VIRTUAL PROGRAMMING LAB FOR ON-LINE DISTANCE LEARNING

作者：**Weidong Cao1, Jiannong Cao2, and Alvin Chan2**
1Department of MathematicsJiangsu Institute of EducationNanjing, Jiangsu, ChinaEmail: wdcao@jlonline.com 2Internet and E-Commerce Lab, Dept. of ComputingHong Kong Polytechnic UniversityHung Hom, Kowloon, Hong KongEmail: {csjcao | cstschan}@comp.polyu.edu.hk

Abstract

In this paper, we describe the design of the WebVL system, a Web-based Virtual Programming Lab for on-line distance learning. The underlying setting is a collection of computers hosting education and programming software. Students use personal computers at home to access over the Internet one of the Lab servers, which is typically located at a school. The Lab server performs functions to accommodate the various requests from the student, ranging from downloading software from the school lab servers, working through interactive demonstration and tutorial sessions, and submitting a program for execution on a Lab computer using specified software package.
The collection of computers forms a virtual programming laboratory because the machines can be located in different buildings and even at different campuses. It facilitates resource sharing among different schools and overcomes the limit of geographical distances. The WebVL system design includes an agent-based client side functions and interface, the structuring of Lab servers, the mechanisms to locate user requested software packages / services and to present various forms of data and information, and the interfaces to various education and programming software packages. Technologies such as virtual reality, Java Applets and Servlets, mobile agent, XML, and Web/HTTP servers are used to implement the underlying mechanisms and facilities of the virtual Lab, such as software resource locating and interfacing, real-time interaction, and information presentation. Using these technologies allows the system to achieve high-performance, scalability, and disconnected operation through reduction in network bandwidth and delay, load balancing, and code mobility.

1. INTRODUCTION

During the last several years, e-learning has emerged as one of the fastest-moving trends in education and is booming. Thanks to the widespread access to the Internet, on-line education is enabling students and professionals to learn from afar, keeping pace with technological and managerial changes. Thousands of technical and management courses are now being offered by universities, for-profit professional development centers, and industry training facilities worldwide [Ube00].
Underpinning the teaching and learning over the Web, the global connectivity of the Internet and a new generation of hardware and software applications have equipped distance learning with new methods of delivery. The convenience of Web education made distance learning effective and flexible, even in the absence of face-to-face interactions in the classroom. Nowadays, students who take courses on-line can access the courses whenever and wherever convenient. They can download the lecture note from the Web, communicate with each other and their instructor through e-mail, and took exams by responding to questions on computer screens.
Although several e-learning software supporting on-line lecturing and tutoring have been found popular [Web, Wbt, Lea] and many works have been done in providing Web-based learning and teaching [Ibr94, Swi97, Shi00, Ube00], there has been not many reports on providing students convenient on-line access to programming facilities available in computer labs. A few work can be found which develops either ad hoc or special purpose Web programming facilities [Sha97, McI98, Row99]. In this paper, we describe the design of the WebVL system, a generic Web-based Virtual Programming Lab for on-line distance learning. The powerful features of the WWW, especially the integration of most widespread Internet protocols, allow us to design software to facilitate the access to almost existing resources available on the Internet in an integrated fashion [Ibr94]. In addition to navigation through hypermedia documents, using the remote access capability of the Internet technology, various software programs can be executed remotely through WWW. In our design, the underlying setting is a collection of computers hosting education and programming software. Students use personal computers at home to access over the Internet one of the Lab servers, which is typically located at a school. The Lab server performs functions to accommodate the various requests from the student, ranging from downloading software from the school lab servers, working through interactive demonstration and tutorial sessions, and submitting a program for execution on a Lab computer using specified software package.
The collection of computers forms a virtual programming laboratory because the machines can be located in different buildings and even at different campuses. It facilitates resource sharing among different schools and overcomes the limit of geographical distances. The WebVL system design includes an agent-based client side functions and interface, the structuring of Lab servers, the mechanisms to locate user requested software packages / services and to present various forms of data and information, and the interfaces to various education and

programming software packages. Technologies such as virtual reality, Java Applets and Servlets, mobile agent, XML, and Web/HTTP servers are used to implement the underlying mechanisms and facilities of the virtual Lab, such as software resource locating and interfacing, real-time interaction, and information presentation. Using these technologies allows the system to achieve high-performance, scalability, and disconnected operation through reduction in network bandwidth and delay, load balancing, and code mobility.

The rest of this paper is organized as follows. Section 2 describes the requirements of the WebVL system design. Section 3 presents the overall system architecture and describes the functional components of the system. Section 4 discusses implementation issues. Finally, Section 5 concludes the paper and describes our future work.

2. SYSTEM REQUIREMENTS

We have carefully studied the requirements of student programming exercises in developing the on-line virtual laboratory services. The virtual laboratory system to be developed should support students located in different geographical areas, who need on-line and real-time access to programming facilities from a number of different sources. Students have choices of submitting a program for compilation and execution, obtaining the results, conducting (possibly interactive) testing/debugging runs of programs, reading help files and software manuals, etc. Our main objective of this research is to prove a general framework for Web-based access to programming lab facilities.

Reflecting these requirements, Our design of the WebVL system aims at architecture with the following features::

- Accessibility: The service of provided by the system must be widely accessible. We decided that the service should be provided on the Internet, using WWW to reach to a wide range of students. Students can access WebVL on campus in the physical laboratory, and from their home through dial-up connection using PPP or SLIP, using various platforms. Web browsers also create user friendly environment.
- Easy to use and Effectiveness: The system should provide a user-friendly interface to facilitate ease use of the virtual lab services and should confirm to real-world programming practice. The user interface allows the user to select the desirable services in an integrated manner. It should provide an introduction to the virtual lab system, a comprehensive window-based menu of services, and other relevant information about the virtual lab.
- Interactivity: Interactivity is important when students doing programming exercises using a software. The system should allow the user to conduct interactive sessions with interaction-enabled software.
- Multimedia: The system should provide multimedia information to the user to enhance their comprehension of the information provided. In addition to text description, the system should provide video and audio clips for viewing properties information.
- Efficiency: The system should efficiently locate the software requested by the student, providing resource sharing with location transparency and workload balancing.
- Modularity and extendibility: The system should be designed with modules that interact with each other only through their interfaces so that they can be replaced without affecting other parts of the system. New components and modules can be added and/or new requirements can be satisfied. This also increases the flexibility of the system and its capability of keeping pace with the new features rapidly occurring in the e-learning world.

To achieve the above goals, the structure of the system must be well designed. In the next section, we present the design of the system architecture.

Figure 1: System architecture of WebVL

3. SYSTEM ARCHITECTURE

The WebVL system has a Web-based client-server architecture and consists of three major functional modules: User Interface, Client-side Agent, and Server-side Agent. The primary purpose of the User Interface component is to provide the means for the user to configure the programming exercises and to obtain feedback on the activities and results of the programming. Client-side Agent is responsible of dispatching the user's requests to a suitable site for execution in the virtual lab environment. Server-side Agent receives the requests from distinct students and interface to the requested software for carrying out the programming exercises. Overall, the system provides an environment through which programs may be accessed and controlled interactively. Figure 1 shows the high-level system structure and illustrates the major components and their interactions in the generic virtual lab environment.

The loosely coupled architecture separates out different aspects of interfaces with the user and the software, underlying programming environment, and dispatching of user requests into separate functional modules. Each module is self-contained and can be replaced and extended. This supports incremental development. In the rest of this section, we briefly describe the roles and functions of the modules illustrated in Figure 1.

3.1 Graphical User Interface (GUI)

The student connects with the WebVL system through a graphical user interface. The WebVL system can be regarded as a collection of software tools. The GUI allows the user to select these tools. The user-driven nature of the WebVL system should be reflected in the design of the GUI. The GUI is aimed at providing students convenient access to the virtual lab services and information and enable students to interact with the remote software in terms of configuring the programming experiments, sending parameters and data, and obtaining results. Among the others, the services to be provided by the user interface module should include:

- Provide information about the virtual lab and enables the student to find required lab software;
- Provide access points to the remote software. Once the student has selected the lab software to experiment, the GUI should allow the student to interact with the software.
- Represent and display information about and from the remote software. Ideally, the GUI should mimic the interface of the software.

Figure 2: The structure of the graphical user interface

To achieve these goals, the user interface has been decomposed into four sections: Introduction, Display&Configuration, Control and Information. Figure 2 illustrates the components and functions of the GUI.

3.2 Client-side Agent (CSA)

Figure 3 illustrates the modules and structure of the CSA. The CSA is responsible of taking the student's request for accessing specified software and dispatching the request to a suitable machine in the virtual lab environment. Its task also include communicate with the server-side agent for the representation and transmission of data and commands passed between the client computer and the remote platform on which the requested software runs. It formats the student's input data for transmission to the remote server and converts the message data received from the server into the format suitable for display at the student's terminal.

Figure 3: The structure of CSA

3.3 Server-side Agent (SSA)

The SSA is responsible of carrying out the request from the distant student for programming using specified software at the server site. It interfaces the software for executing commands, invoking functions, and passing data. It also needs to interact with the CSA. Figure 4 shows the structure of the SSA.

Figure 4: The structure of SSA

Software used in a lab can be classified according to the platform on which they are executed and their interfaces: interactive vs. batch processing, GUI-based vs. command line interpreter based, etc. One important task of SSA is to provide a mapping of the interface of the software to that to be displayed at the student's computer. The simplest case is when the software has a command line interface with possibly some parameters. In this case, a "direct mapping" can be easily performed. Things can get much more complicated if the software has its own graphical user interface and its execution requires interaction with the user. In this case, conversion and wrapping up are necessary. Providing a standard interface to software permits flexible linkage of the software into the particular programming course design adopted by individual teacher.

3.4 C-S Transport Protocol (CSTP)

Once the CSA finds the target server site for carrying out the student's programming exercise, it will set up a connection to the SSA at that site. In principle, the interaction between the student and the software is in the form of input from the student and the response from the software. Since the interaction can be of various forms, the CSTP is needed to provide a communication environment necessary to facilitate the interactions. The CSA and SSA communicate using the C-S transport protocol which provides a message passing facility that permits data and commands to be encoded and transmitted within Web browsers and accessed by CSA and SSA. The CSTP should have the property that all data saved as the CSTP specific format should be able to be accessed by CSA and SSA who know how to recreate the data from this. This will facilitate the mapping of the lab software's interface objects at the student's computer.

4. IMPLEMENTATION CONSIDERATIONS

In this section, we discuss the issues and considerations in the implementation of the proposed WebVL system using current technologies, which provide cross-platform and worldwide accessibility, support for "interactive" documents, ease of integration of multimedia/hypertext materials, and sophisticated network and database connection protocols [Dei00].

Figure 5 shows the physical system implementation model, where students as clients connect to the WebVL system structured as a group of Lab servers. Each Lab server maintains a collection of lab software. Some of the lab software can be partially or fully replicated at the Lab servers.

Figure 5: Implementation model of WebVL

In the implementation of the GUI, major functions of the Web Pages can be implemented using various tools, including Web authoring tools, Java Applets, scripting (e.g., JavaScript), screen-capture software, and VRML. The rooms metaphor, a concept introduced by XeroxParc [Car91, Swi97], can be used to mimic the real-world laboratory environment in terms of interacting with the computer and organizing the lab documents. For example, with the use of Internet-oriented virtual reality technology [Har96, Vrml], the GUI can be designed to provide a illusion that the students enter a lab room to sit in front of a computer to run a program and, at the same time, gain access to different information resources located in the room. Dynamic web pages with increased interaction with the user and sophisticated visual effects can be achieved through the use of JavaScript, DHTML and Applets. Other aspects of the GUI implementation such as two-way dialog with the server (interactivity between the student and the lab software) through the GUI will be discussed in the following sections.

The CSA can be designed as a proxy object to a normal Web browser. To configure the Web browser to enable the WebVL, the user simply configures his/her browser to use locally executed CSA daemon by setting the proxy server as 127.0.1 or localhost. The CSA acts as a mediator between the Web browser's normal Http request/reply and WebVL services on the network. This approach requires no re-invent of a new Web browser. Various task-scheduling techniques with load sharing can be used by the CSA to dispatch the student's request to a suitable site in the WebVL environment for connection. There are several solutions to the problem of providing transparent access to scalable services provided by a group of servers and sharing the load between the different servers. Examples are DNS Aliasing, HTTP redirect, Magic Routers, fail-safe TCP and Active Networks [Ami98, Car99, Yos97].

The interaction between the student and the software is largely supported by Web server and client extensions. Information and feedback can be facilitated by HTML push/pull mechanisms and techniques for local processing of small, embedded applications such as Java Applet. On the other hand, techniques based on the use of forms within HTML documents to let clients initiate and interact with server-side applications enable us to create appropriate tools that can interface Web browsers to software applications. For example, using the forms facility and CGI scripts (Common Gateway Interface), we can build CSA on the Lab servers which provide runtime services to WebVL related requests/replies. The CGI script in Java is called Java servlets, which are Java programs that run in a Java-enabled web server in response to an HTTP request. The web server runs the servlet, and the servlet outputs an HTML page that the server returns to the client [Cam99]. Depending on the lab software to be accessed, the functions of the CGI/Java servlet programs can be as simple as invoking a local command line or shell script program, or as complex as interfacing a sophisticated, interactive software.

A given lab software can be accessed from a command embedded in an HTML document. From the student's point of view, he / she simply follows a link that can cause a WWW server to execute a program or a shell script, which, in turn, can spawn a process/thread that will handle the interactions with the distinct student. The server-side agent, implemented as a cgi-bin or Java servlet program, runs during the whole session and is responsible of interfacing the student with the local software program. It obtains the student provided information, such as program commands, code, parameters, etc, from the fill-out forms, processes it if necessary, and then passes it to the software in the required format. For example, the student can submit Java code for execution in a text area within an HTML form; the server side agent would save the code to disk, attempt to compile the code and, if compilation was successful, run the executable file on some test data. When the output is produced by the software, the server-side agent sends the output to the student, most probably as HTML documents that are dynamically generated. These HTML documents may be just simple text files, or contains fields that have to be filled by the student and then transmitted back to the software program, or links to actual sites of the software vendors. It can also contain embedded links that lead to other executable programs. Figure 6 illustrates the above interaction scheme in WebVL. Note that CGI scripts don't have to be invoked by forms; they can be invoked directly by a web browser.

By utilizing the information expression that Web browsers and the CSTP support, various forms of multimedia data as output from the software can be saved and communicated about over the Web [Nei96]. For accessing lab help facilities such as programming manuals and help files, due to the ease of interfacing with external viewers in WWW browsers, different types of documents can be viewed by using the viewing programs associated to the various types. This has the advantage of extensibility: new types can be defined with appropriate viewers if necessary. Thus, when the student requests to read help files or programming manuals in the software packages, the WWW browsers will retrieve the documents and launch the corresponding viewer for the display.

Figure 6: Interaction of components in WebVL

For implementing the CSTP, we propose to use HTTP as the baseline protocol by extending relevant HTTP headers and the support for the transfer of WebVL specific headers, control information, and data formats. Potentially we can develop a new specification of XML-based [Kha97, Lau99] CSTP transfer syntax. This approach has the advantage of being conforming to standards and portable to various platforms.

We are also considering the use of mobile agents in the development of WebVL. A mobile agent is a computer program that can autonomously migrate between network sites, i.e., it can execute at a host for a while, halts execution, dispatches itself (together with its data and execution state) to another host, and resumes execution there - all under its own control [Lan99]. It has been found that mobile agent is especially suitable for structuring and coordinating distributed applications running in a wide-area environment like the Internet

[Que99]. In our case, for example, mobile agents can be employed to locate the requested software for programming exercises, to collect load information at individual Web servers, and/or to automatically and transparently determine for the client the "best" server to execute a specified task [Cha00]. Mobile agents can also be sent by either the client or the server to perform local interactions or display information at the peer site.

5. CONCLUSIONS

In this paper, we have described the design of a Web-based virtual lab software system, which allows distant students to gain access to various programming lab software by using a standard Web browser such as Netscape and Internet Explorer. The research described in this paper improves the existing work on distance teaching and learning which lack some of the important features that are associated with traditional education activities such as laboratory exercises. By doing so, we are developing special software and laboratory materials designed to facilitate students access to a wider range of information and educational services. Using WebVL, students can use programming lab facilities which they otherwise might not be able to access due to the incapability to be present in the lab or shortage of laboratory time. Furthermore, our design of the virtual programming lab takes into consideration of integrating facilities from several labs, which enables more resource sharing and offsets the high cost in developing programming lab materials by wider usage of the lab materials across different campuses.

Currently, the implementation of the WebVL is undergoing. The various advanced Internet and WWW based technologies discussed in this paper are employed to develop the underlying mechanisms of the on-line virtual laboratory environment. There are challenges and difficulties in building WebVL. For example, we have not arrived at solutions to the general problem of mapping the interface of the lab software to the student-side GUI. The problem is potential inconsistency of user interface, both through different use of metaphors (e.g., use of tabs, colors, and icons) and inconsistent technical standards (e.g., use of frames and determining in what frame indices or content it to appear) [Sha97]. We are actively investigating these issues, starting with simpler cases where solutions exist, e.g., tools to automatically convert programs with MS-Window GUI to fill-out forms coupled with cgi-bin programs [Thr94].

REFERENCES

[Ami98] Y. Amir A. Peterson, and D. Shaw, "Seamlessly Selecting the Best Copy from Internet-Wide Replicated Web Servers", Proc. 12th International Symposium on Distributed Computing (formerly WDAG), Andros, Greece, Sep. 1998.

[Cam99] M. Campione, et al., "The Java Tutorial Continued", Addison-Wesley. 1999. (http://java.sun.com/docs/books/tutorial/)

[Car91] S. Card, G. Robertson and J. Mackinlay, "The information Visualizer, and Information Workspace", Proc. Inte'l Conf. Human Factors in Computer Science, Boston, Mass. 1991. pp. 105-115.

[Car99] V. Cardellini, M. Colajanni, and P.S. Yu, "Dynamic Load Balancing on Web-Server Systems", IEEE Internet Computing, May – June 2000. pp. 28-39.

[CHA00] W.S. Chan, "Distributed Dynamic Load Balancing using Software mobile Agents", Project Dissertaion (under supervision of J. Cao), Dept of Computing, Hong Kong Polytechnic University, May, 2000.

[Dei00] H.M. Deitel, P.J. Deitel, and T.R. Nieto, "Internet and World Wide Web: How to Program", Prentice Hall, 2000.

[Har96] J. Hardman and J. Wernecke, "The VRML 2.0 Handbook, Building Moving Worlds on the Web", Addision-Wesley. 1996.

[Iba94] B. Ibrahim, "Distance learning with the World-Wide Web", Proc. Int'l Conf. On Open and Distance Learning - Critical Success Factors, Oct. 1994. pp. 123-126.

[Iba97] B. Ibrahim, "Use of HTML Forms for Complex User Interfaces to Server-Side Applications", Int'l Journal of Human-Computer Studies (Special Issue on Innovative Applications of the World Wide Web), 46 (1997). pp. 761-771. (http://cuiwww.unige.ch/eao/www/Bertrand.html)

[Kha97] R. Khare and A. Rifkin, "XML: A Door to Automated Web Applications", IEEE Internet Computing, Vol. 1, No. 4, July-Aug. 1997. pp. 78-87.

[Lan99] D. B. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents", Communication of the ACM, Vol. 42, No. 3, March 1999. pp. 88-89.

[Lau99] S. Laurent, "XML A Primer" (2nd Edition), IDG Books Worldwide, Inc. 1999.

[Lea] LearningSpace, Lotus (http://www.lotus.com/home/nsf/welcome/learnspace)

[McI98] D.R. McIntyre and F.G. Wolff, "An Experiment with WWW Interactive Learning in University Education", Computers and Education, 31 (1998). pp. 255-264.

[Nei96] I. Nelson, et al., "Eduction 2000: Implications of W3 Technology", Computers and Eduction, Vol. 26, No. 1-3, pp. 113-122.

[QUE99] P.-A. Queloz, and C. Pellegrini, "Foreign Event Handlers to Maintain Information Consistency and System Adequacy", Proc.

Workshop on Mobile Agnets in the Context of Competition and Cooperation, Autonomous Agents Conference, Seattle, USA, May 1999.

[Row99] G.W. Rowe and P. Gregor, “A Computer Based Learning System for Teaching Computing: Implementation and Evaluation”, Computers and Education, 33 (1999). pp. 65-76.

[Sha97] A. Shabo, M. Guzdial and J. Stasko, "An Apprenticeship-Based Multimedia Courseware for Computer Graphics Studies Provided on the World Wide Web", Computers and Education, Vol. 29, No. 2/3, pp. 103-116, 1997.

[Shi00] T.K. Shih, et al., "An Adaptive Tutoring Machine Based on Web Learning Assessment", Proc. ICME 2000, IEEE International Conference on Multimedia and Expo, 2000. pp. 1667-1670.

[Swi97] K. M. Swigger et al., "The Virtual Collaborative University", Computers and Education, Vol. 29, No. 2/3, 1997. Pp. 55-61.

[Thr94] R.K. Thralls, “Building HTML Application Systems: Converting Existing MS-Windows Applications to HTML”, Proc. 2nd Int’l WWW Conference: Mosaic and the Web, Chicago, Oct. 1994. pp. 17-20.
(http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/WebProd/thralls/WhitePaper.html).

[Wbt] WBT Systems, Topclass, (http://www.wbtsystems.com)

[Web] WebCT, Universal Learning Technology (http://www.webct.com)

[Ube00] Robert Ubell, "Engineers turn to e-learning", IEEE Spectrum Oct 2000. pp. 59-68.

[Vrml] VRML Repository (http://www.web3d.org/vrml/vrml.htm)

[Yos97] C. Yoshikawa, B. Chun, and P. Eastham, “Using Smart Clients to Build Scalable Services”, Proc. USENIX’97. Sep. 1997.