2-2003

Development of a Web-Based Mass Transfer Processes Laboratory: System Development and Implementation

Yusong Li
University of Nebraska - Lincoln, yli7@unl.edu

Eugene J. LeBoef
Vanderbilt University, Nashville, Tennessee

P. K. Basu
Vanderbilt University, Nashville, Tennessee

Louis Hampton Turnver IV
Turner Technology, LLC, 2525 West End Avenue, Suite 930, Nashville, Tennessee

Published in *Computer Applications in Engineering Education* 11:1 (2003), pp. 25-39; doi 10.1002/cae.10036

Contract grant sponsor: National Science Foundation; contract grant number: 0088912.

Submitted November 11, 2002; accepted February 3, 2003.

Development of a Web-Based Mass Transfer Processes Laboratory: System Development and Implementation

Yusong Li,[1] Eugene J. Leboeuf,[1] P. K. Basu,[1] and Louis Hampton Turner IV[2]

[1] Department of Civil and Environmental Engineering, Vanderbilt University, Nashville, Tennessee 37325

[2] Turner Technology, LLC, 2525 West End Avenue, Suite 930, Nashville, Tennessee 37203

Corresponding author—E. J. LeBoeuf, eugene.j.leboeuf@vanderbilt.edu

Abstract

A web-based environment is utilized as a means to introduce advanced mass transfer processes concepts in environmental engineering and science courses. System development and implementation is presented, including detailed descriptions of the techniques employed to link software written in high-level computer languages such as C++ and FORTRAN to an internet-based, user-friendly graphical user interface for both program input and output.

Keywords: mass transfer processes education; environmental engineering; Internet; computer-facilitated learning

Introduction

The field of environmental engineering and science is unique relative to many engineering disciplines due to the myriad of complexity and uncertainty of the natural and man-made systems that it addresses. To provide for effective and efficient solution to environmental problems, therefore, environmental engineers and scientists must first possess a clear understanding of the complex processes controlling the mass distribution, transportation, reaction, and transformation of contaminants within the natural or engineered environment [1]. This inherent complexity emphasizes the need for educators to actively involve students in laboratory experiences, which can imitate the uncertainty and complexity of natural systems, and thus

facilitate a more rapid approach to understanding the physical, chemical, and biological processes that control contaminant distribution [2]. Laboratory sessions have thus been at the heart of environmental engineering and science education [2]. Due to limited resources, including finances, laboratory facilities, and instructional time, however, it is becoming more difficult to provide each student with a traditional laboratory experience for each class. This fact is further demonstrated by an observed trend in reduced traditional laboratory experiences in undergraduate environmental engineering education [3]. The advent of the Internet and other advanced information technologies into the engineering curriculum, however, suggest that web-based laboratories may provide a potential partial solution to this problem [4].

This paper describes the development of a web-based laboratory as a means to introduce advanced mass transfer processes concepts in environmental engineering and science courses. Emphasis is placed on identifying and overcoming many of the challenges associated with linking high-level computer languages to Internet-based systems. Subsequent to this introduction, a brief literature review of learning theory and web-based education is presented, followed by descriptions of system development and implementation, including detailed discussions of the architecture employed to link software written in high-level computer languages like C++ and FORTRAN to an Internet-based system. Techniques employed to construct user-friendly input pages, and customizable output pages are also discussed prior to presentation of Help and Tutorial components. The paper concludes with a summary and identification of needs for future work.

Learning Theory and the Importance of the Laboratory in Engineering Education

Behaviorism, cognitivism, and constructivism represent three fundamental theories of learning processes [5]. Behaviorism theory concentrates on observable behavior, and emphasizes that learning occurs when a correct response is demonstrated following a specific environmental stimulus. The behavioral instructional approach is especially effective in facilitating the learning of introductory-level topics. Describing knowledge acquisition as a mental activity rather than a straight stimulus-response, cognitivism theory views the student as an active participant in the learning process, and recognizes that meaningful and well-organized information is easier to learn and remember. Cognitive education strategies work well in teaching environments associated with strong cognitive emphases, such as analytic reasoning and algorithmic problem solving. Constructivism theory suggests that learning is an active process of constructing a personal interpretation of the world, where learning should take place in realistic settings. Constructivist strategies are especially well suited for advanced or expertise-level knowledge acquisition, such as advanced chemical mass transfer processes, where the student is required to make intelligent decisions within the learning environment.

Constructivism views the learning process as a progression of activities consisting of four separate stages: (i) having an experience; (ii) reviewing the experience; (iii) drawing conclusions from the experience, and (iv) taking an action to confirm the conclusion or generation of a new experience [6]. The aims of constructivist strategies are often synonymous with the aims of engineering education, which demands not only the development of the abilities to accept, evaluate, or use information, but also development of skills in identifying, defining, and problem solving [7]. Following constructivist theory, engineering education can activate and encourage learners to work and learn in new and complex engineering surroundings, both independently and in teams.

Laboratory experiences, which imitate the uncertainty and complexity of authentic life practices, are essential elements in constructivism education [8], and thus engineering education. Several studies [7,9] suggest that laboratory experiences can achieve positive influences on learning skills, understanding concepts, cognitive abilities, and attitudes. For example, Su and Huang [9] illustrated that the quality of the laboratory environment, and the frequency of the laboratory experience, can account for 10–24% of students' attitudes toward science, and 5–27% of students' academic achievements in science subjects. Besides improving students' understanding of theories and principles, laboratory teaching can promote engineering awareness [10,11], improve their ability to diagnose and correct unacceptable process performance [12], and perform economic evaluations [13].

Several types of laboratory experiences are active in engineering education today. Demonstration experiments, as a first-level laboratory experience, provide students the opportunity to observe particular phenomena in a classroom setting as a means to link theory with practice. A second-level laboratory experience can be derived from the analysis of data collected during an in-class demonstration. Third-level laboratory experiences include experiments carried out by the students under close supervision, while

fourth-level experiences may include open-ended labwork. Strongly-guided labwork is beneficial in developing students' experimental skills, while open-ended labwork, where students have autonomy in making experimental protocol decisions, not only enhance experimental skills, but also assist students in developing independent scientific thinking.

In recent years, a great deal of effort has focused on the integration of new technologies such as multimedia video, audio, and animation, and computers, with associated software, into the laboratory experience. These new technologies are used for data collection [14], model building and interactive demonstration analysis [15,16], and graphical representation of data [17], and for combinations of these technologies [18]. A comprehensive survey on the current practice in laboratory education by Sere et al. [18] suggests that use of new technologies for model building during labwork stimulates students to think more about the conceptual background of a specific lab situation than most other contexts of labwork. Recent advances in the use of the Internet in educational settings suggest that laboratories conducted through use of the World Wide Web (hereinafter referred to as 'the web') may provide increased opportunities for laboratory experiences [4].

Web-Based Education

Web-based education possesses many advantages for engineering educators over other instructional approaches, including use of individual software packages [17,19–22]. First, web-based instruction presents information in a non-linear style, allowing students to explore new information via browsing and cross-referencing activities. In such a constructivist, web-based environment, students are provided more freedom to develop their own metacognitive strategies based on individual backgrounds and experiences [5]. Second, web-based teaching supports active learning processes emphasized by constructivist theory. In a web-based modeling system, vis-à-vis reading, students possess the ability to interactively adjust the parameters of a system, which can then be evaluated and displayed in real time. Instead of passively accepting information, students take an active role in the learning process, and knowledge acquisition can be achieved in a more interesting and meaningful way. A third feature of web-based education is enhanced understanding through improved visualization. Visual learning is the preferred mode for engineering students [2], and web-based tools that can provide clear, colorful, and interesting images can lead to improved degrees of understanding. A fourth advantage of web-based education is its convenience. Access to the Internet enables education to occur at anyplace, at anytime [21]. Through web-based systems, students are provided the opportunity to study at a location they want, and at a time they like. Creating a learning environment accommodating individual schedules can thus improve the rate of learning, especially for well-prepared, senior or graduate level students [19,23].

Several categories of web-based educational tools have been developed, including web-based instruction systems, intelligent tutoring systems, virtual laboratories, and web-based modeling systems. Web-based instruction systems represent a developing branch of computer-aided instruction (CAI). This type of instruction emphasizes the use of the web for transfer of educational information, and it may be considered as a replacement or a supplement to traditional delivery methods of lectures and textbooks. Example information that may be distributed through this venue includes course instructional material, example problems, figures, questions, and exams. HyperText Markup Language (HTML) and JavaScript often provide the foundation of such systems [24, 25], and, on occasion, database technology is employed [4]. An online homework system developed by Nakavachara [26], which makes use of ColdFusion (Macromedia, San Francisco, California) for database connection, and MATLAB (The MathWorks, Inc., Natick, Massachusetts) as a computational engine, provides a good example of web-based instruction.

Intelligent tutoring systems (ITS) use artificial intelligence techniques to formulate models of an expert's knowledge and that of a student's knowledge, and then intervenes with tutorial advice when differences between the two models become evident [27]. ITS can alter instruction content and rate based on real-time tracking and evaluation of students' needs and knowledge levels, which often need to be fulfilled by employing advanced artificial intelligence techniques. Most web-based ITS utilize short questionnaires or quizzes implemented with HTML forms and common gateway interface (CGI) programs to determine a student's knowledge level [16,27]. One such system is designed to assign students knowledge-level appropriate chemical engineering laboratory sessions based on the record of student performances in prior laboratories and exams [16]. Recently, multimedia, in the form of video and audio, were incorporated into a web based ITS as a means to improve the rate of learning [28].

Virtual laboratories emphasize the creation of an interactive, multi-dimensional visualization of a laboratory environment, with some possessing the ability to remotely control actual instruments [27, 29].

While numerous examples of virtual laboratories exist [3], most virtual laboratory systems use general sever/ client models as a basis for their system architecture, with specific elements of the architecture customized to meet demands of the particular application. Virtual Reality Modeling Language (VRML) is also widely employed [30, 31] in most systems due to the extensive use of advanced graphics and multimedia technology, including three-dimensional animation, sound, and artificial sensory devices. Ko's [14] system may represent a typical example, where a double client-server structure is implemented by using JavaScript and HTML for the graphic interface on the client side, Laboratory Virtual Instrument Engineering Workbench (LabVIEW, National Instruments, Baltimore, Maryland) for local instrument control, inetCAM (Inetcam, Inc., San Diego, California) as a video server, and CGI for communication between the client and web server. Other technologies and components, including Java Input-Output Application Programming Interfaces (APIs), Java Applet, Java Database Connectivity (JDBC), Java Servlet, and Microsoft Access database (Microsoft Corporation, Redmond, Washington), are also used in virtual laboratories [29].

Web-based modeling systems can be useful tools to assist teaching of high-level, complex engineering principles in a stimulating manner [21]. Like ITS and virtual laboratories, these systems typically use interactive interfaces to guide students through the learning process, but as important, they also possess great computational capacity. With this added computational element, these systems often possess the potential of expanding from the role of an educational tool to that of a research tool. Example applications of web-based modeling systems include simulating livestock grazing effects on pastures [32], teaching modules for internal combustion engines to edify fundamental thermodynamic and heat transfer concepts [33], simulation engines for custom project management education [34] and fluid mechanics and aerodynamics [35], and a gas turbine simulator that provides an interactive graphical environment for rapid and efficient analysis of user-defined gas turbine systems [36].

Combining high-level computation capacity with web-based user-friendly features, however, greatly increases the technical challenges of employing such systems across the Internet. Some of these challenges arise from a need to interface several aspects of the overall web-based model, including high-level computer languages that drive the numerical models, databases to collect input and output data, graphics and possibly spreadsheet packages to generate model outputs, and interfaces to communicate with users' browsers. Other challenges arise from recognizing that not all web browsers support the same computer languages, and the need for computational efficiency, since many of the models involve complex calculations. In an effort to overcome these challenges, most of the aforementioned example systems [33, 35, 36] employ Java Applet as the dominant technology to construct interactive graphical programs that are easily distributed across the Internet, and are then run on the user's machine using Java [36].

While generally robust in its ability to overcome many of these challenges, Java Applet may be restrictive when examining its long-term applicability to web-based models that are likely to grow considerably more complex, and address an ever-growing variety of problems. For example, many simulation codes already exist in languages such as C/C++ or FORTRAN (FORmula TRANslation). Use of Java Applet requires considerable work to translate the existing numerical code to Java. A second restriction is related to speed of download and computational efficiency. Java Applet often has large initial download overhead, making the appearance of the first page very slow. Moreover, as an interpreted system, Java is typically an order of magnitude less computationally efficient relative to C [37]. A third concern is availability. If not present on their computer, first-time users may be required to download Java. Although free, the download process might be restricted by the familiarity of users, and may be prohibited in computer laboratories and public Internet cafes.

To improve the quality of web-based education, enhancements to the variety of architectures available for web-based modeling are required. These enhancements must be able to address each of the aforementioned challenges, and yet remain flexible to address potential system modifications in the future. The remainder of this paper focuses on the presentation of a novel system architecture that addresses many of the challenges facing web-based models of today. Discussion begins with an overview of the mass transfer processes laboratory employed to illustrate the use of the new architecture.

System Development and Implementation

Mass Transfer Processes Laboratory Overview

Mass transfer processes of environmental systems forms the foundation from which a large number of related disciplines, including water treatment, waste-

water treatment, and surface water and groundwater contaminant transport modeling, derive many of their underlying principles. Such principles include examination of the thermodynamics driving the movement of a chemical from one location to another due to chemical activity gradients. Mass transfer processes of interest include interactions that can occur within the matrices through which the chemicals move. Such interactions may include complex sorption reactions between the chemical and the matrix of interest, and similarly complex mass transfer processes involving advection, dispersion, and interparticle and intraparticle diffusion processes (e.g., Fickian, non-Fickian, Case II transport). This virtual laboratory provides a means to facilitate knowledge development of many of these complex processes by allowing students the opportunity to explore the relative effects of differing sorbent characteristics on the rate of mass transfer. Emphasis is placed on those processes occurring at the particle scale.

As an initial proof of concept for the system architecture emphasized in this paper, this web-based laboratory emphasizes the use of a spherical particle model representative of a number of matrices of interest including soil or sediment particles, or catalysts in a reactor. It is robust in that the model allows the user of the laboratory to define a number of characteristics of a particle, including size, mass, sorption, and diffusion parameters of up to five separate reactive regions within each particle, for up to five different types of particles. The user defines initial conditions, and boundary conditions are related to one of two reactor configurations: completely mixed flow reactor and completely mixed batch reactor. The complex, and often highly non-linear sorption and diffusion processes, are computed by a FORTRAN-based hp-finite element in space and finite difference in time numerical model. The reason for favoring the hp-version of the finite element model is that the non-smoothness of the response can be accounted for easily [38] without the need to resort to upwind-mixed finite element technique to avoid spurious perturbations.

User-friendly, graphical user interface-based input and output modules facilitate information flow between the user and the web-based model. Upon access to the home page of the laboratory, first-time users are invited to submit information including usernames and passwords so that users can remain uniquely identified within the system. Users may then choose to follow an available laboratory tutorial, which includes detailed discussions of each component of the program, as well as example applications. Within the input portion of the program, users are provided the opportunity to define particle characteristics. Users also define the initial and boundary conditions, and then select the format for the model output, including user-specified spatial and temporal increments provided in tabular and graphical form, plus downloadable Excel (Microsoft Corporation) spreadsheets. Standard spatial and temporal output information is also generated for all models, should the user not know what data might be most appropriate for display. The user also has the opportunity of comparing numerical solutions to analytical solutions of the same model (where they exist). Throughout the input and output process, the user may also access an on-screen help system to obtain assistance with any element of the model. Once executed, the model reports the approximate time for job completion, and then notifies the user when the job is complete. Further detailed discussions of the mass transfer processes described by this model, including numerous example applications, are provided in a separate manuscript [39]. Emphasis is here placed on the challenges associated with developing the appropriate architecture to manage user-friendly graphical user interfaces with high-level computer models over web-based systems that are independent of client software.

System Architecture Overview

The Model-View-Controller (MVC) design model serves as the basis for the system architecture of the web-based laboratory. The objective of MVC is to separate the input and output interfaces and the modeling portion of the program into three different components: the ''Model,'' the ''View,'' and the ''Controller.'' In the mass transfer processes laboratory (hereinafter referred to as MTPL) application, the ''Model'' includes several server-resident applications, including the numerical model and a database. Common Object Request Broker Architecture (CORBA) serves as a communication protocol. The ''View'' displays and retrieves information from the user, including interfaces that graphically display model data to the user. The ''Controller,'' through use of the Java Server Pages (JSP) and Java Beans, interprets mouse and keyboard inputs from the user to make appropriate changes to the ''Model'' and ''View.'' The advantage of the MVC paradigm is that it effectively limits and defines the interaction between interface components and underlying problem-domain classes. Relative to the aforementioned web-based model systems, the MTPL system architecture is especially appealing for a number of reasons: (i) it supports various program-

ming languages and also supports the mixing of these languages allowing the use of the most appropriate one for a given task; (ii) it allows for the execution of the numerical engine and the web server on different computers by employing CORBA technology; (iii) it accepts different database management systems by using Open Database Connectivity (ODBC); and (iv) it provides downloadable, well-organized, highly-visual, and easily edited output through use of a popular spreadsheet program (Excel, Microsoft Office Web Components, Version 9.0.0.2710). Figure 1 provides a general illustration of the MVC architecture employed in this laboratory.

Briefly, the MVC architecture of MPTL was developed as follows: the finite-element-and finite-difference-based numerical engine used in MPTL was developed in C++ and FORTRAN for use on the server, while interactive input/output interfaces were constructed with HTML, JavaScript, and JSP. In MPTL, users input data via Internet Explorer (Microsoft Corporation). The web server accepts the data using JSP, which creates Java Beans to hold and manage the information. JSP and the Java Beans handle the transmission and storage of the information in the database, thereby building a simulation request. When the simulation request is complete and valid, users use Internet Explorer to initiate the execution of the simulation request. CORBA functions as a bridge to pass the simulation request from the JSP and Java Beans to the numerical engine to begin the simulation process. The numerical engine obtains the input data and parameters from the simulation request and the database via ODBC. During the simulation, the numerical engine provides program output to the database, again using ODBC. Simulation results, in the form of tables and graphs, are then generated from data retrieved from the database using JSP and Java Beans and displayed through Excel. This information is summarized in a detailed flowchart provided in Figure 2.

Architecture Specifications

A number of software development tools were utilized in the development of this system. This section describes these components in detail, including descriptions of why a particular tool was selected, how it was used, and, in the case of particular software packages, how the software was interfaced with other software.

Web Server. A web server acts as a gateway for users' browsers (e.g., Internet Explorer) to access the laboratory model. A number of web sever software programs are available, including Microsoft Internet Information Server (Microsoft Corporation), Apache web server (The Apache Software Foundation, Wilmington, Delaware), and IBM WebSphere (IBM Corporation, Armonk, New York). Because MTPL employs a JSP environment with Windows 2000 (Microsoft Corporation) system software, the web server must be capable of delivering JSP. In this case, Tomcat (The Apache Software Foundation, Version 3.2.3) was employed as the web server software since, unlike other servers, Tomcat delivers on both JSP and Servlets [40], and is a very good server for developing and testing JavaSever Pages [41].

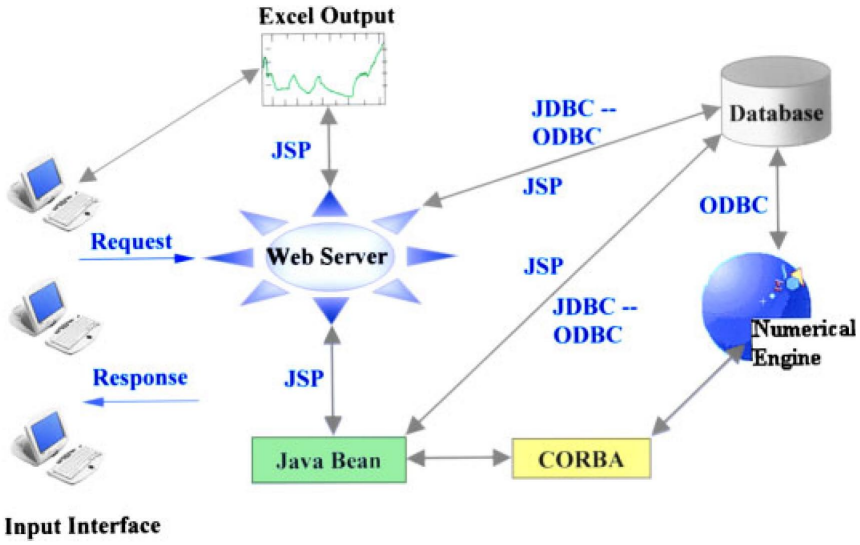


Figure 1. Virtual mass transfer processes laboratory system architecture.

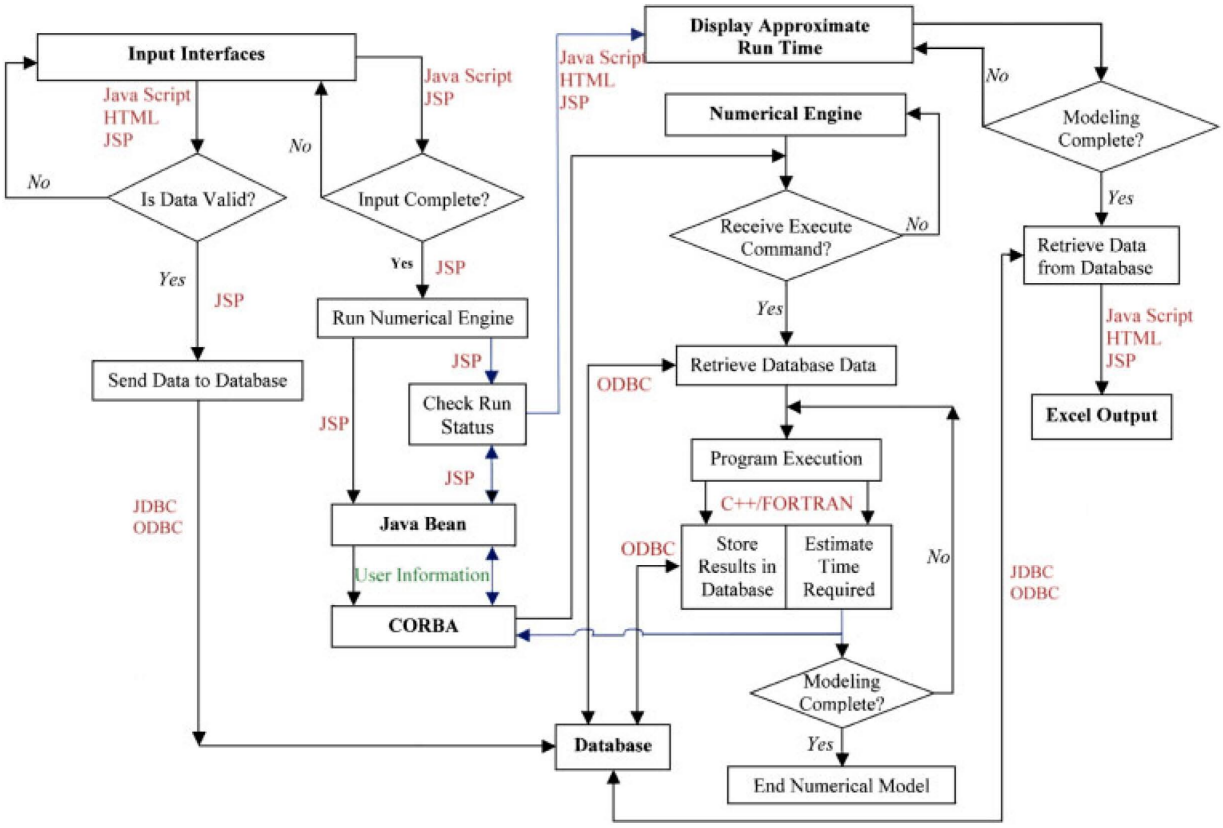


Figure 2. Mass transfer processes laboratory flowchart.

JavaServer Pages (JSP). Server-based scripting languages such as Perl, Active Sever Pages (ASP), PHP (recursive acronym for ''Hypertext Preprocessor''), ColdFusion, and JSP, provide a means to develop dynamic, robust, and easily maintained input and output interfaces. Each scripting language possesses unique features; each with its own particular advantages/ disadvantages [42]. JSP was selected for use in MTPL because it combines the most important features of available alternatives, plus it possesses several unique characteristics. Like other scripting languages, JSP can be embedded with HTML, and allows the development of customized tag libraries. JSP is different in that it provides access to Java APIs, allowing for easy integration of JSP with existing Java files. JSP is also generally more efficient than other scripting languages [41]. For example, JSP pages are compiled and loaded only once, while ASP, Perl, and PHP require reinterpretation for each new client request. In MTPL, JSP provides dynamic input/output interfaces, and transfers data between input/output interfaces and Java Beans, and between input/output interfaces and the database.

Java Beans. Java Beans are reusable Java components that can be recognized and manipulated within visual application builder environments. Value beans and utility beans are two primary types of Java Beans [41] in a JSP-based application. Value beans carry information, while utility beans perform actions such as retrieving data from a database. In MTPL, value beans are used to encapsulate information, such as user ID, reactor type, and particle properties. Utility beans are employed to transmit and retrieve data to/ from the database, transmit an execution message to the numerical model, and query the status of the numerical simulation.

Database. Web-based modeling systems are typically supported by relational databases capable of storing structured data required by the program [43]. A relational database places all information in tables, links tables through shared attributes, and controls redundancy to maintain logical relationships. Several database programs exist, including Structured Query Language (SQL) Server (Microsoft Corporation), mySQL (MySQL AB Company, West Edmonds, Washington), Access (Microsoft Corporation), and Oracle (Oracle Corporation, Redwood Shores, California). While each database program provides relatively similar relational database functions and capabilities, SQL Server 2000 (Microsoft

Corporation), was selected for use in MTPL based primarily on its availability.

Database characteristics of interest include data storage limits, simplicity of programming for customizable solutions, ease of integration into web-based applications, and query performance. For example, initial evaluations of the overall performance of MTPL indicated that an unreasonable length of time was required for presentation of modeling results. Improvements in the logical design through optimized queries, and installation of powerful database indexes for frequently used queries, provided significantly increased performance.

Numerical Engine. Because of its robust compiler designed for high-performance technical computing, the availability of useful subroutines from numerous library sources, and its popularity among many engineering applications, FORTRAN compilers are still commonly employed in solving complex problems requiring extensive computations (e.g., Netlib [44]. Previous uses of FORTRAN-based numerical codes in web-based modeling systems, however, appear to be limited, since the FORTRAN language lacks sufficient support for directly connecting to databases [43] and is not well suited to handling input and output through web servers. These problems are easily overcome, however, by creating a MTPL C++ shell for the FORTRAN program core. C++ possesses the ability to easily communicate with the database for data input/output.

Mixed FORTRAN/C++ language development is relatively simple to employ when the same versions of Visual C++ and Visual FORTRAN are used. In MTPL, Compaq Visual FORTRAN Professional Edition 6.6.0 (Compaq Computer Corporation, Tallahassee, Florida) and Microsoft Visual 6.0 C++ (Microsoft Corporation) were employed to create the numerical model. Editing, debugging, linking, and compiling both software packages are accomplished transparently within Microsoft Visual Studio (Microsoft Corporation). Special consideration, however, must be given for calling, naming, and argument passing conventions between FORTRAN and C++. In addition, if different versions of FORTRAN and C++ are used, FORTRAN code must be complied separately, and the compiled file is required to be added to the C++ program.

Database–Numerical Application Connection: ODBC. To provide an interface between applications and an underlying database, APIs are required. Microsoft SQL Server 2000 supports several classes of APIs for C++ programs: Object Linking and Embedding (OLE DB), ODBC, Embedded SQL for C, and DB-Library for C. Embedded SQL for C and DB-Library are two relatively old APIs, and no future versions of SQL Server will include the files required to perform programming on applications that use these APIs. ODBC is a platform-and database-neutral standard designed to access relational SQL data. A striking advantage of ODBC is its flexibility which makes it possible for users to replace one database program with another without extensive coding [43]. For example, use of ODBC makes it very easy to replace SQL Server 2000 with any other database management system (DBMS) such as Microsoft Access or Oracle. OLE DB, which has evolved from ODBC, can access both relational and non-relational information. OLE DB is creating a better known presence [45]; however, ODBC is still considered the more established interface. As such, ODBC is recommended for employment in web-based modeling systems, as utilized in this system.

ODBC includes an ODBC administrator and a list of database drivers including SQL Server driver. In MTPL, the C++/FORTRAN application calls ODBC functions to communicate with the ODBC SQL Server driver, and submit SQL statements. The SQL Server driver then translates the application's SQL statements into commands that the SQL Server understands, passes SQL statements to the data source, and returns results to the application.

Database–JSP Connection: JDBC. JDBC provides connectivity between a Java program and a database. JSP, which uses Java as its underlying language, requires use of JDBC to communicate with a database. Similar to ODBC, JDBC relies on drivers, rewritten for each specific DBMS, to convert the JDBC functions to the corresponding DBMS. JDBC can be used directly in JSP pages, but this often leads to excessively large programming code. A better approach is to develop a set of custom action elements based on JDBC. In MTPL, custom actions developed by Bergsten [41] are used to achieve JSP and database connectivity.

Common Object Request Broker (ORB) Architecture. CORBA is a platform-independent, language-independent architecture that allows applications to communicate with one another no matter where they are located. CORBA technology is employed in MTPL to enable communication between JSP and the numerical engine. Figure 3 illustrates the flow of information using CORBA technology in MTPL.

In MTPL, users issue requests to the numerical engine via web pages employing Java Beans. ORB, the core

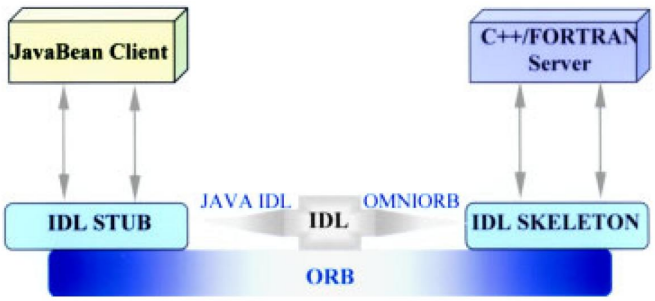


Figure 3. Example CORBA architecture.

layer of CORBA, serves as a client-server bridge, routing requests from the client (i.e., applications that send messages requesting services) to the server (*i.e.,* applications that actually implement the requesting messages), passing model parameter information, and returning the results. To allow both client and server access to ORB, an interface must be established through an Interface Definition Language (IDL). To make the IDL interface understandable to the server-based numerical engine, it is compiled to C++ by OMNI ORB 4.0.0 (AT&T Laboratories, Cambridge, Massachusetts), resulting in files called IDL skeletons. On the client side, Java IDL is used to compile the IDL interface to Java language used in Java Bean, and the resulting files are called IDL stubs. Skeletons and stubs serve as proxies for servers and clients, respectively.

During operation, Java Bean client initiates a client request by calling stubs. ORB receives the call, finds the server, passes the parameters and operation, and calls the server through a skeleton. After the server completes the program execution, ORB returns the results back to the client through stubs. In MTPL, a Naming Service, which associates clients with locations and information, is employed to locate the client and the server by their usernames. By using Naming service, the Java Bean client and the numerical engine server do not necessarily need to be installed on the same computer.

Tutorial and Help Modules

Web-based education tools must be understandable for students to use on a recurring basis [4,20]. Specific challenges arise with web-based modeling in that students must be able to understand the complexities of the modeling system, while maintaining an ability to navigate through the system until the desired model output is obtained. Tutorial and Help modules within most software systems are typically employed to assist in this regard. Tutorial modules are most beneficial in providing a general introduction to the model, as well as step-by-step instructions on its use [46]. Help modules are also employed in most software packages as a means to assist users with a particular problem area or to provide an opportunity to query available functions of the model [4]. This laboratory employs both tutorial and help modules as a means to orient students with the purpose and structure of the laboratory, and to provide opportunities for students to overcome problems while navigating through the system.

The MTPL tutorial provides two primary functions: (1) step-by-step instructions to utilize the full functions of the mass transfer processes laboratory; and (2) illustration of important mass transfer processes phenomena that may be exemplified through use of the laboratory. Written step-by-step instructions are accompanied by visual displays of the web-based environment that the user would see if navigating through the actual program. Through use of an example problem, the tutorial leads the user through input, modeling, and output modules, including detailed instructions on how data is to be entered, and how the user may define the manner in which the output is displayed. To deepen students' understanding of mass transfer processes, many meaningful problems and their solutions are also illustrated. For example, effects of particle size, initial concentration, and boundary conditions are covered in depth as a means to better illustrate the model parameters most responsible for controlling observed mass transfer behavior. Other examples include an evaluation of the effects of sorption linearity versus sorption nonlinearity on the rate of mass uptake, and final equilibrium concentrations. Figure 4 provides a representative portion of a tutorial section, demonstrating the effects of particle heterogeneity on overall diffusion processes within the particle, and the reactor system as a whole.

The Help module includes two components: background information and an on-screen assistant. The background information section, which consists of general project objectives, presentation of the conceptual model, methods of computation, and an introduction to basic mass transfer principles, provides students an overall framework of the MTPL. The on-screen assistant offers instant help on various functions of the laboratory. Upon request for assistance, the on-screen assistant displays an image of the corresponding page that initiated the query. Clicking on the item of confusion or query provides detailed information of that particular program component or function. The main Help page is represented by a frame,

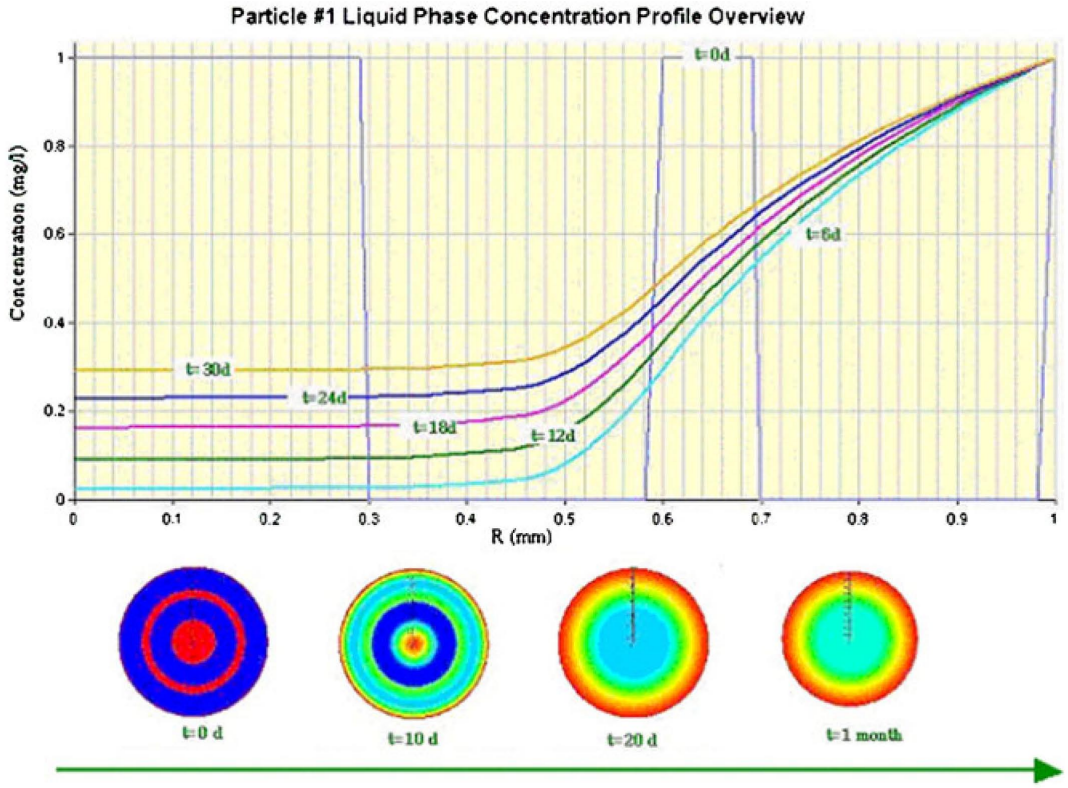


Figure 4. Example tutorial section.

with contents indices in the left column and detailed descriptions in the right column. Example frames of the main Help page are provided in Figure 5.

Tutorial and help modules are incorporated in the system by hyperlinks, hot buttons, and graphical icons. Students can also enter the Help and Tutorial modules from the Welcome page. As an anchor in the main help page, the Tutorial can also be viewed as HTML files or Portable Document Format (Adobe Systems Incorporated, San Jose, California) files. The Tutorial and Help modules are further connected to other program pages through numerous cross-references. While cross-referencing to the corresponding pages, users can create their own constructivist-learning environment based on individual preferences and experiences.

Interactive Input and Output Interfaces

Input and output interfaces are the windows of web-based models, and thus serve as the primary means to achieve the educational objectives of a laboratory employing them. User-friendly, interactive interfaces should be able to attract users' attention, and guide them into use of the model in an efficient manner. Output from such models must be understandable, and should enable manipulation so as to produce tabular and graphical outputs that are most useful to communicating the objectives of the laboratory, as well as satisfying users' needs for individual modeling efforts.

Input Interface

HTML is used for building the static frame of each interface page, while JavaScripts and JSP are embedded into HTML for dynamic contents. Running at the client site, JavaScript helps create truly interactive web pages through programming, and it is also used to validate user input before submission to the web server. Running on the server, JSP is used to develop dynamic web pages and build the bridge between web pages and the database in real time. Examples of input interface are presented in Figure 6.

There are two distinguishing features of the MTPL input system. First is the logical organization. The input interfaces are logically organized like a wizard, leading users in a step-by-step manner to build models. A wizard style interface is intuitive and very easy to use, which effectively avoids overwhelming users with too much information on each page [4]. The second feature is interactivity. As stated before, with JDBC technology, the input interface can communicate

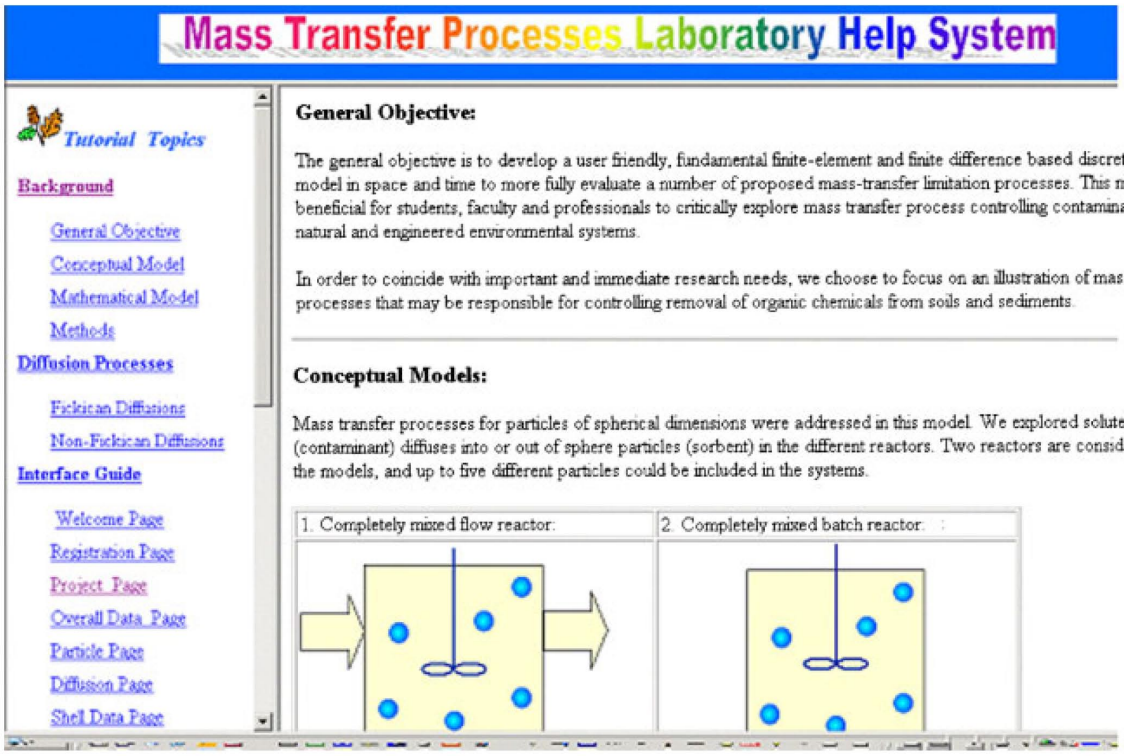


Figure 5. Main help page.

Figure 6. Input interface example pages.

with a relational database in real time. For example, data input into the interface will be transmitted to a database immediately. In addition, whenever users log into their accounts, the data of their latest projects can be displayed. This automatic data retrieval mechanism is convenient in that users do not need to remember, copy, or store the models they have previously submitted. They can work whenever they want, wherever they have access to the Internet.

Output Interface

Knowledge acquisition is easier to achieve if information is stored in memory in an organized and meaningful way [5]. There is little doubt that users may be confused when faced with the large quantities of data produced by the numerical model employed in MTPL. The output module is thus constructed so as to selectively edit and organize the available model output data in an orderly and meaningfully way, while providing enough detail to ensure communication of the essential results of the model run.

The basic structure of the output interface is the same as the input interface, where HTML comprises the main static structure of the output interface, while JSP and JavaScripts provide dynamic contents. The most striking characteristic of MTPL output is the use of Excel. There are two important reasons that Excel is used to present MTPL output results. First, Excel is a powerful, yet easy to understand spreadsheet program for organizing, editing, and plotting data. The second reason is that Excel is the most widely used spreadsheet tool in use today [47], ensuring that a large number of users already possess familiarity with its use. As part of Microsoft Office Web Components, Excel combines seamlessly with other content in the web page. In order to dynamically generate customized Excel tables, lines, or graphs based on user requests, JDBC is used to retrieve the necessary data from the database and JSP code is used to generate the Extensible Markup Language (XML) strings which define the Excel data.

Through use of Excel, MTPL results can be displayed as tables, lines, radar graphs, or pie graphs. Choosing a data table as an output format, users can obtain very detailed numerical results. The data may also be copied, pasted, or directly exported from the on-line Excel table to local Excel files by simply clicking on a single button. While graphical outputs such as line, radar, or pie graphs do not provide specific details like tables, they possess outstanding visualization effects. For example, radar graphs may be used to illustrate the chemical concentration profile in the particle, where concentration data are translated into color codes. Figure 7 illustrates examples of table, line, and radar graph output formats.

In terms of user-friendly flexibility, the MTPL output module provides a general summary of modeling results, detailed results for concentration, mass uptake, and mass flux, as well as specific data sets based on user-defined requests. For example, users may request specific chemical concentration information at any spatial location in the particle at more than one hundred time points by interactively changing data collection locations and times. The MTPL output module thus successfully achieves the objectives

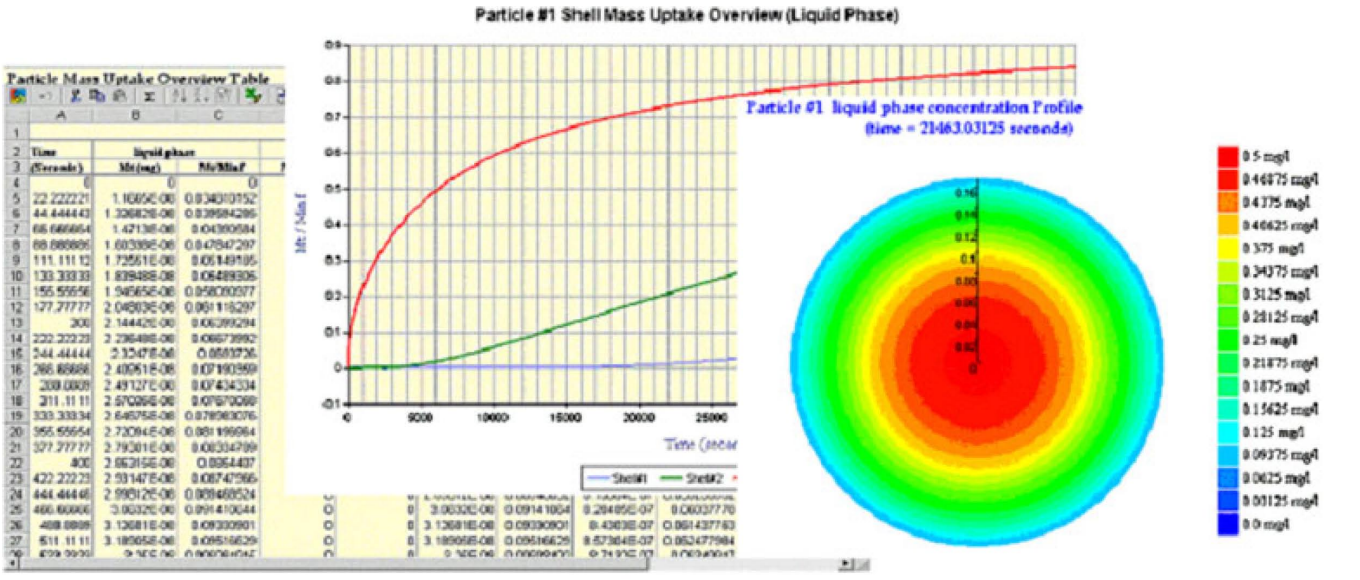


Figure 7. Output interface example pages.

of providing sufficient information, without overwhelming the user with excessive data, while leading users to greater understanding of modeling results, and thus improved knowledge of mass transfer processes in general.

Conclusions

Engineering educators who desire to employ web-based education face numerous challenges. First, educators must understand if, how, and when web-based education may be best employed in their courses to maximize the learning experience. Once a decision is made to employ such a system, educators are faced with the challenge of developing the system to fit the needs of their course. If the educator does not possess the necessary skills for web-based system development, use of outside assistance may be necessary. For more complex web-based education systems such as web-based modeling that employ higher-level complex problems, however, personnel comprising the external assistance may not possess the necessary skills. This scenario greatly increases the difficulties of developing a full capability, high-level web-based modeling system, and may partially explain why development of sophisticated web-based models is still in an emerging phase.

This paper has attempted to identify the specific challenges facing employment of web-based models for education, and has presented novel system architecture to overcome many of these challenges. Learning theories and the use of laboratories in engineering education were utilized as a background to argue for the further development of web-based education tools. Limitations in the use of current web-based models were highlighted in terms of the need to develop more robust system architectures to address the complexity of sophisticated numerical models. A web-based model of mass transfer processes of environmental systems provided an example application of a novel system architecture that overcomes many of the limitations of current web-based models. Presentation of the technical elements of the architecture provided a means to communicate advantages and disadvantages of employing various program interfaces for future web-based model system developers. Specific details of input and output modules, databases for management of data, links to high-level numerical engine, graphics, and spreadsheet packages for displaying output in user-defined formats, and help and tutorial systems, were also discussed to exemplify the user-friendly capabilities of the model.

The mass transfer processes system illustrated in this model focuses on understanding fundamental sorption and diffusion processes for spherical particles in various reactor configurations. Although comprehensive in its coverage of different diffusion and sorption models that may be applicable in most environmental systems, additional phases of work are scheduled to expand the model capabilities for wider applicability. Future work includes:

- Improvement of the numerical engine, to include other standard particle shapes (slabs and cylinders) and user-defined shapes (e.g., polygons), parameter optimization, stochastic simulation, and expansion to multiscale modeling.
- Enhancement of the graphical user interface, to include addition of 3-D animation for result visualization.
- Development of improved feedback mechanisms to provide prompt, two-way communication between the users and the program developers.
- Increase multi-client capacity through use of parallel programming, and use of high-performance distributed computing systems.

Acknowledgments

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. W. J. Weber and F. A. DiGiano, *Process dynamics in environmental systems,* John Wiley & Sons Inc., New York, 1996.
2. P. C. Wankat and F. S. Oreovicz, *Teaching engineering,* McGraw-Hill, Inc., 1993.
3. P. J. Mosterman, M. A. M. Dorlandt, J. O. Campbell, C. Burow, R. Bouw, A. J. Brodersen, and J. R. Bourne, Virtual engineering laboratories: Design and experiments, *J Eng Educ* **85**(3) (1994), 279–286.
4. K. C. Chu, The development of a web-based teaching system for engineering education, *Eng Sci Educ J* **8**(3) (1999), 115–118.
5. T. L. Good and J. E. Brophy, *Educational psychology: A realistic approach,* 4th ed., Longman, New York, 1990, p. 35.
6. J. E. Gillett, Chemical engineering education in the next century, *Chem Eng Technol* **24**(6) (2001), 561–570.
7. S. Kolari and C. Savander-Ranne, Will the application of constructivism bring a solution to today's problems of engineering education?, *Global J Eng Educ* **4**(3) (2000), 275–280.

8. M. Wang, J. Laffey, and M. J. Poole, The construction of shared knowledge in an Internet-based shared environment for expeditions (iExpeditions), *Int J Educ Technol* **2**(2) (2001); http://www.outreach.uiuc.edu/ijet/v2n2/v2n2-feature.html

9. Y. Su and I. T. Huang, The relationship between laboratory climate and students' attitudes toward science, *Chin J Sci Educ* **7**(4) (1999), 393–410.

10. R. England and R. Field, Using the laboratory to develop engineering awareness, *Chem Eng Educ* **23**(3) (1989), 144–148.

11. A. M. Abu-Khalaf, Getting the most out of a laboratory course, *Chem Eng Educ* **32**(3) (1998), 184–189.

12. K. J. Myers, Troubleshooting in the unit operations laboratory, *Chem Eng Educ* **28**(2) (1994), 120–132.

13. T. A. Langrish and W. Davies, Putting commercial relevance into the unit operations laboratory, *Chem Eng Educ* **28**(1) (1995), 40–46.

14. C. C. Ko, B. M. Chen, S. H. Chen, V. Ramakrishnan, R. Chen, S. Y. Hu, and Y. Zhuang, A large-scale web-based virtual oscilloscope laboratory experiment, *Eng Sci Educ J* **9**(2) (2000), 69–76.

15. D. Shin and E. S. Yoon, Web-based, interactive virtual laboratory system for unit operations and process systems engineering education, *Comput Chem Eng* **24**(5) (2000), 1381–1385.

16. D. Shin, E. S. Yoon, K. Y. Lee, and E. S. Lee, A web-based, interactive virtual laboratory system for unit operations and process systems engineering education: Issues, design, and implementation, *Comput Chem Eng* **26**(2) (2002), 319–330.

17. M. F. Iskander, Technology-based electromagnetic education, *IEEE Trans Microw Theor Tech* **50**(3) (2002), 1015–1020.

18. M. G. Sere, J. Leach, H. Niedderer, D. Psillos, A. Tiberghien, and M. Vicentini, *Improving science education: Issues and research on innovative empirical and computer-based approaches to labwork in Europe 1998, 121.*

19. P. Brusilovsky, J. Eklund, and E. Schwarz, Web-based education for all: A tool for development adaptive courseware, *Comput Netw ISDN Sys* **30**(2) (1998), 291–300.

20. P. A. Kirschner and F. Paas, Web-enhanced higher education: A tower of Babel, *Comput Hum Behav* **17**(4) (2001), 347–353.

21. B. Kerrey and J. Isakson, *The power of the Internet for learning: Moving from promise to practice, Web-based education commission* (2000), 185.

22. K. E. Goeller, Web-based collaborative learning: A perspective on the future, *Comput Netw ISDN Sys* **30**(6) (1998), 634–635.

23. P. F. Whelan, Remote access to continuing engineering education (RACeE), *Eng Sci Educ J* **6**(5) (1997), 205–211.

24. S. W. Crown, Improving visualization skills of engineering graphics students using simple JavaScript web based games, *J Eng Educ* **90**(3) (2001), 347– 355.

25. S. W. Crown, Web-based learning: Enhancing the teaching of engineering graphics, *Int Multimedia Electron J Comput-Enhanced Learning* **1**(2) (1999); http://imej.wfu.edu/articles/1999/2/02/index.asp

26. C. Nakavachara, *Facilitating learning and teaching in engineering education: A problem-based approach* 2001.

27. J. Roschelle, J. Kaput, W. Stroup, and T. M. Kahn, Scaleable integration of educational software: Exploring the promise of component architectures, *J Int Media Educ* **6** (1998), 1–31.

28. M. K. Stern and B. P. Woolf, Adaptive content in an online lecture system, In: *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems,* 2000.

29. W. J. Book, K. Koeppen, and M. Rouse, Virtual access hydraulic experiment for system dynamics and controls education, *Mechatronics* **12**(2) (2002), 261–270.

30. Y. Shin, Virtual reality simulations in Web-based science education, *Comput Appl Eng Educ* **10**(1) (2002), 18–25.

31. Y. Dong and M. Zhu, Web-based VR-form virtual laboratory, *Chem Eng Educ* **36**(2) (2002), 102–107.

32. R. H. Mohtar, T. Zhai, and X. Chen, A world wide web-based grazing simulation model (GRASIM), *Comput Electron Agric* **29**(3) (2000), 243–250.

33. A. Kirkpatrick, A. Lee, and B. Willson, The engine in engineering—development of thermal/fluids web based applications, in *Frontiers in Education Conference,* 1997.

34. A. Marin, A simulation engine for custom project management education, *Int J Proj Manage* **18**(3) (2000), 201–213.

35. H. Higuchi, Multi-level, interactive web-based simulations to teach fluid mechanics and aerodynamics from middle school to college levels, *Int J Eng Educ* (2001); http://www.ijee.dit.ie/OnlinePapers/WebBasedSimulationFiltered.html

36. J. A. Reed and A. A. Afjeh, Developing interactive educational engineering software for the world wide web with Java, *Comput Educ* **30**(3/4) (1998), 183–194.

37. B. Eckel, *Thinking in Java,* 2nd ed., Prentice Hall PTR, 2000.

38. P. K. Basu, G. Hsiao, and M. N. Akgtar, Pointwise performance of finite element method in the case of boundary layer problems, *Simulation* **61**(2) (1993).

39. Y. Li, E. J. LeBoeuf, P. K. Basu, and L. H. T. Turner, Web-based interactive virtual laboratory system for mass transfer processes, *J Comput Civil Eng,* in review.

40. Apache Tomcat; http://java.sun.com/products/jsp/tomcat/

41. H. Bergsten, *JavaServer Pages,* 1st ed., O'Reilly & Associates, Inc., 2001, p. 552.

42. J. Bloomberg, J. Kawski, and P. Treffers, Web page scripting techniques, *Soc Sci Comput Rev* **15**(3) (1997), 336–337.

43. S. Jansons and G. J. Cook, Web-enabled database connectivity: A comparison of programming, scripting, and application-based access, *Information Systems Management,* **19**(1) (2002), 14–22.

44. Netlib Repository at UTK and ORNL; http://www.netlib.org/

45. R. R. Ling, D. C. Yen, and D. C. Chou, From database to web browser: The solutions to data access, *J Comput Info Sys* **41**(2) (2000), 58–63.

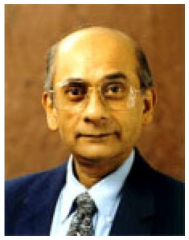
46. S. H. Huang, Q. Su, N. Samant, and I. Khan, Development of a web-based integrated manufacturing laboratory, Comput *Appl Eng Educ* **9**(4) (2001), 228–237.

47. E. Bott and W. Leonhard, *Using Microsoft Office 2000,* Special ed., Ind., Indianapolis, 1999, p. 1.

Biographies

Yusong Li is a PhD student in the Department of Civil and Environmental Engineering at Vanderbilt University. She received her BS degree in environmental engineering from Tsinghua University (P.R. China) in July 1995. From August 1995 to August 1997, she worked as an environmental engineer in Kunming Urban Planning and Design Institute in China. In July 2000, Ms. Li received her MS degree in environmental engineering from Tsinghua University (P.R. China). Her current research is on modeling the fate and transport of pollutants in subsurface environments.

Eugene J. LeBoeuf received his PhD from the University of Michigan in 1998. He has extensive experience in both civil and environmental engineering and military engineering. Prior to joining the faculty at Vanderbilt University, he served several years on active duty with the U.S. Army Corps of Engineers. His research includes evaluation of physical and chemical processes that affect the fate and transport of pollutants in the environment, management and disposal of contaminated sediments, development of GIS-based information management systems, and performance optimization of small-scale wastewater treatment systems. He is a registered Professional Engineer in Missouri and Tennessee, and is a Member of ASCE.



Prodyot K. Basu received his doctoral degree from Washington University, St. Louis. He has extensive teaching and research experience in structural mechanics as well as modeling and simulation of multiphysical systems with application to automotive, aerospace, civil infrastructure, and groundwater flow. He has more than 100 technical publications. He is a Fellow of ASCE and member of ASME, AIAA, and IACM, and he serves on many national committees.

Louis Hampton Turner IV has a PhD in chemical engineering from Rutgers University, and possesses more than 10 years of experience in software development and engineering process modeling. He is the founder of Turner Technology, LLC (Nashville, TN), a software and engineering consulting company, and previously worked for OLI Systems, Inc. (Morris Plains, NJ). His research interests include the development of systems and interfaces that provide a bridge between complex modeling and software technology and the skill sets of less sophisticated computer users. This includes the integration of engineering computer models with World Wide Web interfaces and with Microsoft Excel.