Journal of Computer Science 8 (3): 410-419, 2012
ISSN 1549-3636
© 2012 Science Publications

A Framework for Teaching Programming on the Internet: A Web-Based Simulation Approach

Yousif Al Bastaki
Department of Computer Science,
University of Bahrain, Kingdom of Bahrain

Abstract: Problem statement: This research study describes the process of developing a web-based framework for simulating programming language activities on the Internet, in an interactive way, by enabling executable programs to perform automatically their function. **Approach:** The interaction process is played using Java applets. It emphasizes the importance of building the web-based architecture of the proposed simulation model. **Results:** The research concentrates on developing programming courses on the Internet to contribute to the distribution of education for the benefit of learners. We emphasize on introducing interactivity between the user and the programming environment. **Conclusion:** The project is at its first phase and is still under development but we hope that the design of the course and the interactivity that the Java applets provides by simulating the run of an executable C++ code will appeal to our users.

Key words: Web-based, simulation, interactivity, java and internet

INTRODUCTION

Building abstracted models that simplify programming language constructs is considered an essential activity in today teaching processes. Abstracted models enable learners and trainers to practice and visualize actual activities of the underlying programming concepts. The spread of web-based technologies and satellite stations played a major role in spreading news, information and education. From the other side, the outbreak of the Internet enabled the various news, information, technology and education to reach private homes at the speed of light by a click of a mouse. The Internet and the web-based technologies have become popular in a very short period of time and have become part of the information system of many firms, banks, various institutes and agencies as well as many homes all around the world. Services provided by the Internet like the World Wide Web, the electronic mail, the file transfer protocol, education and entertainment are a major attraction on the net. Hence, the Internet performs vital roles in our business world.

This research addresses teaching programming constructs on the Internet in an interactive way by simulating runs for the executable codes on the web using Java applets. Java has been used because of its web-based attractive qualities such as the capability of expressing applications in an object-oriented mechanism that is simple, secure and robust and most importantly Java is platform independent language. This means that just like how HTML files can be read on any platform, Java applets can be executed on any platform that supports Java capable browser. This feature and the built-in interactivity enable Java to provide valuable tools that should be used on the Web. The research also discusses the importance of educational courses on the Internet and the importance of interactive learning process.

The motivation behind building a simulation model is because the proposed system has a complex structure, which cannot be easily described using mathematical models. In addition, computer programs can easily represent the operations of the simulated models.

The remainder of this study is organized as follows. First it provides information on the importance of educational programs on the web. Then reviews the related work that illustrates the drawbacks of the existing educational programs and provides information supporting our approach in directing the research in this article. It also presents the basic of programming language concepts and shows how they are expressed as finite automaton constructs and describes the architecture of the proposed model. This study explains the importance of Java programming language as a tool for implementing the proposed model and performance and reliability issues of the proposed model are outlined. Finally, presents conclusions and outlines possible future work.

MATERIALS AND METHODS

The importance of educational programs on the web: The roots of the web are growing very fast and are now on the doorsteps of every home. It is considered a perfect host for spreading education. Through the Internet, educational programs can reach everyone, from those who are at schools, universities and at work to those who are at homes whether they are students taking up self-study programs, housewives or disabled people who cannot leave their homes or elderly people who are interested in learning new disciplines.

Classroom Internet can now be accomplished through the Web since the Web has an inviting graphical screen layout which supports multimedia provides simplified access and searching of databases that makes learning more accessible. The Internet classroom has also made it possible for universities to teach courses when the instructor was on leave. It has enabled those universities to teach courses to students that need the courses for graduation or as prerequisites even if the university does not intend to offer the course in that semester.

The search engines available with the WWW, the advertisements and the newsgroup can help and guide learners to sites where they can find educational programs of their own requirements. Therefore, the Internet is an efficient media for those who are interested in expanding their knowledge and progressing in their academic life. In addition, the programs available on the web can always be updated, so each time a learner or trainer approaches these educational sites can receive up to date information (provided the creators of those programs do some upgrading).

Also some of these educational programs allow users to email them assuming continuous upgrading is maintained when they have encountered problems (for example, www.cs.uow.edu.au/people/nabg/ABC/ABC.html) and encourage users to provide comments on these programs and to make suggestions for improving the viability of these sites.

Related study: Numerous studies have been conducted on teaching programming languages using the Web-based technologies but many of these studies focus on developing taught material, case studies and the use of Internet for general educational purposes.

Few years ago, the development of computer software and hardware were directed toward education, teaching and learning processes. They have had a tremendous impact on course delivery (Glahn and Glen, 2002; Katz, 2003) in which higher education has been witnessed fundamental changes from courses delivered in the traditional face-to-face method to those delivered via video cassette and television, to a proliferation of courses and course content delivered via computer technologies. In recent years, the use of Internet resources (i.e., web pages) in course and curriculum development has made a significant impact on teaching and learning. The use of the Internet has evolved from the display of static and lifeless information to a rich multimedia environment that is both interactive, dynamic and user friendly (Powell and Gill, 2003). As a result, the use of the Internet in higher education settings has become a more accepted and widely used tool in academia (Glahn and Glen, 2002; Katz, 2003; Angelo, 2004; Hawkins *et al*., 2004; Maslowski *et al*., 2000).

Most recently, the development and refinement of university and commercially developed Course Management Systems (CMS) like Blackboard, WebCT, have resulted in the proliferation of web use in higher education (Morgan, 2003; Angelo, 2004). These technologies have made it possible to easily and efficiently distribute course information and materials to students via the Internet/Intranet and have enabled greater online communication and interaction to occur (Stith, 2000). While these tools were initially developed for use in distance education pedagogies, their use in on-campus classroom setting to compliment traditional courses is now considered a viable and often a preferred option. As a result, many academic departments are struggling to keep pace with the demand for CMS supported course sites for traditional face-to-face courses.

CMS have shown significant increase in student involvement in many aspects of course manipulation (Stith, 2000). The ability of instructors to control access to a variety of course materials-syllabi, lecture notes, outlines and images, allows students access to such material from virtually any location. For the instructor, a multitude of options exist for developing, implementing, revising and delivering course content. At the department level, these tools can have a profound effect on faculty teaching and student learning, departmental communication and faculty workload.

Numerous studies have been conducted on computer-aided learning. Many of these studies focus on technical topics (Chen and Honavar, 1999; Khanjari *et al*., 2002; Brusilovsky, 1994), case studies (Chorfi and Jemni, 2002) and the use of educational packages (Mani *et al*., 1999). Many computer-aided learning packages are implemented in most of higher

education institution around the world (Chrysostomou and Papadopoulos, 2005). Some of the well-known course management learning systems includes Blackboard and WebCT. There are many other packages used by academic institution and industries including coursekeeper, multibook, A tutor and many other systems listed in IMS 2008.

Most of the computerized learning systems tend to facilitate the distribution of structured online courses (Angelo, 2004; Chrysostomou and Papadopoulos, 2005) Making use of modern information and communication have made it easier for faculty to reach out to students at any time, in any place.

Detailed the success of a Computer-Mediated Asynchronous Learning (CMAL) program of graduate studies in Educational Leadership and Higher Education offered through the University of Nebraska-Lincoln. It details the evolution of the concept focusing on an integrated sequence of high-quality learning to:

- Enhance student learning experiences
- Provide greater accessibility by removing barriers of time and space
- Deliver learning opportunities to participants around the world on a conventional university semester schedule
- Develop learning cohorts representing many cultures and nationalities
- Foster active and substantial participation in the learning process
- Provide multiple pathways to learning
- Facilitate the development of a worldwide community of learners

As one can see from the related work that there are no much work has been conducted on using simulation to teach programming languages on the Internet.

Some of the educational programs available on the Internet are a monotonous piece of text associated with some attraction features which are sometimes discourage learners or trainers from getting the benefit of the available teaching material.

For example, the educational program at www.swcp.com/~dodrill/cppdoc /cpplist.htm provides a very good explanation of the functions and constructs of C++ but does not provide any coded example, which will make it difficult for the user to follow the instructions. It also tends to be boring since the web page comprises of a white page with black text displayed all over the page so the user reads a set of explanations as if were reading a text book. This can be seen in Fig. 1, which was copied from the above site.

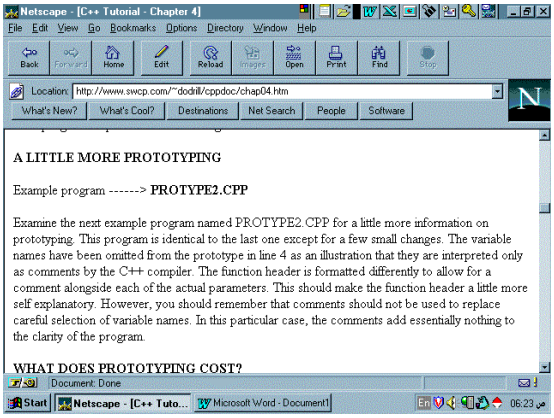

Fig. 1: Explanation of C++ functions

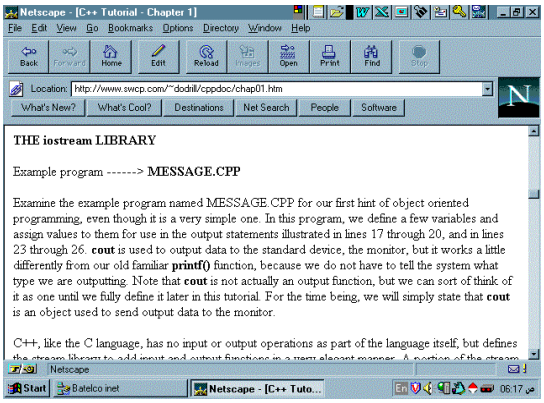

Fig. 2: Interface of some educational environment

Another example is a commercially sold CD for C and C++ tutoring called “C/C++ Interactive Reference Guide” which has an explanation for C and C++ instructions and commands in one section and coded examples in another section. This way a user has to switch from section to section to follow how the instructions are used. Also, the background color is too bright for the user to read the text from for a long period of time.

The environment in which a person is learning in is as important as the quality of the educational material provided. This is because the psychological state a person is in while studying has a tremendous effect on the amount of material he or she will absorb. In any classroom, you will find that a student who participates in class learns more than a student who doesn’t. So we have to somehow create a friendly and effective environment for students/users in which they can interact with the program. Figure 2 shows an educational environment, which does not support friendly interfaces.

Another problem with some books and educational programs are that the text is written in some very difficult technical terms, which the user fails to comprehend which is obviously a total turn off since the commuting property is lost. Other than that, the users should feel that they are learning something useful which will further encourage the user. Therefore, an interactive educational environment with a comprehendible language is necessary.

The emphasis is to write programs that were intended to be read by humans rather than computers. This concept was first coined by Knuth (1992) under the term "literate programming". A program should explain to human what they want the computer to do and to convey to readers the thought processes that led to the program. The proposed architecture aims to automatically extract the main features and mechanisms of programs and displayed to learners. In this way, a learner does not want to go and read information about program from auxiliary documents as programmers usually do (Cordes and Brown, 1991). Therefore, in order to set the stage we need first to look at programming language aspects.

Programming languages and finite automata: Several aspects of programming languages need to be specified. These include (Watt, 1993):

Syntax is concerned with the form of programs. A language's syntax defines what tokens (symbols) are used in programs and how phrases are composed from tokens and sub-phrases. Examples of phrases are commands, expressions, declaration and complete program.

Contextual constraints (sometimes called static semantics) are rules such as scope rules that determine the scope of each declaration and can locate the declaration of each identifier. Type rules enable to infer the type of each expression and thus to ensure that each operation is supplied with operands of the correct types. Contextual constraints are so called because a phrase, such as an expression, depends on its context.

Semantics is concerned with the meanings of programs. There are various points of view on how one can specify semantics. One can take the meaning of a program to be a mathematical function, mapping the program's inputs to its outputs or can take the meaning of a program to be its behavior when it is run on machine.

We can define the grammar of a programming language, G, as a 4-tuple $G = (\sum, N, S, P)$, where $\sum$ is an alphabet, N is a collection of nonterminal symbols, S is some particular nonterminal called the start symbol and P is a collection of replacement rules, called productions, of the form $A \rightarrow w$, where $A \in N$ and w is some string over $\sum \cup N$ satisfying:

- W contains at most one nonterminal
- If w contains a nonterminal, then it appears as the rightmost symbol of w

From the definition we must have that the right-hand side of any production is a string in $\sum^* (N \cup \in)$. Thus the pair (x, y) in $N \times \sum^* (N \cup \in)$ represents the production $x \rightarrow y$.

We can consider the syntax of statements theory, expressed in a terminology that resembles ANSI C/C++ as follows (Okhotin, 2004):

Expr → tId
Expr → tNum
Expr → tLeftPar Expr tRightPar
Expr → ExprFunctionCall
ExprFunctionCall → tId tLeftPar ListOfExpr tRightPar
ListOfExpr → ListOfExpr1
ListOfExpr → ε
ListOfExpr1 → ListOfExpr1 tComma Expr | Expr
Expr → Expr BinaryOp Expr
Expr → UnaryOp Expr
Expr → tId tAssign Expr
Statement → ExprSt | CompoundSt | VarSt | CondSt| IterationSt | ReturnSt
ExprSt → Expr tSemicolon
CompoundSt → tLeftBrace Statements tRightBrace
Statements → Statements Statement | ε
VarSt → tVar – ListOfIds tSemicolon
CondSt → tIf tLeftPar Expr tRightPar Statement | tIf tLeftPar Expr tRightPar Statement tElse Statement
IterationSt → tWhile tLeftPar Expr tRightPar Statement
ReturnSt → tReturn Expr tSemicolon
FunctionHeader → tId FunctioArguments
FunctioArguments → tLeftPar ListOfIds tRightPar | tLeftPar tRightPar
ListOfIds → ListOfIds tComma tId | tId
Function → FunctionHeader CompoundSt & tId tLeftPar all-variables-defind & FunctionHeader returns-a-value
Functions → Functions Function | ε

The proposed architecture provides a number of applets that describe a program structure by displaying a set of the grammar of the language. Applets shows instantiated variable from none instantiated ones. The grammar is used also to return information regarding object constructs. It displays which object inherits

which and what are the components of objects. For example, the statement, FunctionHeader → tId FunctionArguments.

Can return any items of information to the readers. It can return the identifier name i.e., function name through Id and/or details of Function Arguments. This will enhance the understandability of users and enable them to interact effectively with the proposed system. Extracting information from the source code and inserted into a database will enable learners to have a control of the source code. We define a number of basic concepts (Reinfelds, 2002):

- Assignment table σ as a set of slot variables x_1, x_2, …, x_n that are partitioned into sets of equal unbound variables and variables bound to a number, record, or procedure.
- An environment table E as a mapping from variable identifiers to table variables, $\{<x>_1 \rightarrow x_1, \ldots, <x>_n \rightarrow x_n\}$
- A semantic statement as a pair (<s>, E) where <s> is a statement and E is an environment
- An execution state as a pair (ST, σ) where ST is a stack of semantic statements
- A computation is a sequence of execution states starting from an initial state: $(ST_0, \sigma_0) \rightarrow (ST_1, \sigma_1) \rightarrow (ST_2, \sigma_2) \rightarrow \ldots$

Each clause of the above can return information as much as required by the viewer of a program. A variable X, for example, has a scope that determines where in the program this variable is valid. X may meaningfully exist unbound to any value or may be bound to a value via some reference mechanism. X has a name with which programmers refer to it.

An assignment statement links a variable to a value. This definition includes all paradigms. All programming languages have variables. Variables may be linked to values or may be unbound. The difference between paradigms is in the linking rule For example, in imperative languages variables may be linked to values in any place in the program, any number of times. In functional languages a variable may be linked to a value exactly once. In logical programming, variables may be re-linked to values at specific points (choice points) in the program.

Programming aspects of OO concepts: The OO paradigm offers a rich set of problem solving concepts. For example, objects are model entities of the problem area and methods define behaviors that these entities should exhibit. Of course, one can also view objects as bundles of functional behavior whose specific actions are influenced by the internal data of the object. From the programmer's point of view, an object has a global existence that transcends the particular place in the program where the object was created. An object may be passed as an argument and returned as a value. Once created, an object can exist for the duration of the execution of a program. Of course, for storage efficiency, objects that cannot be accessed by any part of the running program are deleted via garbage collection. It is common though not always essential to link objects to variables. These variables have the same block-based scope of definition as all the other named entities of the program, so if a programmer asks the question: "Where can I use a variable that is bound to an object?" the answer is "According to the same block scope rules as for any other entity that is bound to a value." It is important to recognize that the scope of the object, which is global, is not the same as the scope of any particular variable that may be used to refer to the object. If a programmer asks the question: "What methods are available for this object?" the block-based scope does not apply, but an extension of the scope concept does. Whenever it is necessary to decide whether an object may call a particular method, the programmer has to look at the scope of method names of the class from which that object was constructed. In this sense, the scope concept extends to method calls.

Encapsulation, orthogonality and polymorphism: Understanding of the fundamental elements of the programmer's theory of programming will be of little value without a vision of how these elements can and should be composed to form a complete program that is organized in a coherent fashion and capable of being extended in the future. Although it is not easy to solve problems using programs, it is even more difficult to solve problems in a fashion that permits these programs to be adapted cleanly to handle future problems or additional requirements. The essence of program design is to create a program that is so well organized that adaptations are straightforward. Three key concepts are central to quality program design: Encapsulation, orthogonality and polymorphism. The practical usefulness of a programming language is often determined by the degree to which these concepts are supported. The coherence of a program or family of related programs is likewise determined by the degree to which these concepts are utilized. Encapsulation involves capturing some aspect of programming knowledge into an entity that is named and reusable. Here are some simple object-oriented examples: Similar first class entity with almost no effort. To take Java as a

specific example, items 1, 2, 4 and 6 provide first class encapsulations. Methods and functions (item 3) are definable via declarations but are not first class. This is a major source of annoyance in using Java and workarounds, also known as design patterns, are needed to overcome this gap. Finally, contracts (item 5) are only weakly supported in Java so contracts must often be specified in comments not code. Orthogonality is a term borrowed from mathematics to describe the ability of some entity to be varied independently along several axes of parameters. Orthogonality is desirable in programs since it allows the programmer to make independent changes to decisions and settings. Data orthogonality is the ability to modify or substitute data values independently as long as the meaning of the program requires no mutual constraints. Data orthogonality is widely supported in programming languages. Functional or algorithmic orthogonality is the ability to vary functions or algorithms. This type of orthogonality is not so widely supported and so it often becomes important to develop patterns that enable such orthogonality to be achieved in spite of the language. Suppose, for example, that an algorithm is embedded inline in the body of a method in a Java class. It is not altogether easy to change that algorithm. The most common solution to this problem is to define a derived class in which the method in question is overridden to use a different algorithm. This inheritance mechanism may be acceptable if there are only one or a few possible replacement algorithms but becomes unwieldy if there are many possible options. Moreover, if there are two methods with separate algorithms each of which may possibly be replaced, the entire process of inheritance breaks down in excessive complication. A second solution is to use inline definitions that combine the definition of an object with the replacement of one or more of its algorithms.

By modeling the syntax of programming languages and exploring their behavior we can strengthen our trust in understanding of the definitions, concepts and thought processes of large systems (Meyer, 2001; Roy and Haridi, 2002).

RESULTS

The proposed architecture: The proposed system, which teaches programming languages at run time level, uses client/server architecture. The system resides on a server and responds to requests from multiple clients (learners) over the Internet, as shown in Fig. 3.

On the client side, the system (application) is hosted by a browser. The application's user interface takes the form of HTML pages that are interpreted and displayed by the client's browser.

On the server side, the system runs under Internet Services IIS. The Internet service manages the system, passes requests from clients to the application and returns the application's responses to the client. These requests and responses are passed across the Internet using HTTP.

System platform: ASP.NET is used as the platform to create the system and its services that run under Internet services. ASP.NET provides a high level of consistency across Web application development. ASP.NET is part of the. NET Framework and is made up of several components.

A Web application consists of three parts: Content, program logic and Web configuration information. Table 1 summarizes these parts and gives examples of where they reside in an ASP.NET Web application.

The Web form is the key element of a Web application. A Web form is a cross between a regular HTML page and a Windows form. It has the same appearance as and similar behavior to an HTML page, but it also has controls that respond to events and run code, like a Windows form.

In a completed Web application, the executable portion of the Web form is stored in an assembly (.dll) that runs on the server under the control of the ASP.NET worker process (asp_wp.exe), which runs in conjunction with IIS. The content portion of the Web form resides in a content directory of the Web server, as shown in Fig. 4.

When a user navigates to one of the Web forms from his or her browser, the following sequence occurs:

- IIS starts the ASP.NET worker process if it is not already running. The ASP.NET worker process loads the assembly associated with the Web form

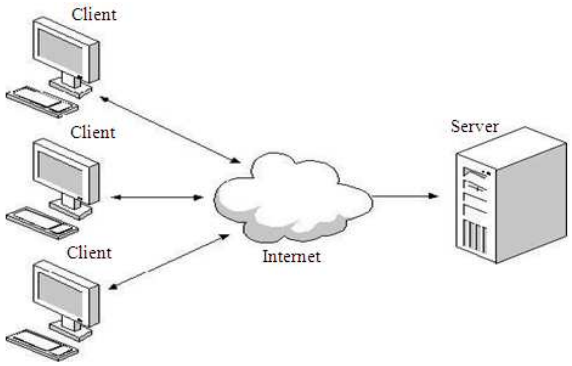

Fig. 3: System architecture

Table 1: Parts of an ASP.NET Web application

Part	Types of files	Description
Content	Web forms, HTML, images, audio, video, other data	Content files determine the appearance of a Web application. They can contain static text and images as well as elements that are composed on the fly by the program logic (as in the case of a database query).
Program logic	Executable files, scripts	The program logic determines how the application responds to user actions. ASP.NET Web applications have a dynamic- link library (DLL) file that runs on the server and they can also include scripts that run on the client machine.
Configuration	Web configuration file, style sheets, IIS settings	The configuration files and settings determine how the application runs on the server, who has access, how errors are handled and other details.

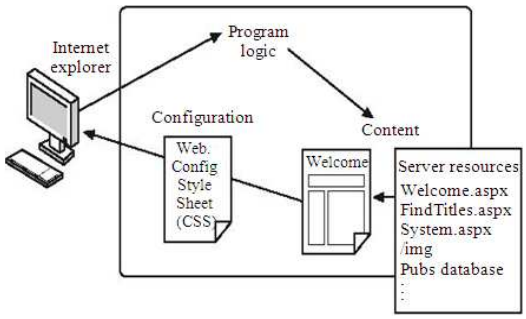

Fig. 4: ASP.NET Web application parts on a Web server

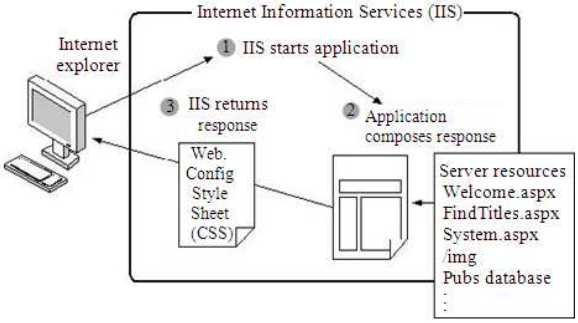

Fig. 5: How the parts interact

- The assembly composes a response to the user based on the content of the Web form that the user requested and any program logic that provides dynamic content
- IIS returns the response to the user in the form of HTML

Once the user gets the requested Web form, he or she can enter data, select options, click buttons and use any other controls that appear on the page. Some controls, such as buttons, cause the page to be posted back to the server for event processing and the sequence repeats itself, as shown in Fig. 5.

Screen shoots: In order to demonstrate the viability of the proposed architecture, we run the system and capture a snap shoot of screens as shown in Fig. 6.

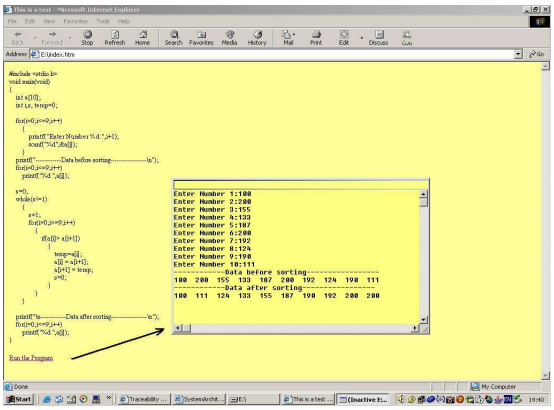

Fig. 6: Screen snap shoot

The proposed approach is a collection of Java packages designed specifically to create executable programming examples. The architecture is made up of a number of components:

- Code viewer
- Terminal emulator
- Variable display component
- Parser of the underlying language
- Information retrieval component

The program execution starts as follows (Reinfelds, 2002):

- The initial execution state is ([(<s>, ϕ)], ϕ). The initial semantic statement is (<s>, ϕ) with an empty environment and the initial table is empty
- At each execution step, the first element of ST is popped and execution proceeds according to the form of the element
- The final execution state is one in which the semantic stack is empty (if it does exist)
- The semantic stack can be in one of three run-time states: Running in which ST can do an execution step, terminated in which ST is empty and suspended in which ST is not empty but cannot do a step

Adding above theory execution concepts to the atmosphere of a teaching process will enhance the proposed system with more attractive and more dynamic. This can be achieved by exploiting the multimedia features, employing a graphical screen layout, decorating the pages or the classroom walls with GIF files (graphical interface files) and maybe some sound in the background which can all be transmitted by the Internet. A number of attributes such as colorful environments, moving pictures and music should be considered to stimulating learners. At the same time the text should be clear and the music level should be appropriate.

The language should be easy to comprehend or else the whole purpose of the educational program is lost, i.e., the users will not find the program useful which will in turn make them seeking other useful material.

Introduce certain degree of interactivity between the course/program and the learner will make the teaching process more effective. Interaction almost always yields good results as far as the learners' gain of knowledge is concerned. With interaction a learner concentrates more on the flow of information provided. The following section discusses how programming languages can be taught in an interactive way on the Internet.

There are many programming courses provided on the Internet and many others commercially produced on CDs, which use various teaching methods. The study material in most of these courses is comprised of very rich and informative text that provides the learner with most required information.

But these programs get monotonous as time passes by and the reading process becomes tedious with limited interactivity and mainly depends on the learners' enthusiasm. On the other hand, educational materials available on CDs and the Internet have various techniques of executing procedures and multimedia features.

We, at the Department of Computer Science at the University of Bahrain are developing a programming course that provides a friendly and lively atmosphere for our users, a readable and comprehendible text which will definitely use loads of example, which is a very effective of teaching students. The text describes the steps in programming and immediately after each step some coded example will be provided and most importantly these can be run online. Yes, the user will run the example online and this will be done using a java code, which yields the same result as the code of the language being taught.

We are going to teach C++ as a prototype of our teaching technique on the Internet and for the first time our users can run a simulation of C++ programs online. This will give our users a better understandability and a more enthusiastic approach to programming. Since java is a modified C++ language the codes of both the languages are going to be similar. The next section briefly describes why Java is used.

Java as a tool: Java is a new object oriented language that is based on C++ with some modifications. Unlike C++, Java does not have any pointer and pointer arithmetic, dynamic memory allocation or overriding operator that makes the language simpler and at the same time robust since these features are the cause of many run-time errors. However, the most attractive feature in Java is that it is portable or platform-independent, that is, Java programs can be moved between different computer architectures. This is possible because compiled Java programs are in the form of bytecodes, which are a set of instructions that look like machine code but are not specific to any processor.

Java programs are basically of two types: applications which are like any general program and applets which is a dynamic and interactive program that is run on a Java capable browser like Netscape 2.0. Applets are inserted into HTML pages just like images but unlike images, applets are dynamic and interactive and can create animation, multimedia presentation, real-time video games, multi-user networked games and real interactivity those allow users to input data and communicate. The browser downloads the applet to the local system and executes it (Lemay and Perkins, 1996; Tittel and Brogden, 1997).

Hence Java applets have the advantage of being able to be run online unlike other languages like C/C++ or Pascal. So Java is an excellent tool that can implement interactivity and can be run on HTML pages and therefore can be used to build interactive Web pages. Its portability and interactivity makes it a perfect tool for our prototype since we can create a simulation of an executable code of C++ online. This will save our users from the hassle of typing down codes, executing them and then running them. The interactivity that Java provides in this way will make our teaching process easier to understand and more appealing to the users since the users will be able to visualize what programs look like and what their outputs are.

DISCUSSION

Performance and reliability issues: The Java programming language and the web browser, implementing the Java virtual machine, are still relatively new.

Fig. 7: Opening screen

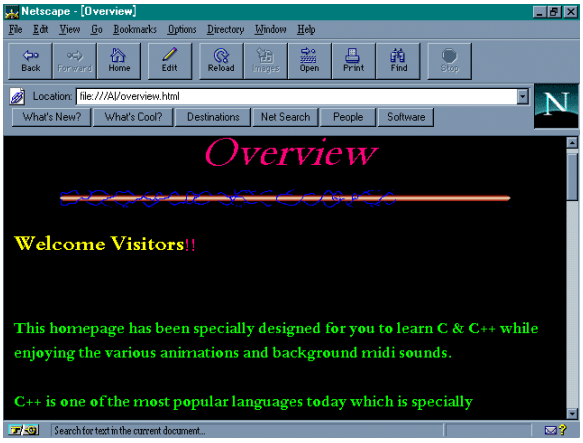

Fig. 8: Welcome message

The course is divided into three levels for users at the introductory level, intermediate level and at the advanced level. The course uses a lot of coded examples to illustrate the use of the various instructions and ore importantly the user can run most of these examples. The executable examples are actually Java applets, which yield the same result as an executable C++ program.

So our user will have an explanation of the C++ instructions in a clear and comprehendible language followed by coded examples for that instruction and a run for the example all on the same web page. The coded examples and the run helps the user in understanding the instructions in a simple way and having them on the same page enables the user to follow what is happening without having to refer to another page or section to see the coded examples.

The opening page of the course is a web page containing the title of the course and is decorate with entertaining GIF animations and with background music embedded using MIDI files which can be seen in Fig. 7. This page is linked to the "overview" page (Fig. 8) that contains an outline of the course and the user then links to any of the three levels (introductory, intermediate or advanced) he/she chooses to take. Each level is divided into a number of lessons, which the user can link to from the "Content" page of each level. The introductory level is for learners who are new to C and C++ and will cover the basic command such as the keywords, simple input and output, data types, arithmetic and logical operators and their precedence, expressions and assignment statements and conditional statements. The intermediate level has been designed for those of you who are familiar with C++ or C and we will start with the different types of looping in C and C++. Then we will move on to functions, arrays, pointers and strings, files and structures. The advanced level will start with classes and objects, then we will move on to linked lists and we come back to cover some special features of classes.

All the pages are decorated with animated GIF files and have background music, which are mainly MIDI files to make the pages more appealing to our users.

CONCLUSION

This study described the processes of building a web-based framework for simulating programming language activities on the Internet in an interactive way through the dynamic running of executable programs using Java applets. The approach emphasizes the importance of building the web-based architecture of the proposed simulation model for online learners.

Although learning on the Web is not a new idea and there is a variety of educational programs available on the Internet, many of these programs do not appeal to users due to lack of interactivity and the absence of a user-friendly interface. In addition to that these courses/programs tend to become "boring" which could discourage the user from continuing with it due to lack of proper code visualization.

This new methodology is aimed to developing programming courses on the Internet to contribute to the distribution of education for the benefit of all learners. We emphasize on introducing proper interactivity between the user and the program.

REFERENCES

Angelo, J.M., 2004. New lessons in course management. University Business.

Brusilovsky, P., 1994. Student model centered architecture for intelligent learning environments. Proceedings of the 4th International Conference on User Modeling, Aug. 15-19, User Modeling Inc., Hyannis, MA, USA., pp: 31-36.

Chen, C.H. and V. Honavar, 1999. A neural-network architecture for syntax analysis. IEEE Trans. Neural Netw., 10: 94-114. DOI: 10.1109/72.737497

Chorfi, H. and M. Jemni, 2002. Evaluation and perspectives of an innovative Tunisian e-learning experimentation. Proceedings of the International Advances in Infrastructure for e-Business, e-Education, e-Science and e-Medicine on the Internet, L'Aquila, Jul. 29-Aug. 04, Italy, pp: 1-7.

Chrysostomou, C.P. and G.A. Papadopoulos, 2005. An evaluation of e-learning technologies and trends: Establishing an object-oriented approach to learning object design and development. The Pennsylvania State University.

Cordes, D. and M. Brown, 1991. The literate-programming paradigm. Computer, 24: 52-61. DOI: 10.1109/2.86838

Glahn, R. and R. Glen, 2002. Progenies in education: The evolution of internet teaching. Commun. College J. Res. Prac., 26: 777-785. DOI: 10.1080/10668920290104868

Hawkins, B.L., J.A. Rudy and J.W. Madsen, 2004. EDUCASE core data service. 2003 Summary Report.

Katz, R.N., 2003. Balancing technology and tradition: The example of course management systems. EDUCAUSE Rev., 38: 48-54.

Khanjari, Z.A., F. Masoud, K. Swami and M. Hatim, 2002. Plugging Software Tools in the Existing eLearning Portals. Proceedings of the 2002 International Arab Conference on Information Technology (ACIT2002), Qatar University, Qatar.

Knuth, D.E., 1992. Literate Programming. 1st Edn., Center for the Study of Language and Information, Stanford, Calif., ISBN: 0937073814, pp: 368.

Lemay, L. and C. Perkins, 1996. Teach Yourself Java in 21 Days. 1st Edn., Sams. Net, Indianapolis, ISBN: 1575210975, pp: 527.

Mani, S., W.R. Shankle, M.B. Dick and M.J. Pazzani, 1999. Two-Stage Machine Learning model for guideline development. Artif. Intell. Med., 16: 51-71.

Maslowski, R., A.J. Visscher, B. Collis and P.P.M. Bloemen, 2000. The formative evaluation of a web-based course-management system within a university setting. Educ. Technol., 40: 5-19.

Meyer, B., 2001. NET is coming. IEEE Comput., 34: 92-97. DOI: 10.1109/2.940017

Morgan, G., 2003. Course management system use in the university of wisconsin system. University of Wisconsin System.

Okhotin, A., 2004. A Boolean grammar for a simple programming language. Technical Report 2004–478, School of Computing, Queen's University, Canada.

Powell, W. and C. Gill, 2003. Web content management systems in higher education. Educause Q., 43-50.

Reinfelds, J., 2002. Programming as an engineering discipline. Proceedings of the 32nd Annual Frontiers in Education, (AFE'02), IEEE Xplore Press, pp: F2G-5-F2G-9. DOI: 10.1109/FIE.2002.1158173

Roy, P.V. and S. Haridi, 2002. Teaching programming broadly and deeply: The kernel language approach.

Stith, B., 2000. Web-enhanced lecture course scores big with students and faculty. T.H.E. J., 27: 20-22.

Tittel, E. and W.B. Brogden, 1997. Discover Java. 1st Edn., IDG Books Worldwide, Inc., Foster City, CA., ISBN: 0764580248, pp: 312.

Watt, D.A., 1993. Programming Language Processors: Compilers and Interpreters. 1st Edn., Prentice Hall, New York, ISBN: 013720129X, pp: 452.