ELabMate: A Tool for Delivering Programming Courses Effectively

http://dx.doi.org/10.3991/ijac.v5i3.2189

Rihab Eltayeb Ahmed
Sudan University of Science and Technology (SUST), Khartoum, Sudan

***Abstract*—In the Sudan, at the university level, mastering one of the current programming languages is typically required in order for a student to graduate from computer science majors. In Sudan University of Science and Technology (SUST), Traditional teaching methods of introductory programming courses involve lectures and practical sessions where students and teachers meet and discuss. Other resources are devoted to the courses including free lab sessions and tutorials with supporting staff and teaching assistants. Learning management systems and page-turning courses are available, complemented by a mass of online information, little of which is structured or written to help students learn successfully. To help students in the learning process, more training and practice on lab problems with a guided help is needed in addition to the normal sessions. As a result finding the middle ground between student's needs and the limited staff and time schedules is challenging. In this context we propose the development of an e-learning tool (ELabMate) to provide assistance to students and teachers. The two main potential users of the tool would be students and teachers with a dedicated interface for each, other administrative users can be found with respect to the academic rules drawing attention to the active role of every part involved in the learning process. The goal is to help students learn programming concepts based on assisting and engaging them in their learning process in a way that improves their performance. The tool real innovation is not being a text editor, but the ability to monitor students while writing their code and to provide hints the way the instructors do**.

***Index Terms*—E-learning, Object-oriented programming, Teaching programming, Technology-enhanced learning, Workplace E-Learning.**

I. INTRODUCTION

In the Sudan, at Computer Science/Information Technology colleges and educational institutes, learning and mastering one of the current programming languages is required for a student to graduate. Since SUST believes in the importance of such courses, rules were set to insure this necessity and to *enforce* the importance of various programming courses to the students in the College of Computer Science and Information Technology. For example the Fundamentals of Programming using Java course which is taught in the first year. The course would prevent a student from being transferred to the second year if the student fails, although a carryover is permitted for other subjects. An enhancement to the previous rule was to allow the carryover but not to the fourth class, again preventing the student from graduating unless all programming courses were successfully passed. Conducting similar rules raises importance of the courses, and values more the student's further studies and their future careers.

In a contribution to the development of a quality learning model, resources were devoted to the course including library materials, labs and tutorials with supporting staff and teaching assistants. Learning management systems and page-turning courses were available, complemented by a mass of online information, little of which is structured or written to help students learn successfully.

Researchers have been investigating the causes that lead students to perform poorly in programming courses [1,2]. Finding solutions and alternative learning and teaching methods could be implemented to aid in the learning process. In a study [3], the undergraduate students who were studying programming subjects agreed that among their problems were designing a program to solve a task, dividing the functionality, learning the syntax and finding bugs in the program. In the other hand they have less problems using development environments, and most of them responded that they learn effectively in practical sessions.

As such, these findings were considered a base for the study [3] to further suggest game as another suitable teaching method that allow for practice, examples, feedback and hints. In general Feedback to students is an important part that improves and accelerates the learning process. It needs to be timely, intimate and individual, empower learning and manageable [4].Electronic feedback with its wide range of possibilities speeds up the delivery and reception of feedback and assists with generating quality feedback.

From our experience in teaching programming languages for years, our students need more training and practice on lab problems with a guided help. This should be an addition to the normal sessions, but we can't handle that with the limited staff and time schedules. An e-learning tool is needed to provide assistance to students and teachers [5,6]. In this context we propose the development of an e-learning tool (*ELabMate*) to provide assistance to students and teachers. By integrating the tool in the learning process we then can resolve some programming languages courses problems.

Among the objectives of building such a tool are, to help students learn programming concepts based on assisting and engaging them in their learning process in a way that improves their performance, to help students prepare for tests and earn better grades in the introductory programming course, and to create an intrinsically motivating environment.

The main approach the *ELabMate* follows is to capture the instructor knowledge, provide it to the student in a way that guides their thinking and to provide help when required, the way the thinking, the way training should.

The tool is based on the testing of skills and knowledge. It uses techniques to share knowledge and improve learning and performance. The learning methodology is a mixture of methodologies. Tutorials, problem solving strategies and a notion of coaching are represented in the expert's solutions part in the tool.

In the next section we provide the main concept, in section III we describe the main interfaces in ELabMate the instructor view and the student view. In Section IV we conclude with providing some practical techniques for implementing the tool.

II. TOOL CONCEPT

ELabMate will enable students to learn how to program. They will be able to write their code inside the tool. It is more than a normal text editor. When a student is writing a code to solve a problem, the tool will monitor how students express their idea of solving a programming problem and gives hints at compilation and runtime.

Using the tool an instructor is going to create a problem for the students to practice on. This involves linking each step of the algorithm with every possible code statements that the step can be translated to. These possible code statements will serve as samples, patterns and guidelines to look at. For example if the algorithm step is to read a number from the console, a possible code statement in Java can be:

```
Scanner sc=new Scanner (System.in);
int num=sc.nextInt();
```

The tool will guide the student code writing by the ability to show the algorithm that solves the problem and the possible solutions for a specific step in the algorithm when required. Here the tool is not judging the student code but helping the student to write the closest best solution based on the guidelines provided. In other words *ELabMate* innovation is to react to the student specific educational needs in a specific point in time, not only judging the code after completion. Such approach simulates the job we have been doing as instructors and teaching assistants during lab sessions.

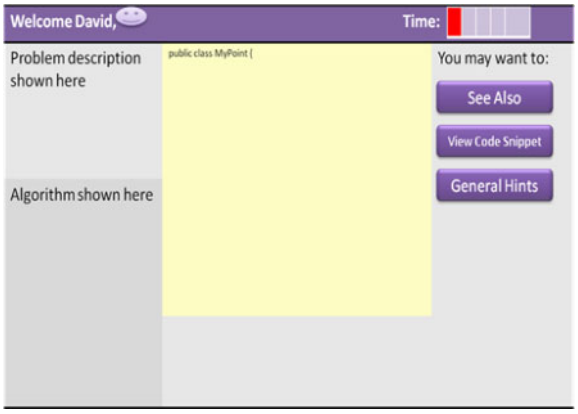

Figure 1 Student Problem Screen / Student Interface.

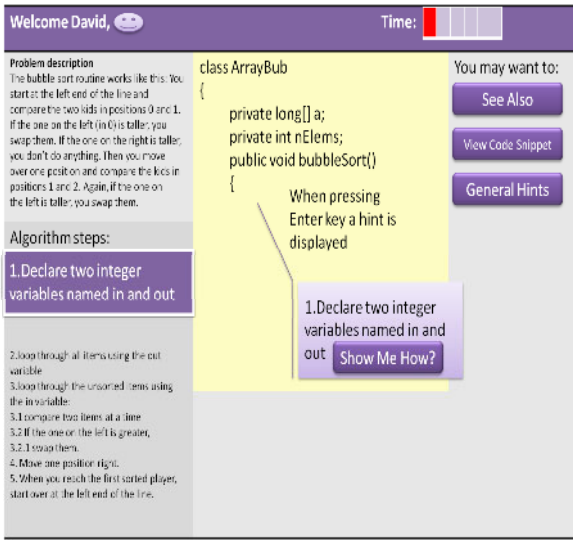

Figure 2 Automatic Hints/Student Interface

It can also be shown how an expert would solve a problem so the student can see how the problem is being addressed from another point of view. This can include looking at a different algorithm, less code statements that produce the same results and a better approach of handling the problem. The point is that the student will be able to compare his code with another so that he can rethink the problem in an efficient way.

Integrated Development Environments (IDE) and text editors such as Eclipse [11] and TextPad [12] provide features that make programming easier. In particular they can provide syntax highlighting, code formatting, editing/debugging, code completion/fixing and more. However, the primary purpose is for the code writing process

These nice features work as time savers and to increase programmers productivity but they fail to notice how the student is handling the programming problem itself, how close to the right solution and specially how to help the student solve the problem. Skills to use an editor or an IDE for code writing is required, but the most required is the effective learning process that leads to the generation (writing) of the code.

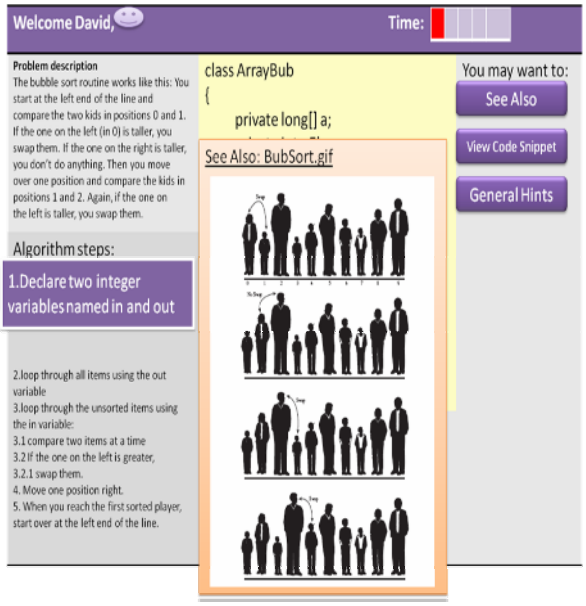

Figure 3: See Also pressed in Student Interface

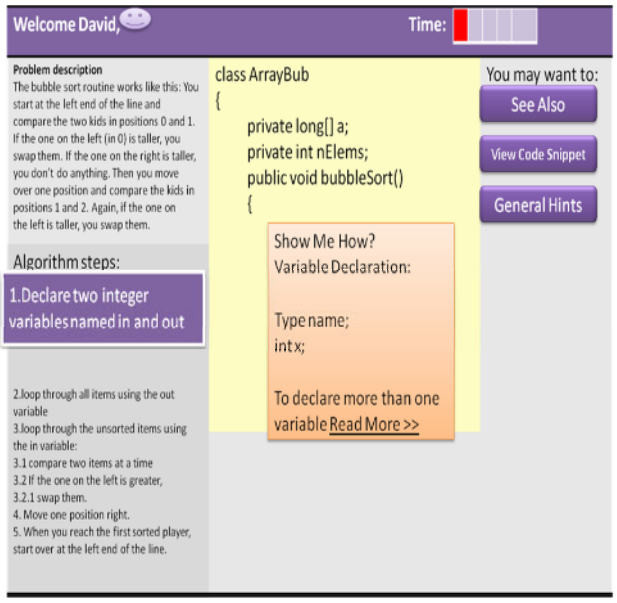

Figure 4: Show Me How pressed in Student Interface

ELabMate is considered as a kind of an intelligent instructor that judges, corrects and helps his students in order to learn how to do programming in a creative way. It would capture the instructor experience and tacit knowledge and present it in a way that benefits the students and eliminate the communication problems.

III. TOOL DESCRIPTION

A. *Student Interface*

Typically, students start up the tool and choose a problem to work on as depicted in fig 1. The interface provides the following functionality:

- Begin with the challenge of the day which is displayed at the first screen.
- Select a specific problem level (simple, difficult etc,) and a specific problem to start solving it.
- Navigate through the resources that the instructor had setup for the problem. The file will be linked to, if it is a link or displayed if it is a text, presentation or video.
- Check the general programming hints.
- View code snippets that might help student get the idea of the code organization, declarations, … etc

ELabMate is going to monitor what the student is writing and supply help when needed. When the student is expected to be in a specific position in the code, the equivalent part from the algorithm is displayed so that the student will know what to do next as depicted in fig 2. This is very important because it will derive the student thinking towards the right directions. One problem the students struggle from is the lack of alignment of their thinking towards the solution. With the algorithm available to them and with linking the algorithm step with the code student is writing, a line will always exist towards the right solution or otherwise student will be able to see the defect. Revealing the misalignments as soon as they appear will minimize the number of errors and prevent them from propagating to the entire code.

When the student doesn't know how to write the equivalent code of a step in the algorithm, ELabMate will respond with displaying a hint that gives more feedback as shown in fig 2. This kind of hints is related to the programming language syntax. The tool will also monitor the compilation and runtime errors and provide hints as illustrated in fig 3. When the student has finished the task successfully, the expert solutions can be shown, so as to compare student work with the ideal solutions, giving the student a chance to enhance the current and future code writing skills.

Example Student Interaction

Let us assume that the student started with the bubble sort problem that the instructor had set. The problem description is displayed. An example description could be:" The bubble sort routine works like this: You start at the left end of the line and compare the two kids in positions 0 and 1. If the one on the left (in 0) is taller, you swap them. If the one on the right is taller, you don't do anything. Then you move over one position and compare the kids in positions 1 and 2. Again, if the one on the left is taller, you swap them.

The student can check the See Also, View Code Snippets or navigate through the General Hint. The "See Also" part in the tool links to the resources supplied with the problem and in a form of a link to a file of any type.

Fig 3 shows an image related to the sorting problem in which an example is given.

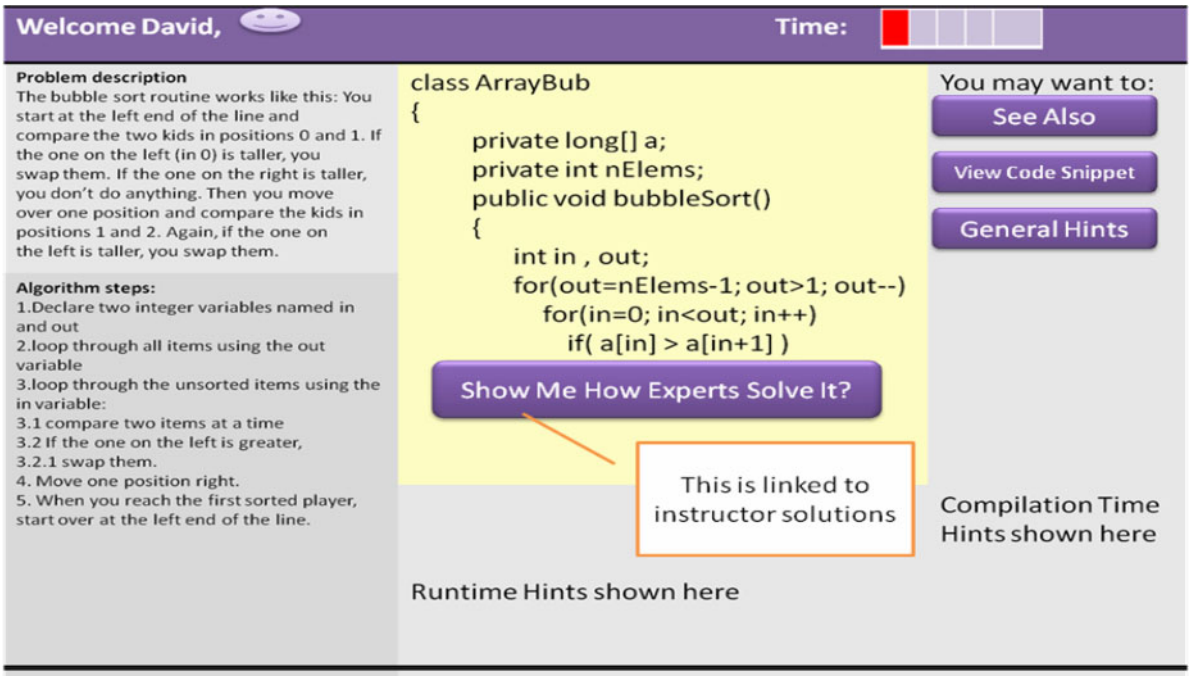

Figure 5: Expert Solution in Student Interface

The algorithm steps are also visible to the student. The algorithm tracking would start from step one as shown in fig 3. Step one indicates that the student is going to declare two integers named in and out respectively. As depicted in fig 2 when the student presses the Enter key in the code writing area, the hints related to the algorithm is displayed accompanied with the Show Me How facility that would display relevant help. In the case of declaring the in and out variables, variable declaration is displayed as shown in fig 4. More links are provided to access more information where the student can learn how to declare both variables in one declaration statement. The student can move to the next algorithm step if finished writing the code for the current one.

Compiling and executing the code are part of the tool options. Compile and runtime messages are visible to the student. A sample solution to the problem can be dispalyed when the time for solving the problem elapsed. Fig 5 represents this option.

B. *Instructor Interface*

The instructor interface is a management interface to manage problems and their related objects such as problem specific hints, files, resources and the general hints objects. Management includes creating and inserting, updating, deleting of those objects.

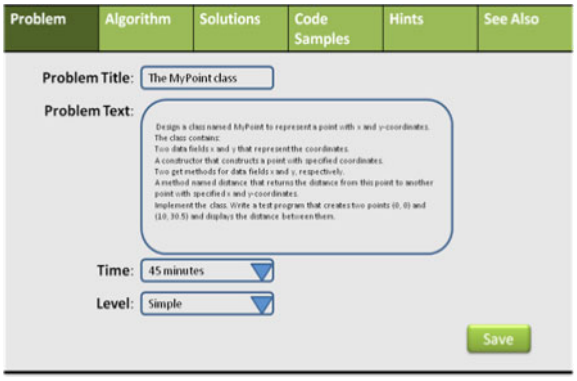

Figure 6: New Problem Creation/Instructor Interface. Note the other tabs related to the problem.

The interface as proposed in fig 6 will help the instructor to achieve the following:

- Create, update, and delete problems. The problem text description is to be provided along with other required attributes such as the time needed to complete a solution.
- Rank a problem to reflect its level of difficulty.
- Prepare the algorithm which is the steps needed to reach a solution, explained in plain text.
- Link the algorithm steps with examples of code statements. The instructor can select from predefined statements or add new statements.
- Supply different code solutions to the problem.
- Prepare complete program examples (code snippets) that enable the student to learn from.
- Provide hints to remind the students about specific statement, coding practices and coding styles. Hints are of two types. The first one is specific hints about the problem itself that are derived from the steps to be done in the algorithm such as a hint that a student would receive to calculate the value of a variable in a specific

step. Specific hints can be generated automatically by monitoring the point where the student is in when writing the code and the steps of the algorithm. The other are general hints that can be applied to different problem constructs such as a hint on naming conventions, spacing and indentation, variable declaration and the use of a specific key word.

- Provide a "see also "link that links to a file (text, html, pdf, video and audio) such as the programming language API as a resource to look at.

IV. DESIGN ISSUES

The here shown sample screens in the various figures for the instructor and student interfaces are an elaboration of what the system can look like. The most powerful attribute of this system is the automatic monitoring and responding to what is expected at that point of code. The whole system should be intelligent enough to guess what the situation is and respond to that effectively. All design issues related to designing an ELearning product should be carefully followed [7].

The management part should be simple and straightforward interface that enables an instructor to do the job efficiently. For example the wizard-like interface is preferable when creating a new problem where save, back and next buttons are shown. The course name, instructor name, user name and password should be considered in the application but not shown here. Emphasizing the active role of every part involved in the learning process, more interfaces can be added to the tool to manage the problems with different roles and privileges such as a teaching assistant role.

V. IMPLEMENTATION TECHNIQUES

ELabMate can be fairly represented as a desktop application implemented using one of the programming languages. Java is a good candidate for implementing the tool for many reasons. Java is a platform independent language that enables installing the tool in various operating systems as long as the java virtual machine is installed. Beside that Java has a light weight graphical user interface capabilities for applications and applets. Many database vendors provide Java enabled drivers for manipulating databases from Java code.

A work in progress implementation of the tool is under development at SUST-CCSIT as a graduation project under the author supervision. The implementation is going to produce a desktop application using Java for coding and MySQL for database end. The produced tool would be assessed by the graduation assessment committee that is constructed from staff and industry members.

Another implementation of the tool can be as a web based application. PHP Hypertext Preprocessor, Java Server Pages, Active Server Pages in .NET framework or any other suitable server side languages can be used in this case. If JSP is considered then the only difference from implementing it as a desktop application would be the web pages versus swing tools in the GUI part of the tool. In other words the core classes that provide the functionality (implementation layer) can be separated from the presentation layer.

Building the text editor part of the tool is not one of the motivations. For example Eclipse provides a general structure for building an editor [8] that can be extended to provide the specific features of the ELabMate around it.

In implementations techniques, the database design and the application's UI and business logic should be carefully addressed and the actual implementation can be mapped to IEEE Learning Technology Systems Architecture (LTSA) [5] that specifies a "high level architecture for information technology-supported learning, education, and training systems".

VI. RECOMMENDATIONS

The tool can be integrated with a Learning Management System LMS such as MOODLE [10], the open source e-learning platform that helps in creating online courses and provides collaboration and interaction. The main reasons are to add more functionality to the LMS, to benefit from the database and management facilities available in the LMS and to provide an effective educational environment.

ACKNOWLEDGMENT

The author would like to thank Dr. David Guralnick who encouraged and gave valuable comments on this work. He also gave us the chance to actively participate and be present on the group of the "Workplace E-Learning: Creating an Effective E-learning User Experience" among the professionals and researchers of several different knowledge areas.

REFERENCES

[1] Steven D. Sheetz, Gretchen Irwin, David P. Tegarden, H. James Nelson and David E. Monarchi Exploring the difficulties of learning object-oriented techniques, *Journal of Management Information Systems JSTOR,*Vol. 14, No. 2 (Fall, 1997), pp. 103-131.

[2] Phit-Huan Tan, Choo-Yee Ting ; Siew-Woei Ling," Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception", *International Conference on Computer Technology and Development, 2009. ICCTD '09*, 2009, p.42-46

[3]

[4] Lahtinen, E., K. Ala-Mutka, and J. Hannu-Matt," A study of the difficulties of novice programmers", in Proceedings of the *10th annual SIGCSE conference on Innovation and technology in computer science education*, ACM Press: Caparica, Portugal, 2005, p. 14-18.

[5] Race, P., "Using feedback to help students learn", *The Higher Education Academy*, 2001, available from http://www.reading.ac.uk/web/FILES/EngageinFeedback/Race_using_feedback_to_help_students_learn.pdf.

[6] M. Ivanović , S. Xinogalos and Ž. Komlenov, "Usage of Technology Enhanced Educational Tools for Delivering Programming Courses," ,in *iJET* – Vol. 6, Issue 4, December 2011.

[7] J.Peter Robinson, "MyPyTutor: an interactive tutorial system for Python," , In: John Hamer and Michael de Raadt, Proceedings Australasian Computing Education Conference (ACE 2011). *Thirteenth Australasian Computing Education Conference*, Perth, Australia, pp.155-160,. 17-20 January 2011.

[8] J Ismail, "The design of an e-learning system: Beyond the hype," , in *Elsevier the internet and higher education,* Vol.4, pp.329-336, 2001.

[9] http://sourceforge.net/projects/eclipseexeditor, last accessed March 2012.

[10] IEEE Learning Technology Standards Committee, IEEE Standard for Learning Technology-Learning Technology Systems Architecture (LTSA), IEEE Computer Society, IEEE P1484.1-2003.

[11] http://moodle.org/, last accessed March 2012.

[12] http://www.eclipse.org/jdt/overview.php, last accessed March 2012.

[13] http://www.textpad.com/products/textpad/features.html, last accessed March 2012.

AUTHORS

Rihab Eltayeb Ahmed is a lecturer and a PhD student at Sudan University of Science and Technology ,College of Computer Science and Information Technology, Khartoum, Box 407 Sudan (e-mail: rahbon@ hotmail.com, rihab.eltayeb@gmail.com). She had completed the design and development of Asynchronous collaborative eLearning System in her postgraduate studies. Her research interests include LMS, Workplace eLearning, Programming languages, Open Source Software and Model Driven Architecture.

This article is an extended version of a paper presented at the conference ICELW2012, held June 2012, at Columbia University, in New York, NY, USA. Manuscript received 19st July 2012. Published as resubmitted by the author 5 August 2012.