutors and the students. This is because tutors can use software to create their own programming tasks to be automatically assessed. While students can learn how to code this way at their own pace, and not have to wait for feedback. Also, in the case of this artefact, if the student is given a score, this will give the student the motivation to try and get their score to be perfect, as they attempt tasks over and over to try and achieve the maximum. This is the main reason behind this system being useful as a learning tool, as students will aspire to constantly take their skills to the next level.

From a tutors perspective, the system can help them track a students progress. This can be done in many ways, for example; the system can log the number of attempts made at a task and the score of each attempt. This would give the tutor an idea of any areas the student might be struggling with.

‘The evaluation of a student program can be simplified by compiling and executing the student's code in order to see what the program actually does’ Kenneth M. Dawson-Howe (1995). By compiling the students program automatically, this essentially takes away the task the tutor would have to do themselves. This can save them even more time as the tutor would not have to worry about whether or not they have issues with compiling it themselves for whatever reason.

Universities often have modules on computer science courses that require students to complete programming assessments. This means that the use of such a system can be apart of an entire course as well as just certain modules. Each school will likely have their own take on an automatic marking system for a certain language etc. This project looks into the idea of having a system that can be adapted to ay programming language and that can be there to be used by any educational establishment.

‘By using automatic marking systems of our own design we can check student work and obtain the results immediately. It reduces our labor, enables us to grasp individual students’ learning states, and allows us to tailor our instruction to each student's needs’ Hidekatsu Koike et al. (2006).

An existing idea of automatic marking is the use of static analysis. This is something that is in place in some schools and is an idea considered for this project. An example of such an existing system is ‘AutoGradeMe’. ‘The system provides continuous feedback during the development process. However, having multiple checkers might be a disadvantage because you cannot rely on them having equal reliability’ Zahid Ullah et al. (2018). Static analysis is a more complex idea which can provide more than just information about errors and warnings. Fundamentally, it involves assessing a program without having to execute it, which is what this artefact aims to achieve.

Main introduction

The idea of automatic code assessment is something that has been looked into many times before, but never really implemented or used widely enough for it to be considered a norm. It is clear that research is being conducted in this area to see how systems can be put in place to give students accurate feedback on their code in the best way possible.

In computer science, it is easy for a student to enter code into an IDE and it will show them what errors/warnings they have and even fix them for the student. This is an effective learning technique, but when it comes to learning how to code, it is important to be confident in your own ability and keep striving for improvement. This is why many people, especially university tutors/lecturers in computer science, encourage not using an IDE and simply reading compiler information to see what’s wrong with the code. This means the student can start of with small programs and learn how to debug properly, then even go onto develop their own set of coding standards as they develop their skills.

The idea of this project is setting up tasks for students that are learning how to code in Java, from basic to more intermediate, as part of an overall assessment. The students would be writing their own code without the use of an IDE, so they can see how their own code can be improved. As well as seeing where they went wrong, the student will be given an overall score, with the max score being perfect code with no errors or warnings.

References

1. Brad Vander Zanden, Michael W. Berry. (2013). ‘Improving Automatic Code Assessment’. Volume 29. Issue 2. Pages 162-168.
2. Kenneth M. Dawson-Howe. (1995). ‘Automatic submission and administration of programming assignments’. Volume 27. Issue 4. Pages 51-53.
3. Hidekatsu Koike et al. (2006). ‘Using an automatic marking system for programming courses’. n/a. n/a. Pages 173-178.
4. Zahid Ullah et al. (2018). ‘The effect of automatic assessment on novice programming: Strengths and limitations of existing systems’. Volume 26. Issue 6. Pages 2382-2341.