FACT: A Formative Assessment Criteria Tool for the Assessment of Students' Programming Tasks

Rami Rashkovits, Ilana Lavy

Abstract—**In this study we present our developed formative assessment tool for homework assignments in computer science and its use. The tool enables instructors to define a list of criteria by which the students' assignments are evaluated. Each assignment may include many problems, each is assigned with specific weights for each criteria. The instructors feed the assessments into the tool adding literal comments and justifications. The tool then generates automatic report to each student including summative report on the current assignment referring to their achievements in each criterion in each problem, the student's relative score and her progress across the criteria along the course timeline. The tool generates automatically charts to present the above information. The tool was examined on a pilot group of college students that study a course in Object-oriented programming. Preliminary results reveal that most of the students were satisfied with the assessment process and the reports produced by the tool. They particularly praised its contribution to their ability to provide solutions that are not only correct but also modular, readable and tested.**

Index terms—Computer-based formative assessment tool, computer science education

I. Introduction

PROGRAMMING is an essential skill for computer science graduates. They acquire their programming skills through a series of courses in which they learn basic and advanced programming concepts. In order to shape their programming skills the students are often required to submit many programming assignment that need to be assessed by the teaching staff. An international study of computer science academics conducted by Carter et al [4] reveals that 74% of respondents assess programming assignments submitted by their students merely for their correctness.

Most educators examine and grade the students' assignment manually, but many prefer automatic tools to ease the efforts required for this task in large courses. The most common technique to test the correctness of the provided solution is to execute it on predefined data and inspect the output compared with the expected results. There are many such automatic tools in use (e.g., Online Judge [5], CourseMarker [8], BOSS [10], Assyst [11], HoGG [14],).

Ala-Mutka [1] describes the methods and techniques used by automated assessment tools and shows how they are generally used. In addition to correctness, some assessment tools analyze program efficiency, coding style and the existence of inline documentation. The use of automatic testing process forces students to be very accurate in order to gain maximal score. However, automatic tools cannot examine whether a variable name is meaningful or if a class inheritance was properly designed. It focuses mainly on the correctness of the solutions and neglects other important properties such as program design (e.g., modularity) and clarity. Moreover, the students conclude that the only factor that counts is the correctness, and hence focus their attention achieving this goal at the expense of other properties. Howles [9] discovered from a student survey that only 5% of the responding students invest time and efforts to design their work before coding and only 39% test their code statically. Majority of the students tested their code dynamically (e.g., unit testing) only sometimes or never. A possible explanation for these results may be that when students' work is assessed by an automated tool, they invest more efforts in design and unit testing, but most students do not devote efforts on these activities as the assessment process do not considered them to be importance.

Formative assessment can contribute significantly to the student learning process with or without the use of automatic assessment. It helps students become more aware of any gaps that exist between their desired and their current knowledge and encourages them to close these gaps during the semester before the final exam takes place. However, the feedback that the students receive on each submission is usually personal and isolated from other feedbacks. For instance, the students cannot learn about their relative score comparing to the rest of the class, or follow their progress across the various assessment criteria from one submission to the next. The feedbacks are usually in the form of one grade and few comments.

In this paper we suggest a novice assessment tool that can serve as a scaffold for the various stages of the assessment

Manuscript received February 28, 2012; revised April 15, 2013.

R. Rashkovits. The Academic College Of Emek Yezreel, Israel (phone: 972-4-6423519; fax: 972-4-6423517; e-mail: ramir@yvc.ac.il).
I. Lavy. The Academic College Of Emek Yezreel, Israel (phone: 972-4-6423520; fax: 972-4-6423517; e-mail: ilanal@yvc.ac.il).

process including designing the list of tasks for the students; setting criteria set for each task and receiving summative reports regarding the class progress along the course timeline. The students can benefit from the suggested tool by receiving the criteria a priory; receiving detailed assessment of their learning progress; explore their relative achievements; and track their progress across the various assessment criteria. To address both the teachers' need for a constructive assessment tool and students' need for meaningful feedback on their assignments, the aim of this study is to examine both the teacher's and the students' impressions of the suggested assessment tool. For that matter, the tool is tested nowadays on a pilot group of education college students studying 'object-oriented programming' course in which they are required to hand programming assignments. In the scope of this paper, we focus only on the students' impressions of the feedbacks received by tool and present preliminary accumulated results.

II. THEORETICAL BACKGROUND

In this section we present a brief theoretical background regarding formative assessment and the various criteria by which programming tasks should be assessed.

A. Formative Assessment

Formative assessment (FA) is considers to be one of the effective assessment techniques since it helps students become more aware of any gaps that exist between their desired and their current knowledge. FA refers to various assessment procedures used by educators during the learning process, aiming at modifying learning activities to improve learners' performance [6]. FA involves the setting of learning goals and the assessment of students' fulfillment of these goals. Effective feedback on students' assignments provides specific comments about errors and specific suggestions for improvement and encourages students to focus their attention thoughtfully on the task rather than on simply getting the right answer [2].

Being aware to the advantages of FA, we developed a computer-based tool to support the assessment process to both teachers and students. Teachers will be able to plan the assignments and criteria according to which their students' assignments will be assessed. They also will be able to feed in their assessments in a standard way and relate to each assessment criterion explicitly. Students will be notified in advance on the criteria their assignments will be examined and receive accumulated feedback represented visually and literally of all the assignments they handed along the course.

B. Assessment of Programming tasks

The solution of programming task usually contains source code developed by the student accompanied with documentation. The provided source code should solve the problem described in the task, and its assessment is based mainly on its success to provide the expected results. In order to evaluate to what extent the provided solution is accurate one has to test the program with various input data and compare the actual results with the expected ones. Although an inspection of the source code can add more insights on the correctness of the solution, many teachers test the provided solutions automatically by running them using predefined input and assign them scores based on their success to output the exact expected results [7].

Although the correctness of the solution is obviously of high importance there are others factors that makes a source code good one [3]. Computer programs tend to be changed often due to maintenance operations such as bug corrections, adapting to new business and technology requirements and improving performance. The source code should allow these maintenance operations to be done easily by programmers other than those who developed the original source code and hence should be clear, modular and include unit tests that cover a lot of code fragments. The clarity of the codes is expressed in its readability (e.g., using spaces and indentations) and its understandability (i.e., using meaningful names to define user-defined constructs such as classes, functions and variables). Modularity refers to the use of separate program constructs (i.e., classes, functions, libraries) to implement different concepts. Modularity is important since it enable programmers to easily track a code fragment that requires modification and isolate the changes done from the rest of the modules. The coverage of the solutions with testing code improves the quality of the source code by reducing the number of software error residing in it, and that might be revealed by the testing code. Moreover, when the source code is changed during maintenance operation the programmer can use the testing code to verify that the modifications did not harmed other parts of the program (i.e., regression tests). Other criteria that may be used to evaluate the quality of the source code are the efficiency of the code (e.g., efficient use of resources such as space and time), the extent to which the student reuse previously developed code, and many other factors.

Students who learn how to program should develop good programming skills from the first computer program they write, and therefore it is desired that other factors in addition to the correctness will be used to assess the programming tasks. FA can contribute significantly to the assimilation of these factors and as a result students will become better programmers. The teacher can assign different weights to the assessment criteria along different problems and task to reflect the focus of the course on various aspects. The student can use the feedbacks to improve and become a good programmer at the end of the course.

III. THE STUDY

In this section we provide a detailed description of our suggested assessment tool and its use followed by a description of the pilot study done to evaluate the tool.

A. The Assessment Tool

The architecture of the suggested tool is based on Microsoft Excel, where its entire logic is developed, using Visual Basic for Application (VBA) to provide all its functionality. This environment was chosen since it is part of the Microsoft office suit and is available everywhere with no special installation needed. The tool was designed and constructed based on our educational perceptions. We believe that assessment process should meet several conditions: (1) the students should be notified on the criteria list by which their assignments will be evaluated in advance. This way they can adjust their learning accordingly [13]; (2) the evaluation process should reflect the students' progress across each assessment criterion along the course timeline, so that they can focus their efforts in issues they encounter difficulties; (3) the evaluation process should demonstrate the student's relative position with comparison to the whole class achievements. This information might help the student to better grasp his learning situation; (4) evaluation via the tool encourage the teaching staff to assess the students' assignments systematically and efficiently. The standard format of the assessment process avoids differences between the provided assessments especially when more than one assessor is involved; (5) the provided assessment should be clear and concise. Therefore, the tool should provide both textual and visual feedback of each assessment criterion.

Figure 1 presents the menu of the assessment tool that enables the teacher to select the desired operation.

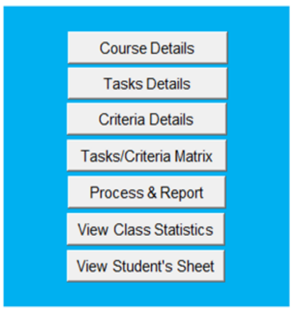
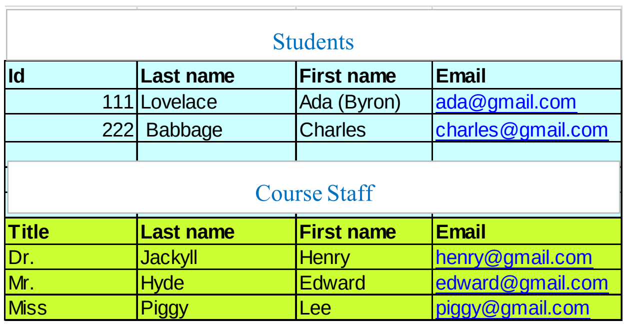


Fig. 1. Menu

In what follows we present a sequence of steps demonstrating the tool's operations.

Step 1: Setting the course details including the course staff (Title, teacher_name, e-mail address) and the students' details (student_no., student_name, e-mail address). Figure 2 presents an example of the course sheet.



Students

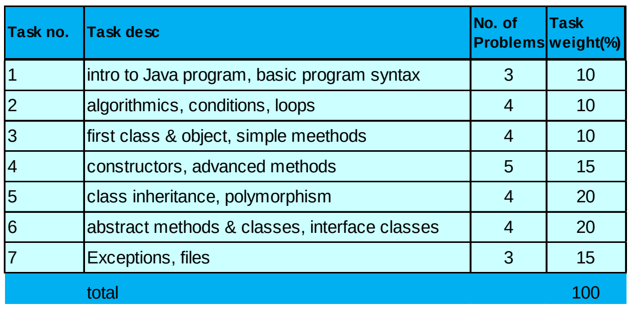
Id	Last name	First name	Email
111	Lovelace	Ada (Byron)	ada@gmail.com
222	Babbage	Charles	charles@gmail.com

Course Staff

Title	Last name	First name	Email
Dr.	Jackyll	Henry	henry@gmail.com
Mr.	Hyde	Edward	edward@gmail.com
Miss	Piggy	Lee	piggy@gmail.com

Fig. 2. Course details

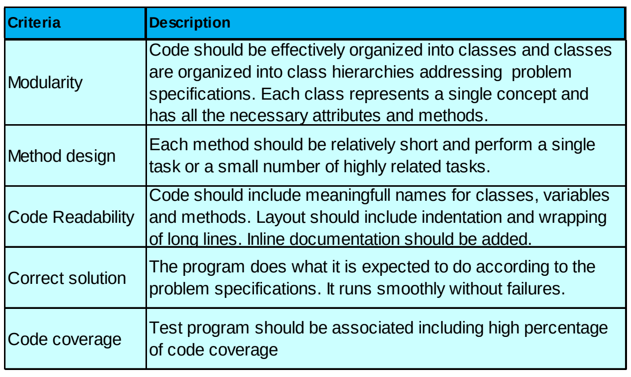
Step 2: Setting the tasks details. The teaching staff has to enter the list of the tasks planned for this course and their relative weight, and the number of problem in each task. Figure 3 presents the tasks planned for the pilot group.



Task no.	Task desc	No. of Problems	Task weight(%)
1	intro to Java program, basic program syntax	3	10
2	algorithmics, conditions, loops	4	10
3	first class & object, simple meethods	4	10
4	constructors, advanced methods	5	15
5	class inheritance, polymorphism	4	20
6	abstract methods & classes, interface classes	4	20
7	Exceptions, files	3	15
	total		100

Fig. 3. Planned tasks

Step 3: Setting criteria list to be used for the assessment of the various tasks. It should be noted that not all the criteria have to be used in each task. Figure 4 presents the criteria according to which the tasks of the pilot group were assessed.



Criteria	Description
Modularity	Code should be effectively organized into classes and classes are organized into class hierarchies addressing problem specifications. Each class represents a single concept and has all the necessary attributes and methods.
Method design	Each method should be relatively short and perform a single task or a small number of highly related tasks.
Code Readability	Code should include meaningfull names for classes, variables and methods. Layout should include indentation and wrapping of long lines. Inline documentation should be added.
Correct solution	The program does what it is expected to do according to the problem specifications. It runs smoothly without failures.
Code coverage	Test program should be associated including high percentage of code coverage

Fig. 4. Criteria set

Step 4: Setting the task/criteria matrix. The teacher assigns relative weights to criteria for each problem in each task. Figure 5 presents the criteria and relative weights of the first task given to the pilot group.

Task 1						
Problem no.	Modularity	Method design	Code readability	Correct solution	Code coverage	**Total**
1	10	20	10	50		100
2	25	15	10	50		100
3	15	15	15	45	10	100

Task 2						
Problem no.	Modularity	Method design	Code readability	Correct solution	Code coverage	**Total**
1	30	10	10	30	10	100
2	25	10	20	40	5	100
3	25	15	10	40	10	100
4	10			90		100

Fig. 5. Relative weights to crite**ria**

Step 5: Task assessment according to criteria – after examination of the students' tasks by the teaching staff, the scores are entered to the suitable sheet and justifications to each score is provided. Figure 6 presents a partial assessment of one of the problems in a task given to the pilot group.

Modularity	**Comments**
21	Class Dog should be extracted from class Animal.
Method design	**Comments**
12	Constructor of Anumal is too long. It should call set methods instead of initializing the attribute itself
Code readability	**Comments**
7	methods' parameters are not documneted. Methods' names must not start with a capital letter.
Correct solution	**Comments**
40	very good!
Code coverage	**Comments**
8	A test with a Cat is missing

Fig. 6. Assessment example

Step 6: Processing task data and generating reports for the students – after all scores and justifications for the problems of the current task are entered, the data are automatically processed and the students receive a report of their achievements by email. The report includes detailed assessment of the current task, literal and graphical description. Figures 7 and 8 present the literal and the graphical assessment reports of one student from the pilot group. In addition, the report includes charts presenting the student's progress across the various tasks in each criterion and the student' relative position in class in each task. Figures 9 and 10 present the progress and the relative position of one student from the pilot group.

Task 3	Modularity	Method design	Code Readabilit y	Correct solution	Code coverage	**Comments**
Problem 1	80	40	80	55	85	Modularity : Class Dog should implement Carnivore interface ; Method design: Animal.eat() is too long, Dog.eat() does not call super.eat(); Code Readability: quite good, but classes should start with a capital letter. Code coverage: tests for Cat and Dog are missing]

Fig. 7. Student's literal report on a problem

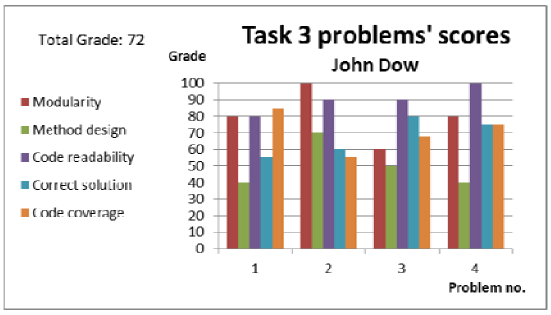


Fig. 8. Student's graphical report on a task

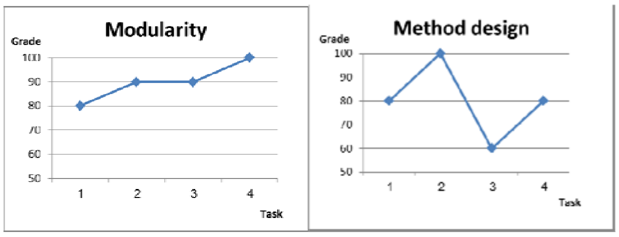


Fig. 9. Student's progress along selected criteria

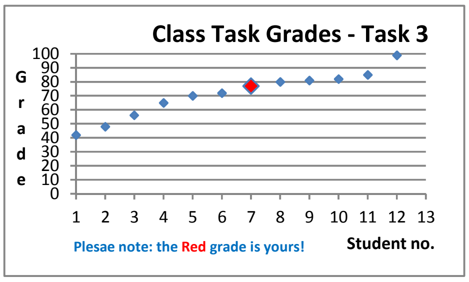


Fig. 10. Student's assessment summary

B. The Pilot

The suggested tool is currently under examination and the results that are presented herein are preliminary. The pilot group includes 45 college students studying towards B.A. degree in management information systems. We used the tool in the course 'Object-oriented programming' which is studied immediately after the course 'introduction to programming' where students learn the basics of programming. The students learn the principles and constituents of object oriented, namely classes, methods, class inheritance, polymorphism, method override, abstract methods, abstract classes, interface class, exception mechanism and graphical user interfaces. The main focus of the course was on using these principles to provide modular, clear and qualitative software solutions to given problems. The course instructor and his teaching assistant planned seven homework assignments each aimed to practice different issues. The instructor defined five criteria by which the students' work will be assessed and assign weights to these criteria for each problem in each task. These criteria and their assigned weights were published to the students in advance. In addition, when new concept (e.g., class inheritance, abstract class) was presented to the students, its contribution to the quality of the code (e.g., modularity, clarity) was emphasized. The teaching assistant used the tool to feed in scores and feedbacks and generated the reports which were distributed automatically to the students via email. Until the time of writing this paper the students received seven reports, one on each assignment. Each report included feedback on each criterion for each problem, including justifications for score reductions and praises for good solutions. The students could compare their achievements to the rest of the students and track their progress from the first assignment until the current one.

IV. RESULTS AND DISCUSSION

As was previously mentioned, this study presents preliminary results. Analysis of the students' responses to the question "describe your experience with the assessment tool", revealed the following issues: reference to the tool's constituents and reference to the assessment process.

A. The Tool's Constituents

Many of the students made statements similar to the following:

David: "*This is the first time that the feedback refers to all criteria explicitly for each problem! Each score reduction is justified. I feel that my work was reviewed thoroughly and with full attention*".

Noga: "*I usually forget the grades I receive in my homework assignments, and forget easily the reasons for loosing score. But this time I could easily remember all the scores and all the reductions, since they were included in each report*".

Tal: "*The combination of literal and visual feedback is perfect for me. I watch the graphs to examine my scores and read the comments to understand the score reductions*".

Evgeny: "*The graph that presents the relative score compared to the rest of the class is most useful to me. I'm very curious about my relative achievements and find it more important than the absolute grade. The higher my relative scores the higher my satisfaction* regardless its absolute value".

Eli: "*The graphs that present the progress along the course tasks provided me a great way to track my achievement and to identify my weaknesses. I immediately saw it on the graphs and could focus my efforts to get better on these issues*".

Most of the students referred to the graphical presentation of the feedback assessment according to the categories saying that it helped them to monitor their efforts to the categories in which they encountered difficulties. Moreover, they pointed out the advantage of receiving all the grades accumulated along the course timeline so that they could track their learning situation.

In the traditional assessment process usually the feedback students receive on their homework includes summative grade for all the included problems and few justifications to explain the grade reductions. In such assessment process it is difficult for the students to figure out what are the specific issues in which they have difficulties. The suggested tool enables the student to follow each criterion in each problem within a certain task along the various homework assignments during the course timeline. The different forms in which the students' progress is presented, helps them realize their accurate learning situation in each of the assessed criteria. The assessment tool also provides the students with the information regarding their relative position in class which can serve as a learning catalyst and motivation for better success. Students tend to appreciate rich and meaningful feedback attached to the scoring of their homework assignments, and feel disappointed otherwise [12].

B. The Assessment Process

Many of the students made statements similar to the following:

Dana: "*Knowing the assessment criteria in advance helped me to improve my answers in all aspects. For instance, I made several passes on the code before submitting; added comments to the code changed variables' names and even broke down long and complex methods into several simple ones just to make sure that I'm not going to lose points for sloppy submission. It surely improved the quality of my solution*".

Ben: "*When I had to provide a source code that solves a certain problem, I saw that the criteria list includes various aspects regarding the quality of the source code such as modularity, readability and good coverage. This list helped me to assimilate these important factors and properly apply it in my solutions"*

Gabi: "*I was quite surprised when I saw the heavy weights the instructor assigned to the readability modularity and coverage criteria. I'm not saying that these criteria are not important, but in the first programming course correctness was the only issue. I had to adapt my coding style to the new requirements*".

Joseph: "*The teacher explained at the beginning of the course that good solution refers to more aspects than its correctness. At first I didn't understand the importance of it but according to the criteria I invested some thinking to create modular and readable code. Now, at the end of the course I can say that I understand much better why these attributes are significant*".

In their previous programming course the students' assignments were graded mainly for their correctness, and as a result they did not pay much attention to readability, modularity and coverage. According to the requirements reflected by the weights assigned by the instructor they had to change their perception on these criteria and indeed provided better solutions. The students understood the importance of the factors that affect the quality of their solutions according to Boehm [3], and assimilated the significance of code clarity and modularity to the future maintenance of the software. They also learned that writing unit test to cover as many lines of code as possible improves the quality of the code and reduce the number of software errors.

Notifying in advance the students about the criteria list by which they are going to be assessed has the following benefits: (1) the teacher conveys a clear message regarding his expectations from students. For example, if the task includes source code, through the criteria list the teacher can convey the students the message that there are another important aspects relating to source code in addition to its correctness; (2) Acknowledging students with the criteria list according to which their work will be assessed can help them better monitor their learning. Via these criteria and relative weights they receive a clear message concerning the relative importance of a certain criterion and the amount of efforts they should invest in it.

V. CONCLUDING REMARKS

The preliminary presented results show that the students have positive experience with the assessment tool. This experience is a results of several factors stated above. The main factor refers to the fact that the tool enables prior notification of the criteria set to the students so that they could adjust their learning efforts. They found the assessments they received to be fair and useful in a way that helped them to focus their efforts in issues they encounter difficulties. Hence, we may say that via the assessment tool the students' knowledge can be shaped. We plan to test the assessment tool on more programming courses such as 'introduction to programming' and 'data structures and algorithms' in which other criteria should be considered. We also plan to extend the tool in the following directions: (1) add summative reports for the teachers; (2) add assessment scale according to which teachers reduce points on faulty or inaccurate answers; (3) add statistics measures to compare the achievements of different groups (e.g., across semesters, across lectures).

REFERENCES

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

[2] Bangert-Drowns, R.L., Kulick, J.A., and Morgan, M.T. (1991). The instructional effect of feedback in test-like events. *Review of Educational Research,* 61 (2): 213-238.

[3] Boehm, B. W., Brown, J. R., & Lipow, M. (1976). Quantitative evaluation of software quality. *In Proceedings of the International Conference on Software Engineering, pages 592*-60*5. IEEE Computer Society Press, October*. Los. Alamitos, CA.

[4] Carter, J., English, J., Ala-Mutka, K., Dick, M., Fone, W., Fuller, & Sheard, J. (2003). How shall we assess this? *ACM SIGCSE Bulletin,* 35(4), 107 – 123.

[5] Cheang, B., Kurnia, A., Lim, A., & Oon, W.-C. (2003). On automated grading of Programming Assignments in an academic institution. *Computers & Education* [4]

[6] Crooks, T. (2001). The Validity of Formative Assessments. *British Educational Research Association Annual Conference,* University of Leeds.

[7] Douce, C., Livingstone, D. and Orwell, J. (2005). Automatic test-based assessment of programming: a review. *ACM Journal of Educational Resources in Computing,* 5(3):4

[8] Higgins, C. A., Gray, G., Symeonidis, P. and Tsintsifas, A. (2005). Automated assessment and experiences of teaching programming. *ACM Journal on Educational Resources in Computing*, 5(3):5.

[9] Howles, T. (2003). Fostering the growth of a software quality culture. *ACM SIGCSE Bulletin*, 35(2), 45 – 47.

[10] Joy, M., Griffiths, N. and Boyatt. R. (2005). The BOSS online submission and assessment system. *ACM Journal of Educational Resources in Computing,* 5(3):2.

[11] Jackson, D., & Usher, M. (1997). Grading Student programs using ASSYST. *Proceedings of the 28th SIGCSE technical symposium on Computer science education*, USA, 335 – 339.

[12] Lavy, I. & Shriki, A. (2012). Engaging prospective teachers in the assessment of geometrical proofs. *In Tso, T.Y. (Ed.). Proceedings of the 36th Conference of the International Group for the Psychology of Mathematics Education,* vol. 3, pp. 35-42. Taipei, Taiwan: PME.

[13] McTighe, J. & O'Connor, K. (2005). Seven practices for effective learning. *Educational Leadership*, 63,(3) 10-17

[14] Morris, D. (2003). Automatic Grading of Student’s Programming Assignments: An Interactive Process and Suite of Programs. Frontiers in Education. 33rd annual S3F 1-6 Volum 3.