Proceedings of the 1st International Conference on EUropean Transnational Education (ICEUTE 2010)

A Tool for the Automatic Comparison of Student Code in the Context of Computational Programming Teaching

J.A. Becerra, F. Bellas, R. J. Duro, G. Varela and F. López-Peña

Integrated Group for Engineering Research, University of Coruna

emails: ronin@udc.es, fran@udc.es, richard@udc.es, gvarela@udc.es, flop@udc.es

Abstract

In this paper we address the problem of motivating students to improve their programming practices through the development of a specific tool that allows them to compare the efficiency and structure of the codes they generate against those of other students or the references provided by the teacher. This tool has been created in order to serve as a competitive incentive or game that forces the students to stop and consider the problem and techniques for creating code that, in addition to performing the task it is supposed to perform, does it in an efficient manner. The tests performed with the tool have shown that students that use it improve their programming style and actually have more fun through the competition the tool induces.

Key words: programming, efficient code, teaching programming skills

1. Introduction

In the realm of the new European System of Higher Education there is a large emphasis on having the students work by themselves and in groups in order to acquire the skills they will require. In particular, in the context of Engineering, and, specifically, in teaching first year engineering students algorithms and basic programming, student work and practice of programming skills become of paramount importance. These skills can only be acquired through practice and a need arises to motivate students to go beyond simply learning to program a computer to do things and address the problem of how to do it efficiently. One standard way to motivate students to program is to do it through games and similar interactive approaches. However, if what is required is an improvement of programming style leading to better and more efficient programs, some type of feedback must be provided against which the student can calibrate his/her evolution. To this end, strategies based on comparison and competition would be desirable. Also in a transnational setting, where students from different origins and not always using the same language must work together or compete, graphical user interfaces that provide visual and numerical information for the competition games they are performing aid in bonding and teamwork.

To obtain a higher motivation from students is a crucial topic in modern learning methodologies. For example, [1] is a state of the art reference of how motivation is a crucial factor in many theories of learning. In the particular case of e-learning, there are several classical empirical studies about motivation. Malone and Lepper [2] present a theoretical framework of intrinsic motivation in the design of educational computer games. Garris et al. [3] found that incorporating game and competitive features into instruction increased motivation and consequently produced greater attention and retention. Our experience on motivating students is based on using elements that are familiar to them in the non-academic context [4].

To program well and efficiently has a lot to do with how the code is structured and how to evaluate the efficiency of the code when run. Thus, what we propose here is to develop a tool that can compare two sets of code for the same program and provide feedback on structure and efficiency in terms of memory and CPU usage as well as other relevant parameters so that a student can compete with the teacher written program or programs written by other students or even previous versions of his/her own programs in terms of these parameters and thus extract better programming practice and skills.

There are different tools in the literature that are related with the comparative analysis of programs with didactic objectives such as the work by [5] on C++ automatic program style assessment or that of English [6] on the automated assessment of GUI program using JEWL. In particular, and related with the C language, there are typical “profiler” or “inspector” programs that have been around since the eighties [7][8]. However, a comparative analysis of these solutions shows that they are not adequate for the purpose we seek. This is not because the information they provide is not useful, but rather that the way this information is presented requires the programmer to be an expert, which defeats the purpose. Here we are seeking to teach students to program and thus require tools that are accessible to these novel programmers.

Thus, what is required is to develop a new application that combines the most important features of other existing applications, mainly “gprof” and “mpatrol”, preferably using free software tools and through the implementation of a JAVA applet in order to make it independent from the operating systems and allow the students to use it within or outside the classroom using just a web navigator.

2. Particular Context

This tool is aimed at courses in the first year of engineering programs that are devoted to providing Industrial Engineering students with a basic knowledge on software programming. Currently, these contents are of fundamental importance for their working future and, consequently, the practice sessions this proposal addresses are very relevant in this context. The courses considered here last one quarter and involve quite a low teaching

load for the large amount of contents that must be addressed. For this reason, the teaching team is forced to seek and propose new didactic methodologies that accelerate concept understanding and skill acquisition, concentrating on motivating students to practice (it is well known that in programming as in so many other areas, practice makes perfect). In addition, we must seek strategies that motivate the students to work on their own and try new things.

In the last two years we have detected that the emphasis placed on maximizing content has led to a decrease in the depth to which the students go into other aspects that are also important in their programming: the efficiency and structure of the programs developed. This topic is of special relevance in the case of industrial engineers who must often deal with programming systems with very limited computational resources in industrial environments or have to address very complex problems where the efficiency in the implementation of the algorithms is of paramount importance in order to obtain a solution in a reasonable time frame.

Objective indicators of this situation are the programs the students turn in throughout the course (autonomous work) and the exams, where we have detected that the students handle advanced programming concepts (pointers, dynamic memory allocation, etc.) but at the same time are not capable of creating efficient algorithms or even detect implementations that are not efficient. This leads to programs that use more memory space than necessary and that are very slow to run. This seems to be a generalized situation, and, as a consequence these aspects are seldom taken into account when evaluating. In addition, as the real time the teacher spends with each student in practice classes is quite limited, it is not feasible to promote these aspects directly through individual tutoring.

This are the reasons the project we describe here was initiated, so that we could provide the students with a software module implemented within the tools they are already familiar with and that could be used to provide them feedback on how they were doing efficiency wise through the extraction of statistics about their programs. In addition, this tool provides a way to establish a motivation to improve through competitive comparisons: It also allows for a simple way of establishing thresholds for evaluation purposes. This permits the evaluation by the teachers of how the students are progressing both in an individualized and a collective manner leading to possible real time modifications of the exercises the students must carry out in order to improve their evolution.

3. Application

The program analysis module has been designed and implemented in C. Its validation has been carried out during the practice classes this year. From these results some improvements have been made and there is now a stable version of the tool available for its use in class.

The basic premises for this application, after analyzing the existing technological options related with the comparative analysis of C programs for teaching purposes, were that none of the profiler/inspector type programs found satisfied the requirements due to the fact that they were oriented towards expert programmers and not novel programmers, which is the case here. Summarizing, and as commented above, the basic requirements are:

1. A need for a tool that combines the most important features of other existing applications, mainly “gprof” and “mpatrol”,
2. Based on free software tools
3. Implementation in the form of a JAVA applet in order to make it independent from the operating systems and allow the students to use it within or outside the classroom using just a web navigator.

Some more specific design decisions were the following:

- Separate the functionalities related to loading files from those of the results.
- Force the comparative analysis using two programs. The applet does not allow the analysis of code in an independent manner. It always requires loading two programs to be compared and all the results are presented taking this into account.
- Permit data input from files or directly through the console.
- Consider compilation options.
- Show the following comparative results:
    - CPU time
    - Execution time for each function
    - Size of the executable file
    - Number of variables used for each basic type in C
    - Number of code comments used

A JAVA implementation of the tool was carried out with these specifications using the Netbeans IDE. The two basic analysis tools used were the “gprof” profiler and the “mpatrol” memory analysis tool. Regarding these two tools:

- “gprof” (www.cs.utah.edu/dept/old/texinfo/as/gprof_toc.html)is a profiling tool, that is, a tool that allows analyzing the efficiency of a program, providing data on how long it requires for its execution (separated into user time and system time), how long it spends on each function, etc. It is the standard de facto free tool. It is a command line program and although there are programs that provide a graphical interface for “gprof”, in this project the command line version was invoked and used from the applet in a user transparent manner.

- “mpatrol” (mpatrol.sourceforge.net/)is a tool that permits detecting different memory problems, such as memory leaks, buffer overflows, accesses to memory blocks that have

already been freed, incorrect use of pointers, etc. Additionally, it can be used to obtain statistics on memory used, functionality we have made use of. Mpatrol was chosen over Valgrind (valgrind.org/), which is a better tool, because its most recent versions do not work correctly in the Windows environments, which is the operating system in the practice classrooms being used.The next few sections are devoted to providing a more detailed description of the operation of the tool.

1. Applet Initiation

The first thing that must be done in order to run the application is to load the applet in a navigator window, accessing the (temporal) url: www.gii.udc.es/comparador_codigo. This operation starts the graphical interface shown in figure 1.

2. File Handling

Two files with the source code of the programs that the user wishes to compare must be loaded. This would usually be the solution proposed by the student and the reference solution provided by the teacher. If the execution requires console inputs, these will be included in a file thorugh the "Fichero de entrada" (input file) option. If an input data file is required, it will be loaded using the option "Fichero de datos" (data file). Finally, "Ejecutar" (Run) button is clicked. Figure 1 displays an example where two solutions to the same program were loaded. These programs require data input from a file called "punto_silla1.txt".

Figure 1: Main interface with the example described in the text.

3. Displaying Results

The results of the comparison appear automatically if there are no compilation or execution errors (these errors may be checked using the “Ver Salida” button). The results interface is presented in figure 3.

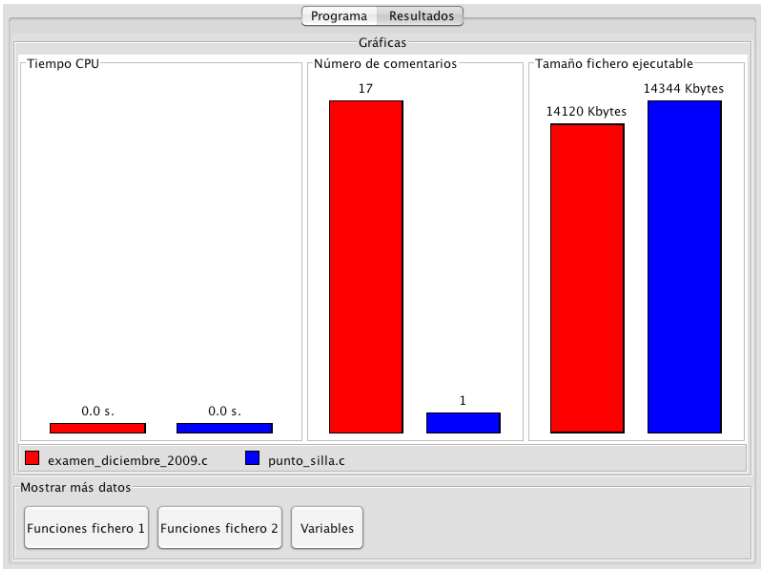

Figure 3: Primary results interface.

As shown, there are three basic areas in the main screen, each one of them displays a bar graph. This makes the comparison between solutions quite straightforward. In this particular example, the execution of the code is so fast that gprof does not generate a CPU usage time. Nonetheless, clear differences can be appreciated in the number of comments and the size of the executable file.

More detailed data on the use of functions can be obtained. For instance, by clicking on “Funciones fichero 2” (File 2 functions) in this example we would obtain the screen shown in figure 4. The table in this figure displays data on the execution time and the number of calls to each function. This information is quite interesting from a pedagogical point of view in order to emphasize to the students the relevance of the correct usage of functions.
On the other hand, within the main results window there is a button called “Variables” that displays the total number of each type of variables used by the user. This window is presented in figure 5.

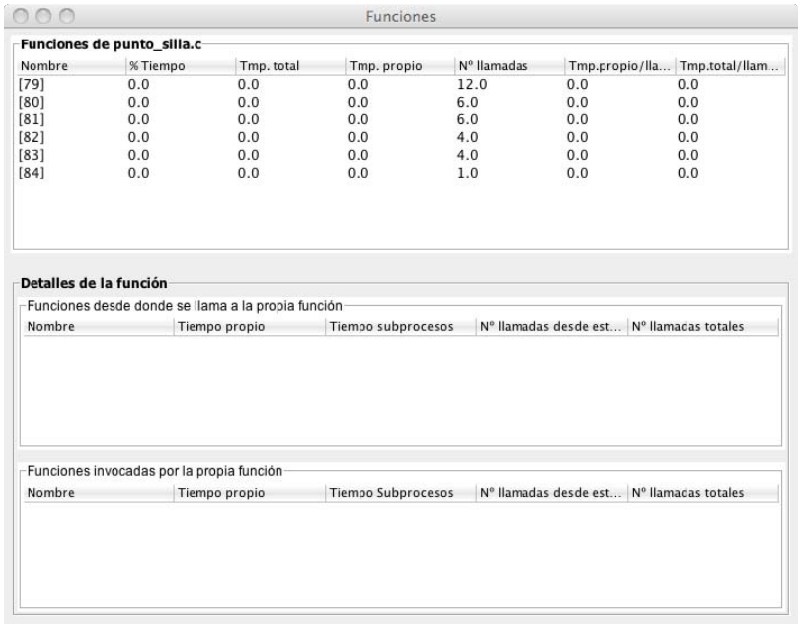

Figure 4: Function information.

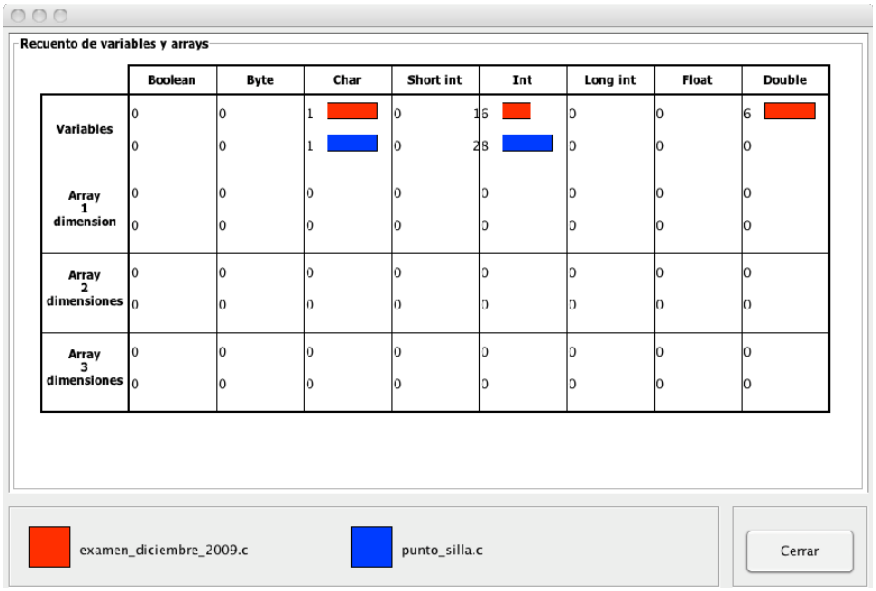

Figure 5: Information on the use of variables and arrays.

Again, it is a simple and clear way of comparing programs, in this case with respect to the use of the most adequate variables, which is an aspect that is quite important during the course.

4. Discussion and Conclusions

In this paper we have presented a tool that was developed with two objectives in mind. On one hand we wanted to provide the students with feedback on the efficiency and good programming practices of their programs and on the other we wanted to motivate them to try out modifications and different approaches to the same program so that they could improve their programming skills. The tool is focused on the comparison of different code versions for the same program it provides information on memory usage, variable usage, comments and functions and execution times of the programs. It can be used in order to set up a competitive environment where the students compete to produce the most efficient or fastest code or it can be used as a feedback tool where a given student can try out different programming strategies for the same program and experiment on the consequences of doing things in different ways.

The results obtained from the application of the tool indicate that the students achieve a better understanding of the elements that go into making a program better and faster and in the end produce better programmers in a more autonomous way, without the teachers having to tutor the students throughout the whole process, something that would not be currently practical due to the teacher hours per student ratio in Spanish universities.

5. References

[1] C KATZEFF *The design of interactive media for learners in an organisational setting – the state of the art*. In Proceedings for NordiCHI (2000).

[2] T.W. MALONE, & M.R. LEPPER, *Making Learning fun: A Taxonomy of intrinsic motivations for learning. Aptitude*, learning and instruction. Volume 3: Cognitive and affective process analysis, (1987) 223-253.

[3] R. GARRIS, R AHLERS, & J.E. DRISKELL, *Games, motivation, and learning: A research and practice model*. Simulation and Gaming, 33(4), (2002) 441-467.

[4] P.DUFFY *Engaging the YouTube Google-Eyed Generation: Strategies for Using Web 2.0* in Teaching and Learning. The Electronic Journal of e-Learning Volume 6, Issue 2, (2008) 119 - 130

[5] K. ALA-MUTKA, T. UIMONEN, & H.-M. JÄRVINEN,. *Supporting Students in C++ Programming Courses with Automatic Program Style Assessment*. Journal of Information Technology Education, vol. 3, (2004) 245-262

[6] J. ENGLISH, *Automated assessment of GUI programs using JEWL*, Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, June 28-30, (2004), Leeds, United Kingdom

[7] S. L. GRAHAM, P. B. KESSLER, M. K. MCKUSICK *An execution profiler for modular programs* Software: Practice and Experience, V 13, 8, (1983) 671-685

[8] B. ZORN AND P. N. HILFINGER, *A Memory Allocation Profiler for C and Lisp Programs*, EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-88-404, Feb. 1988.