

An Enhanced Generic Automated Marking Environment: GAME-2

Roozbeh Matloobi, Michael Blumenstein, Steve Green.

School of Information and Communication Technology, Griffith University

Key Words: *Tele-learning, Automated Marking, Fuzzy Logic*

Abstract:

In this paper we describe an extension of the Generic Automated Marking Environment (GAME-2) and provide an analysis of its performance in assessing student programming projects. GAME-2 has been designed to automatically assess programming exercises written in C, C++ and JAVA languages based on a number of factors including meaningful comments, the structure of functions and the detection and correction of compile-time errors. The assessment is marked based on these metrics using heuristic and fuzzy rules. In this research, GAME-2 has been tested on a number of student programming exercises and assignments. The system has attained encouraging result as compared to a human marker.

1. Introduction

The development of automatic assessment systems has become important with application in a number of areas over the last few decades. In particular, interest has grown in the development of tools for automatic assessment of computing programs. These tools can be used in the field of education, proving to be useful for educators in assisting students' based on their capabilities and providing useful feedback according to their skills [1].

The final aim of automatic marking systems is to perform the grading of assignments as close as possible to a human marker. Although some limited systems currently exist, there is still much research that must be undertaken to develop an accurate marker that can be used for small and large computer programs written in a variety of languages. However the difficulties in determining a clear marking methodology and also understanding the comments contained in a program remains a major challenge for researchers [2].

Fuzzy logic enables Automated Marking to assess programming assignments, with the sort of decisions which a human would make. This is exactly what fuzzy logic can do. What is more impressive is that fuzzy logic offers a way of processing these decisions so that a final result is still correct [3].

In the next section, we describe some existing systems that have been designed for automatic assessment. The description of GAME-2 and its functionality is reported in section 3. Section 4 provides some experiences in evaluating GAME-2 and in section 5, results and experiments are shown. Finally, in section 6, conclusion and future work is presented.

2. Current Automated Marking Systems

There are numbers of automated systems, which have been developed [2]-[9]. Reek [4] proposed a program called TRY, for helping the instructors of a graduate course. The system took the student's PASCAL source file, compiled it, and ran it on a predefined set of input files. Their system did not take into account style or design issues. Jackson and Usher [2] planned a system called ASSYST that tests the accuracy of an ADA program by analyzing its output and comparing it to a correct specification using in-built tools of the UNIX operation system. PRAM [5] is a Prolog Automatic Marking system, while the main objective is to mark students' Prolog programs. A final mark covering style, complexity and correctness of programs is presented to the student by PRAM along with some comments on the code. Saikkonen [6] proposed a system called Scheme-robo for assessing programming exercises written in the functional programming language Scheme. The system assesses individual procedures instead of complete programs. Ghosh *et al.* [7] developed a fully automated system for marking and plagiarism detection of programs written in the C language. The system compiles and executes each student program and performs a simple comparison between the program's output and a model output file. Automatic control educational software (ACES) [8] presents an interactive software platform. ACES automatically grades assessment items and produces a mark in the range according to the grade of correctness of a student's answer, instead of giving full credit for correct answers and zero credit for wrong answers using a fuzzy grading technique. Finally, GAME [9] has been designed to automatically assess programming assignments written in a variety of languages based on the structure of the source code and the correctness of the program's output. The system is able to mark programs written in Java, C++ and the C language.

2.1 Problems of Previous Systems

One of the main limitations of most automated marking systems is that they can only mark one type of programming language. GAME was designed with the intention that it could mark a wide variety of programming languages, by looking at what was common to all programming assessments. The last version of GAME was able to assess C, C++ and JAVA. However, there were still some deficiencies in the marking methodology, which affected its accuracy, including the problem of dealing with errors in programs.

ACES grades the answers of students' exercises and produces a mark based on the fuzzy grading principles and overall grade is produced. If the answer of student is correct, he receives full credit. Any other answers than is wrong, it gives a partial credit base on the closeness of the correct answer [8]. ACES provides fuzzy sets with triangular membership functions and grading principles applied [10]. Although ACES demonstrates

good results in marking HTML code, and some programs written in Visual C++, the ACES platform is unable to mark programs written in Java and other object-oriented (O-O) programming languages and thus is not generic.

3. System Design of GAME-2

GAME-2 was based on the previous version for marking Java, C, and C++ assignments (GAME) [9] and was developed in SDK 1.5. It was designed to overcome the limitations of the C-Marker system, and other existing systems. A summary of GAME-2 is presented in Figure 1. The final aim of automatic marking is to be able to mark assignments as close as possible to a human.

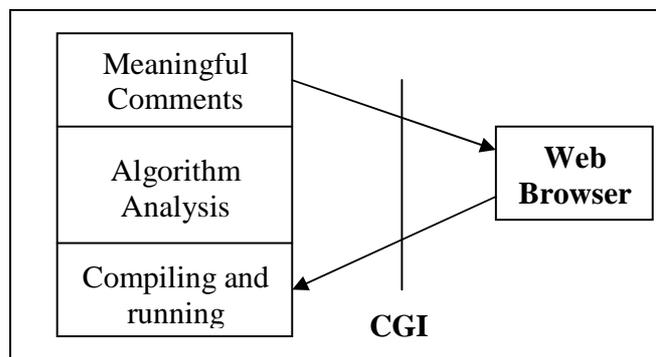


Figure 1 Design of GAME-2

3.1 GAME-2 Functionality

From previous results were obtained by GAME system [9], it was evident that the next version of GAME needs to address the following points: 1) The analysis of meaningful comments, 2) The ability to accurately examine the structure of students' source code and 3) Amending of simple compile errors. In an earlier version, GAME pointed out the all the tree requirements mentioned above. In following sections, a detailed description of GAME-2's functionality is presented including a discussion of the components addressing the above deficiencies.

3.2 Description of the Fuzzy Grading

Computer logic is exact, deterministic, relates to finite states and numbering systems. Computer logic marks distinct boundaries between any states. Whereas, fuzzy logic enables a computer to make decisions, which are more in line with the sort of decisions made by humans. Computer logic is exact, deterministic, relates to finite states and numbering systems. Computer logic marks distinct boundaries between any states. In this research, a fuzzy grading system is applied and is composed of a single fuzzy unit. The inputs to the Fuzzy unit are the comments and function/Method structure in source code. The meaning of comments and function/method algorithms are two measurements to detect limitations in a program. Depending on the stage of the marking process, a fuzzy decision can be made to find an appropriate mark [10].

3.3 Meaningful Comments

A large aspect of writing maintainable code is to provide meaningful commenting. Source code headers, function pre and post conditions, and in line comments provide valuable documentation into the understand the source code. A human marker considers the quality of comment as part of their marking criteria. The question that may be asked is, which criteria need to be considered to identify useful comments? GAME-2 processes a comment in two ways, Meaningful and Artificial comments. An Artificial comment is identified as such when more than one quarter of the comment's words is a member of "A", which is considered as code that has been commented out and Fuzzy logic assigns zero marks for this type of comments.

IF (more than one fourth of words present in a comment is a member of A) **THEN** (assign a zero marks) (1)

$A = \{ ; \{ \} [] = := == != \text{"if"} \text{"else"} \text{"import"} \text{"while"} \text{"do"} \text{"switch"} \text{"for"} \text{"white space"} \}$ (2)

Meaningful comment words are comprised of alphabetic words (a..z, A..Z). There are two kinds of Meaningful comments: Block and Inline Comments. A block comment is usually located prior to the header of the function/method and inline comments are located inside of the function/method body.

Each Meaningful comment is marked according to simple mathematical calculation. The value for block comments is considered as 10% of total marking (5% of whole marking is assigned for inline comments) [11]. Each function/Method should have a block comment as a header to explain the function operation. So the number of block comments might be equal to function/Method number. As a result, the value for each block comment is [10% / number of functions]. Processing meaningful comments is applied by the following metrics. At present the GAME-2 system performs two operations with respect to meaningful comments: 1) comment words are analysed in terms of the number of nouns and preposition article present and 2) an evaluation of the number of comment words that are conjunctions. The result is evaluated using simple fuzzy rules. A particular advantage of using fuzzy rules for this operation is that it will be possible to extend this component with relative ease in future versions of the system by adding additional rules.

MeaningfulProportion=
(NumberOfNounArticles/NOW),(NumberOfConjunctionWords/NOW) (3)

The number of word in each comment (NOW) was calculated by counting the number of nouns, prepositions and conjunction articles. The "NumberOfNounArticle" was obtained by determining the number of noun and preposition articles. Finally, "NumberOfConjunctionWord" is considered by the number of conjunction articles present. A human marker was consulted to determine a reasonable fuzzy grading schema that would assess a suitable commenting percentage as required in a source code file. The most important component for human marker was the number of noun and preposition words used in the comment and the conjunction words were the other important factor to

make the meaningful comments. Fuzzy rules were set based on these tests, one third of the mark is allocated by analyzing the number of words, half of the marks were allocated by the number of words composed of nouns and preposition articles and one sixth was allocated by the conjunction words. The exact shape of the fuzzy membership function employed (e.g., bell-shaped) and the corresponding span of the fuzzy membership function are obtained through experimentation on different types of students assignments. Then, GAME-2 grades the comments and produces a partial mark instead of giving full credit for correct answers and zero credit for wrong answers, a fuzzy grading system was used as detailed in Appendix A.

IF (each word in a comment is not a member of set A and each character is not alphabetic) **THEN** (mark are allocated based on appendix A). (4)

3.4 Algorithm Analysis

Good algorithm design [9] is another important area that students need to learn in order to produce high-quality software and for making it accessible to succeeding programmers. To examine the quality of an algorithm (specifically when a program dose not compile), GAME-2 currently looks at the "main" aspects present in the student's source file present in a given assignment. These points are program complexity which includes the number of iteration statements, selection statements as well as the number of data structure (i.e. arrays), assignment statements, inline comments and other important aspects of algorithm structure. So, the accuracy of algorithm is based on:

AlgorithmAccuracy = (NumOfIterations, NumOfConditions, NumOfAssignments, NumOfInlineComments, NumOfArrays). (5)

The fuzzy *AlgorithmAccuracy* is analysed by calculating the ratio of the number of items obtained, to the number of items from a model algorithm set by the instructor. To assign *AlgorithmAccuracy* marks, the result obtained from the human marker were analysed to create realistic fuzzy rules that would be instructive to mark the function/method algorithms. For example, the algorithm of a search function is compared to the model of an instructor's algorithm. Consequently, if the number of iteration and condition statements is two, GAME-2 gives full mark for this part if there is a match with the model solution. Otherwise, a partial mark is allocated by the system. Therefore, GAME-2 gives a partial mark (2.5 out of 3), if a search function has 2 iteration statements and 1 condition statement, as is shown in Figure 2. Moreover, the number of assignment statements and arrays is 4 providing a full mark for this part. For other situations, depending on the number of arrays and assignments, partial marks are allocated. However, the mark is full (1 out of 1) in the current case.

3.5 Assessing non-compiling Assignment written in different Languages

One of the main negative aspects of the previous version of the GAME system [9] was that it could not analyse programming projects containing compiling errors. So, one of the new features of the GAME-2 system was to enable marking of programming

assignments containing compile errors and the correction of the limited errors rather than providing the student with a mark of “zero”. The GAME-2 system is currently able to mark programming projects with compile errors and can give partial marks for the source files. This was achieved by representing all student source files as a "Compile object". The Compile object employs a simple method such as to identify missing semicolons (“;”) and parentheses in programming assignments. After recognising the errors in a program, GAME-2 is able to amend the limited errors, which have been detected (currently less than 4) and save the compiled program in a new file for further process.

3.5 The GAME-2 GUI

The GAME-2 GUI provides a simple interface to enable the user to mark programming assessment. In order to operate the system, the user is first required to select a student assignment in the root directory containing all student folders and their programming assessment.

Once the student assignment has been entered, the user may then press the “Mark” button. When the student's program has been marked, a summary of the students' source code details is displayed as a graphical document within the system's main window. The summary includes a function/method analysis and a percentage indicating meaningful comments (See Figure 2).

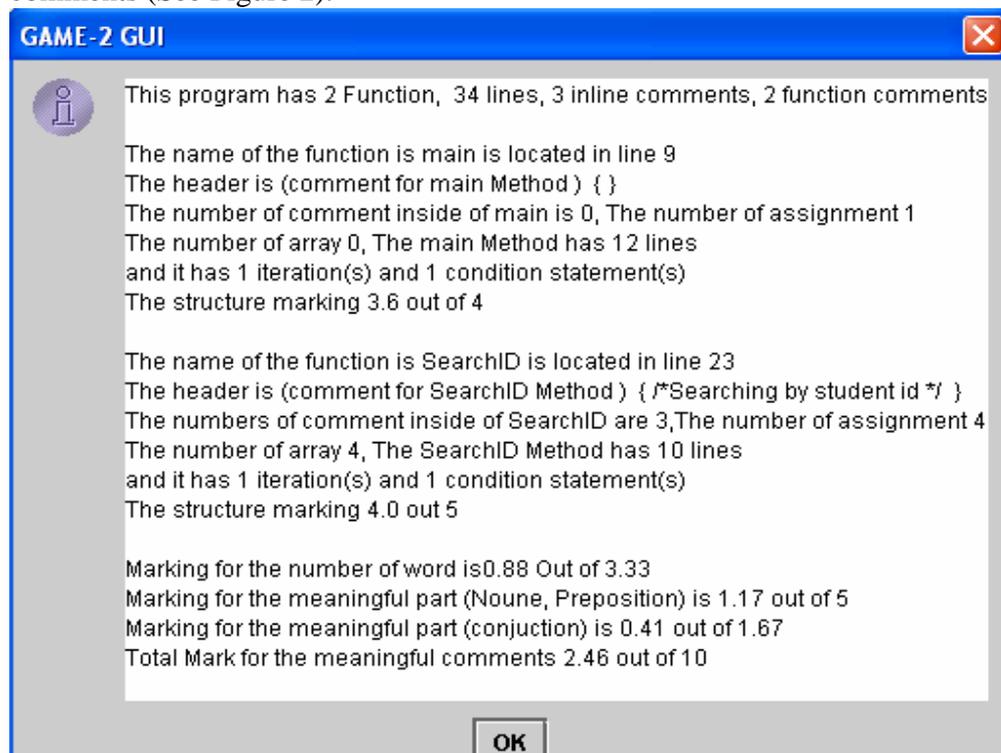


Figure 2 GAME-2 GUI

The program output displays a total mark for meaningful comments explained in sub-section 3.3 and function/method analysis described in sub-section 3.4 based on fuzzy rules. When the marking component is completed, the user may then click on the “Error

Detection” button. If there are some limited and simple errors, the “Amending source file” button will correct the errors and save the amended source code in a separate file.

4. Experiments

A number of experiments were carried out on real-world data to examine the capability of GAME-2. Three different types of programming assessments were marked and the results produced by GAME-2 were compared to a human marker and the previous version of GAME. The three different types of programming assessments were written in Java, C and C++. All three assessments were marked based on the meaning of comments, source code analysis and correction of compile errors.

4.1 Object Oriented (O-O) Programming Evaluation

The first type of assessment item was set in O-O courses at Griffith involving a simple programming structure. We have conducted the experiments with 12 Java files and 13 C++ student files. The assessment type was obtained from a first year programming course. This assessment involved only a small amount of coding using single source files. In performing source code analyzing, the GAME-2 performed reasonably well, with eighty-four percent (84%) agreement with the human marker. However, the original GAME could not analyse the algorithm in the source code and hence could not be used for comparison. In the meaningful comment component of the marking criteria, the GAME-2 system performance compared well to the human marker with (76%) similarity. On the other hand, the previous version of GAME simply identified a comment or non-comment and could not provide result in a meaningful way.

4.2 C Programming Assignments

The second major experiment devised to test GAME-2 involved assignments from a first year C programming course. Twenty five student assessments of IT Master students were selected for marking by the GAME-2. The assignments involved not only arrays but also linked list structures. The results obtained were similar to that of the O-O assignments in both meaningful comment and algorithm analysis. In the case of meaningful comments, the human marker agreed with GAME-2 in (78%) of the cases and for algorithm analysis in (88%) of cases.

4.3 Compiling and Amending Errors

All the student assignments including the O-O and C courses assessment were examined by GAME-2 for correctness in terms of correcting limited compilation errors. The amending component was to modify the missing semicolons and parentheses. The results obtained from GAME-2 show that the system can detect and add the missing elements in a separate file without any errors. However, the previous version of GAME could not correct these compilation errors.

5. Discussion

5.1 Object Oriented Programming

GAME-2 disagreed with the human marker in four out of the twenty five (16%) assessment items it was presented with in terms of algorithm analysis. The main reason for the disagreements is that GAME-2 is only able to assess limited types of functions in student assignments. However, there is no limited ability with a human marker. Hence, there are some functions, which are not assessed by GAME-2. So, the human marker gives partial or full marks based on the algorithms present, but GAME-2 gives a mark of zero.

The largest discrepancy for comments was obtained for the marking of parameter explanations. Four out of twenty-five assessments showed that the human marker disagreed with in terms of explanation of parameters in comments. The main reason (64%) is that the GAME-2 system could not identify that the comments were describing the function parameters. Therefore, GAME-2 provides full or partial marks in this situation based on what it thought were meaningful comments. However, human marker gives lower marks gave lower marks because the comment is not exact explanation of the parameters.

5.2 C Programming Assignments

The largest percentage of disagreement for marking C assignments was for the commenting mark, where the human marker differed from GAME-2 in 22 percent of assessments. The human marker in 72% of disagreement gave fewer marks than the GAME-2 system. The main reason is that that the GAME-2 could not tell that the comment is related to the function. In these cases the comments were meaningful but the explanation was different to the function/method purpose application. So, after GAME-2 recognized that the comment is meaningful, it provided corresponding marks. However, the human marker provided lower marks because the comment was not describing the function/method operation.

The next component of the structure mark was the programming (algorithm) analysis. In this section, the human marker disagreed with the GAME-2 system 12% of the time. The main reason that the human marker provided higher marks on these occasions was that the human marker examined the algorithm according to the function operation. However, GAME-2 could only identify function operation based on its name. In two assignments, the function name was not related to the function operation. Hence, GAME provided lower marks as compared to the human marker.

6. Conclusions and Future Work

We have designed and investigated and enhanced version of Generic Automated Marking Environment (GAME-2), an automatic system for marking programming assessment in

university level courses. The generic and fuzzy nature of the system makes it a powerful tool for marking different assessment items of varying criteria and languages. The results obtained for object-oriented course assessment and C assignments, comparing GAME-2's performance to a human marker and the previous version of GAME, are very encouraging. In the future, there are many areas to examine in order to reduce the discrepancy between the human marker and GAME-2. Firstly, a more appropriate metric for modifying errors in source code will be used. Secondly, to improve GAME-2's ability to mark the meaning of variables, a schema will be considered which more accurately capture the meaning. Thirdly, a greater number of metrics for algorithm analysis will be investigated to provide adequate feedback to students in order to enhance their programming skills.

Appendix A

This appendix explains, by example, how “fuzzy grading principles” have been applied in meaningful comments. If the number of functions in the project is two, the fuzzy membership function is applied [10] in the following manner. In terms of the number of words, full (10/3) marks is applied when the number of words is in the range of 15 to 20; whereas, a student receives partial marks (50%) in the range of 7 and 25. For less than 2 and more than 30 words a student receives no marks. In the noun and preposition articles component, if the number of nouns and prepositions in a comment makes up 18 to 22% of all words, it was allocated a full mark (10/4). Otherwise, a student receives a partial (50%) mark in the case where 14 to 31% of comment words are noun and preposition articles. For less than 10 and more than 40% of noun and preposition articles out of the total number of words, the mark will be zero.

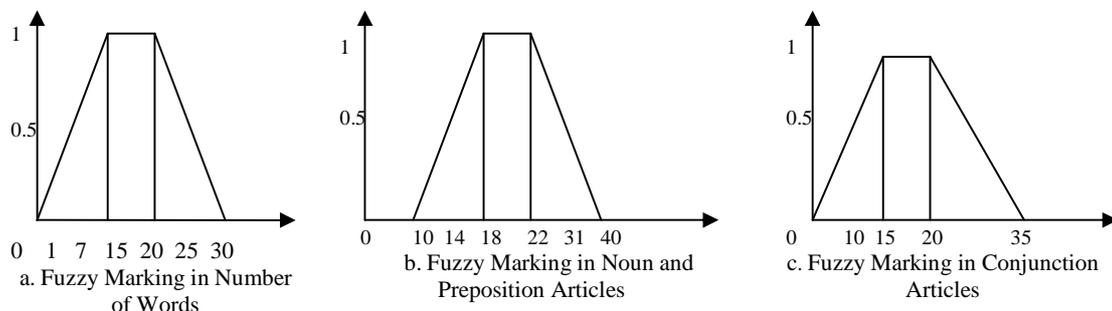


Figure 3 Fuzzy Marking Graph for Meaningful Comment

If the comment includes 15 to 20% of conjunction words, it receives (10/12) of the full marks by the system and partial (50%) marks (10/24) for the range of 10 to 28%, and zero marks for greater than 35% of the total number of comment words.

References:

- [1] Malmi, L., Saikkonen, R. and Korhonen, A., Experiences in Automatic Assessment on Mass Courses and Issues for Designing Virtual Courses, *Proceedings of The 7th Annual Conference on Innovation and Technology in Computer Science Education, (ITiCSE' 02)*, (Aarhus Denmark, 2002), 55-59.

- [2] Jackson, D. and Usher, M., Grading student programs using ASSYST, *Proceedings of 28th ACM SIGCSE Tech. Symposium on Computer Science Education, San Jose, California, USA* (1997) 335-339.
- [3] Hájek, P, What is mathematical fuzzy logic, *Fuzzy Sets and Systems*, Volume 157, Issue 5, 2006, Pages 597-603
- [4] Reek, K. A., The TRY system or how to avoid testing student programs, *Proceedings of SIGCSE 1989* (1989), 112-116.
- [5] Mansouri Z.F, Gibbon C.A. RAM, Higgins, A.C. PRolog Automatic Marker *Innovation and Technology in Computer Science Education (ITiCSE '98)*, (Dublin, Ireland 1998), 166-170.
- [6] Saikkonen, R., Malmi, L. and Korhonen, A., Fully automatic assessment of programming exercises, *Proceedings of The 6th Annual Conference on Innovation and Technology in Computer Science Education, (ITiCSE' 01)* (Canterbury United Kingdom, 2001), 133 136.
- [7] Ghosh, M., Verma, B. and Nguyen, A., An Automatic Assessment Marking and Plagiarism Detection, *First International Conference on Information Technology and Applications (ICITA 2002)*, (Bathurst, Australia).
- [8] Petridis.V, Kazarlis. S, and Kaburlasos.G.V, an Interactive Software Platform for Self-Instruction and Self-Evaluation in Automatic Control Systems. *IEEE Transactions on Education*, vol.46, No 1, (2003)102-110.
- [9] Blumenstein, M., Green, S., Nguyen, A. and Muthukkumarasamy, V., GAME: A Generic Automated Marking Environment for Programming Assessment, *International Conference on Information Technology: Coding and Computing (ITCC 2004)*, (Las Vegas, USA 2004), 212-216.
- [10] J. R. Echauz and G. J. Vachtsevanos, "Fuzzy grading system," *IEEE Trans. Educ.*, vol. 38, (1995), 158–165,
- [11] Jackson, D, A Software system for Grading Student Computer Programs, *Computer and Education*, Vol. 27, No. 3/4, (Grate Britain, 1996), 171-180.

Author(s):

Mr. Roozbeh, Matloobi <Roozbeh.Matloobi@student.griffith.edu.au>

Dr. Michael, Blumenstein <m.blumenstein@griffith.edu.au>

Dr. Steve, Green <S.Green@griffith.edu.au>

School of Information and Communication Technology, Griffith University
 Griffith University, Gold Coast Campus, PMB 50, Gold Coast Mail Centre
 Queensland 9726