WBCSim: A Prototype Problem Solving Environment for Wood-Based Composites Simulations

A. Goel[1], C. Phanouriou[1], F. A. Kamke[2], C. J. Ribbens[3], C. A. Shaffer[3], and L. T. Watson[4]

[1] *Graduate Research Assistant, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106 USA*

[2] *Professor, Department of Wood Science and Forest Products, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0503 USA*

[3] *Associate Professor, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106 USA*

[4] *Professor, Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106 USA*

Abstract. *This paper describes a computing environment named WBCSim that is intended to increase the productivity of wood scientists conducting research on wood-based composite materials. WBCSim integrates Fortran 77-based simulation codes with a graphical front end, an optimization tool, and a visualization tool. WBCSim serves as a prototype for the design, construction, and evaluation of larger scale problem solving (computing) environments. Several different wood-based composite material simulations are supported. A detailed description of the prototype's software architecture and a typical scenario of use are presented. The system converts output from the simulations to the Virtual Reality Modeling Language (VRML) for visualizing simulation results.*

Keywords. Problem Solving Environment; Computer Model; Wood-based Composite Materials; Visualization

1. Introduction

Scientists and engineers in many application domains commonly use modeling and simulation codes developed in-house that have poor documentation and a poor user interface. Typically only the developers of a code can make effective use of it, and these codes are not typically integrated with tools for visualizing the results. Further, the code is often tied to a particular computing environment. This situation reduces the productivity of many research groups. This paper describes a computing environment named WBCSim that is intended to increase the productivity of wood scientists conducting research on wood-based composite (WBC) materials. WBCSim integrates Fortran 77-based simulation codes with a graphical, Web-based user interface, an optimization tool, and a visualization tool.

The objective of WBCSim is twofold: (1) to increase the productivity of our WBC research group by improving their software environment, and (2) to design and evaluate a specific prototype problem solving (computing) environment (PSE) as a step toward understanding how integrated PSEs should be created. The philosophy of such computing environments, a detailed description of the software architecture of our prototype, and several different wood-based composite material simulations are discussed.

A problem solving environment is a system that provides a complete, usable, and integrated set of high level facilities for solving problems from a prescribed domain [15], [18]. PSEs allow users to define and modify problems, choose solution strategies, interact with and manage appropriate hardware and software resources, visualize and analyze results, and record and coordinate extended problem solving tasks. In complex problem domains, a PSE may provide intelligent and expert assistance in selecting solution strategies, e.g., algorithms, software components, hardware resources, data, etc. Perhaps most significantly, users communicate with a PSE in the language of the problem, not in the language of a particular operating system, programming language, or network protocol. Expert knowledge of the underlying hardware or software is not required. Experience in dealing with large-scale engineering design and analysis problems has indicated the critical need for PSEs with four distinguishing characteristics: (1) facilitate the integration of diverse codes, (2) support human collaboration, (3) support the transparent use of distributed resources, and (4) provide advisory support to the user.

WBCSim is a prototype PSE for making legacy programs, which solve scientific problems in the wood-based composites domain, widely accessible. WBCSim currently provides Internet access to command-line driven simulations developed by the Wood-Based Composites Program at Virginia Polytechnic Institute and State University. WBCSim leverages the accessibility of the Web to make the simulations with legacy code available to scientists and engineers away from their laboratories. The simulation codes used as test cases are written in Fortran 77 and have limited user interaction. All the data communication is done with specially formatted files, which makes the codes difficult to use. WBCSim hides all this behind a server and allows users to graphically supply the input data, remotely execute the simulation, and view the results in both textual and graphical formats.

The remainder of this paper is organized as follows: Section 2 reviews some related work in the field of PSEs. Section 3 gives a detailed description of the three simulation models supported by WBCSim. Section 4 explains the WBCSim user interface. The various software architecture layers of WBCSim and the interaction between them is described in Section 5. To help prospective WBCSim programmers locate files on the WBCSim server, Section 6 explains the file structure of WBCSim. Section 7 describes how WBCSim is typically used. Finally, Section 8 describes the visualization phase of the simulations.

2. Related Work

In the past, problem-specific PSEs have been developed for a wide variety of application domains. Recently, increasing attention is being given to broader issues such as: (1) developing a model or architecture for PSEs; (2) leveraging the Web; (3) supporting distributed, collaborative problem solving; and (4) providing software infrastructure ("middleware") to make PSE-building easier.

One problem domain where PSEs are common is the numerical solution of partial differential equations (PDEs). An early example is ELLPACK [6], a portable Fortran 77 system for solving two and three dimensional elliptic PDEs. Its strengths include a high-level language which allows users to define problems and solution strategies in a natural way (with little coding), and a relatively open architecture which allows expert users to contribute new problem solving modules. ELLPACK's descendents include Interactive ELLPACK [13], which adds a graphical user interface and allows greater user interaction, and Parallel ELLPACK (PELLPACK) [19], which includes a more sophisticated and portable user interface, incorporates a wider array of solvers, and can take advantage of multiprocessing. PELLPACK also includes an expert or "recommender" component named PYTHIA [21], [37]. Another system which provides a high level, problem-oriented environment for PDE-solving is SciNapse [1], a code-generation system that transforms high-level descriptions of PDE problems into customized C or Fortran code, in an effort to eliminate the need for programming by hand. Other PSEs in the PDE problem domain include DEQSOL [35], PDEase2D [38], and PDESOL [33].

PSEs are being built for a number of other scientific domains as well. For example, Johnson et al. [29] describe SCIRun, a PSE that allows users to interactively compose, execute, and control a large-scale computer simulation by visually "steering" a dataflow network model. Bramley et al. [4], [16] have developed Linear System Analyzer, a component-based PSE, for manipulating and solving large-scale sparse linear systems of equations. Dabdub et al. [11] have built a PSE for modeling air pollution in urban areas. The WISE environment [24] lets researchers link models of ecosystems from various subdisciplines.

An important goal of PSE researchers is to define a generic architecture for PSEs and to develop middleware (typically object-oriented) to facilitate the construction and tailoring of problem-specific PSEs [15]. This emphasis, along with work in Web-based, distributed, and collaborative PSEs, characterizes much of the current research in PSEs. For example, in [8] the authors describe PDELab, a multilayered, object-oriented framework for creating high-level PSEs. PDELab supports PDESpec, a PDE specification language that allows users to specify a PDE problem in terms of PDE objects and the relationships and interactions between them. Parallel Application WorkSpace (PAWS) [28] is a CORBA-based, object-oriented server for connecting parallel programs and objects. Other researchers investigating object-oriented frameworks for PSE-building include Gannon et al. [16], Balay et al. [3], and Long and Van Straalen [25].

With the rise of the Web, PSEs are now beginning to support distributed problem-solving and collaboration. Regli [31] describes Internet-enabled computer-aided design systems for engineering applications. Net PELLPACK [26], PELLPACK's Web-based counterpart, lets users solve PDE problems via Java applets. Other Web-based PSEs include NetSolve [7] and NEOS [10]. Current PSE-related research projects that emphasize distributed collaboration include LabSpace [12], the Intelligent Synthesis Environment (ISE) [17], Habanero [9], Tango [5], Symphony [34], and Sieve [20].

3. Simulation Models

WBCSim contains three simulation models of interest to scientists studying wood-based composite materials manufacturing. Each of these models is described briefly.

3.1. Rotary Dryer Simulation (RDS)

The rotary dryer simulation model [22], [23] was developed as a tool to assist in the design of drying systems for wood particles, such as used in the manufacture of particleboard and strandboard products. The rotary dryer is used in about 90 percent of these processes. It consists of a large, horizontally oriented, rotating drum (typically 3 to 5 m in diameter and 20 to 30 m in length). The wet wood particles are mixed directly with hot combustion gases at the inlet. The gas flow provides the thermal energy for drying, as well as the medium for pneumatic transport of the particles through the length of the drum. Interior lifting flanges serve to agitate and produce a cascade of particles through the hot gases. This process uses a co-current flow.

The RDS model consists of a series of material and energy balance equations, which are defined for each cascade of wood particles. A cascade cycle begins when a particle drops off a lifting flange and falls to the bottom of the drum. This is followed by travel along the periphery of the drum, when the particle is caught by a lifting flange. The cascade ends when the particle attains its maximum angle of repose and tumbles off of the lifting flange. The heat and mass flows between cascade cycles, and the distance of travel along the length of the drum for each cycle, are determined by algebraic equations. The user must supply the inlet conditions of the hot gases and wet wood particles, as well as the physical dimensions of the drum and lifting flanges, flow rates, and thermal loss factor for the dryer. The RDS model predicts the moisture content and temperature of the wood particles for each cascade in the drum, and predicts the gas phase composition and temperature at each cascade.

3.2. Radio-Frequency Pressing (RFP)

The radio-frequency pressing model [32] was developed to simulate the consolidation of wood veneer into a laminated composite, where the energy needed for cure of the adhesive is supplied by a high-frequency electric field. Radio-frequency pressing is commonly used for thick composites and for laminated composites that are nonplanar. Wood is a dielectric material, where the presence of water (a common constituent of wood) and polar adhesive molecules assist in the absorption of the electric field energy. The model may be used to help design alternative pressing schedules.

The RFP model consists of a collection of nonlinear PDEs that describe the heat and mass transfer within the veneer layers. The primary variables are temperature and moisture content. The moisture content is further divided into three phases: bound water, liquid water, and water vapor. These water phases must satisfy a criterion of local thermodynamic equilibrium as represented by a nonlinear algebraic equation. The model is one-dimensional, with a fixed resistance to heat and mass flux at the boundary. The results of the model include the time-dependent temperature and moisture content profiles in the veneer layers. A time- and temperature-dependent equation also predicts the extent of adhesive cure. The user must supply the initial density, moisture content, and temperature of the veneer, as well as veneer thickness, and the electric field strength.

3.3. Composite Material Analysis (CMA)

The composite material analysis model was developed to assess the strength properties of laminated fiber reinforced materials, such as plywood. The model can perform two tasks. First, the normal and shear stresses together with the strains and curvatures induced by a user-defined deformed shape in the material can be calculated. Second, it can calculate the stresses and strains caused by the combination of different loading conditions such as tension, moment, torque, or shear. The calculations are based on the composite lamination matrix theory (CLMT) and the Hoffmann failure criteria. The model predicts the tensile strength, bending strength, and shear strength of the composite material. The strength calculations are performed iteratively. The load level is increased by a specific increment until all the layers in the composite fail. The load at the point of failure of each lamina, and the induced stresses and strains in the laminate, are recorded.

This simulation was designed to allow easy specification of the type of material, thickness,

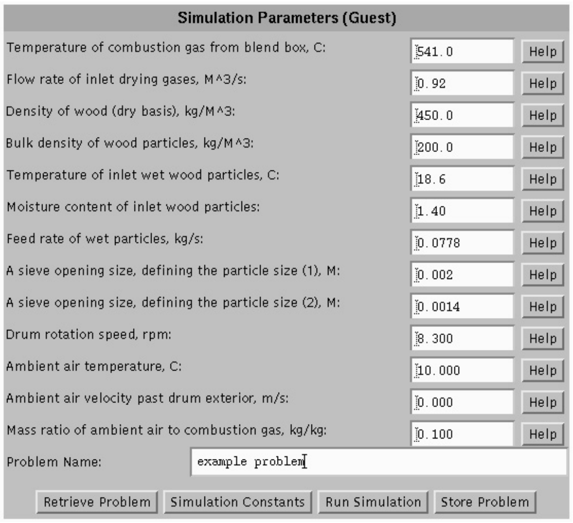

Figure 1. Input screen for RDS simulation.

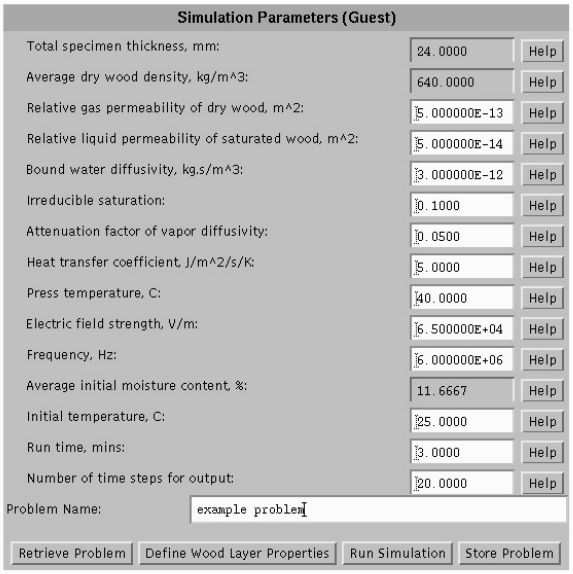

Figure 3. Input screen for RFP simulation.

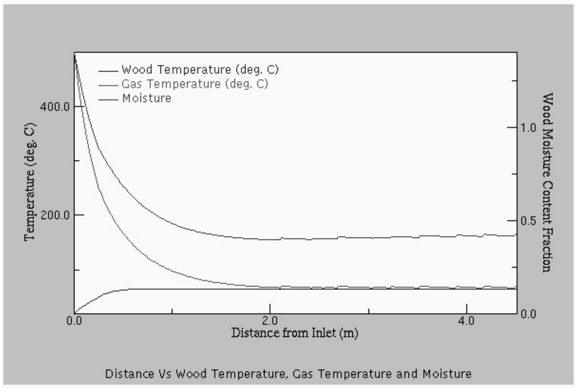

Figure 2. Output graph for RDS simulation.

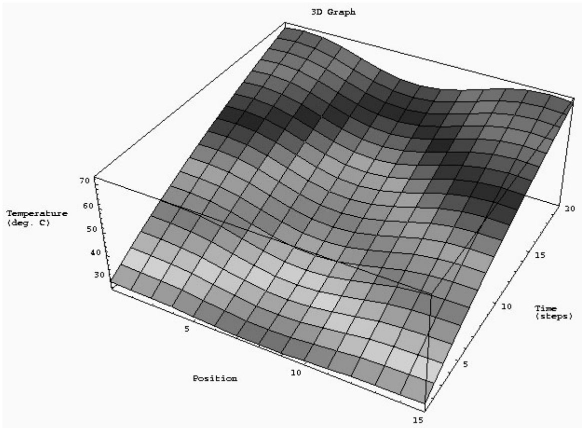

Figure 4. Output graph for RFP simulation.

and orientation of the fibers at each layer of the composite. The mechanical and failure properties of different materials are predefined. The detailed calculations at each step of the iteration process are stored as text files during the solution phase. The resulting stresses and the failure load of each layer are displayed in a three-dimensional model of the laminated composite.

4. WBCSim User Interface

The WBCSim user interface is composed of Java applets. Figure 1 shows the applet that takes input for the RDS simulation. The interface consists of a set of text boxes where the user can enter values for various input parameters. The "Store Problem" button is used to store the current set of input values, which can be retrieved later using the "Retrieve Problem" button. Some simulation parameters are not accessible by the user but may be viewed by clicking on "Simulation Constants". Clicking on "Run Simulation" executes the Fortran code with the input values supplied by the user through the applet, and produces output data in text and graphical forms. For example, Figure 2 shows an output graph for the temperature and moisture content at various distances from the dryer inlet.

The input screen for the RFP simulation is similar to the RDS input screen, as shown in Figure 3. The user can enter values for the parameters through text boxes, and the results are produced in both text and graphical forms. For example, Figure 4 shows an output temperature graph as a function of position through the thickness of the laminated composite and time during processing. The graphic is a fixed-frame three dimensional plot generated by Mathematica.

Figure 5 shows the input screen for the CMA simulation. Each row of input data represents a layer of the wood composite. The user can add layers to the composite by clicking on the leftmost checkbox for that layer. Currently, a composite can have at most ten layers. For each layer, the user can select from a menu of materials, including several wood species and synthetic materials. The material properties are predefined. Thickness and fiber orientation may be specified for each layer. The user can enter values for various forces acting on the composite by referring to the image displayed within the applet. Clicking on "Run Simulation" executes the Fortran code. An example of the output produced is shown in Figure 6.

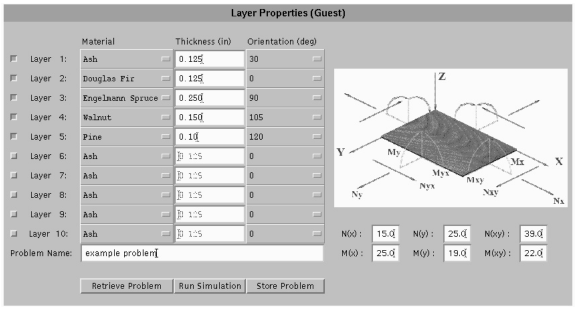

Figure 5. Input screen for CMA simulation.

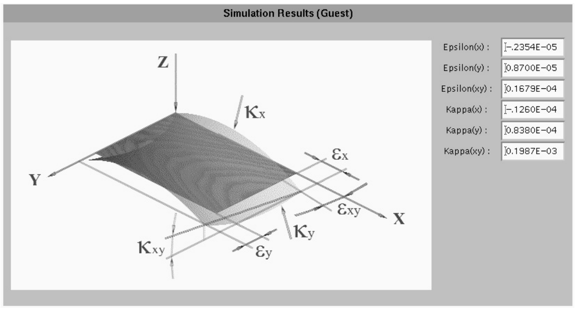

Figure 6. Example output for the CMA simulation.

5. Software Architecture

The software architecture for WBCSim uses a three-tier model. The tiers correspond to (1) the legacy simulations and various visualization and optimization tools, perhaps running on remote computers; (2) the user interface; and (3) the middleware that coordinates requests from the user to the legacy simulations and tools, and the resulting output. These three tiers are referred to as the developer layer, the client layer, and the server layer, respectively, as shown in Figure 7.

WBCSim supports legacy programs written in any programming language. The only restriction is that the program must take input parameters from the command-line, one or more input files, or the standard input stream. In particular, WBCSim supports the three Fortran 77 simulation codes described in Section 3.

5.1. Developer Layer

The developer layer consists primarily of the legacy programs on which WBCSim is based. The server layer expects a program in the developer layer to communicate its data (input and output) in a certain format. Thus, legacy programs are "wrapped" with custom scripts. The scripts are written in Perl, and each legacy program must have its own wrapper script. The script receives input parameters from the server, and converts those parameters as appropriate for that legacy program. The legacy program is executed with these parameters fed to its standard input stream. The program's input may direct it to load appropriate input files. When the legacy program terminates, the wrapper script packages the program's output to create an HTML page, and passes the URL of this page to the server.

The developer layer also includes tools to help developers get more from their simulations. In WBCSim this concept is implemented by integrating the legacy programs with an optimization tool and various visualization tools.

The optimization tool provided with WBCSim is the Design Optimization Tool (DOT) [36]. DOT allows the user to provide ranges, as opposed to fixed values, for the input parameters and get a solution that either maximizes or minimizes a given output value. DOT is a sophisticated engineering optimization subroutine incorporated into WBCSim.

WBCSim examples use two visualization tools: Mathematica [39] and VRML [2]. Mathematica is used to generate static three dimensional graphs of the simulation output. The output is also translated to VRML. With a VRML viewer, the resulting graphs can be viewed from various directions in the three dimensional viewspace. In principle, developers can add custom filters to convert a program's output to a useful form for any viewer of their choice.

The simulations generate text output files containing raw data. The script that wraps the RFP simulation executes Mathematica to generate GIF files, and the VRML translator to generate VRML files. The HTML page generated for the results of the simulation includes links to these files.

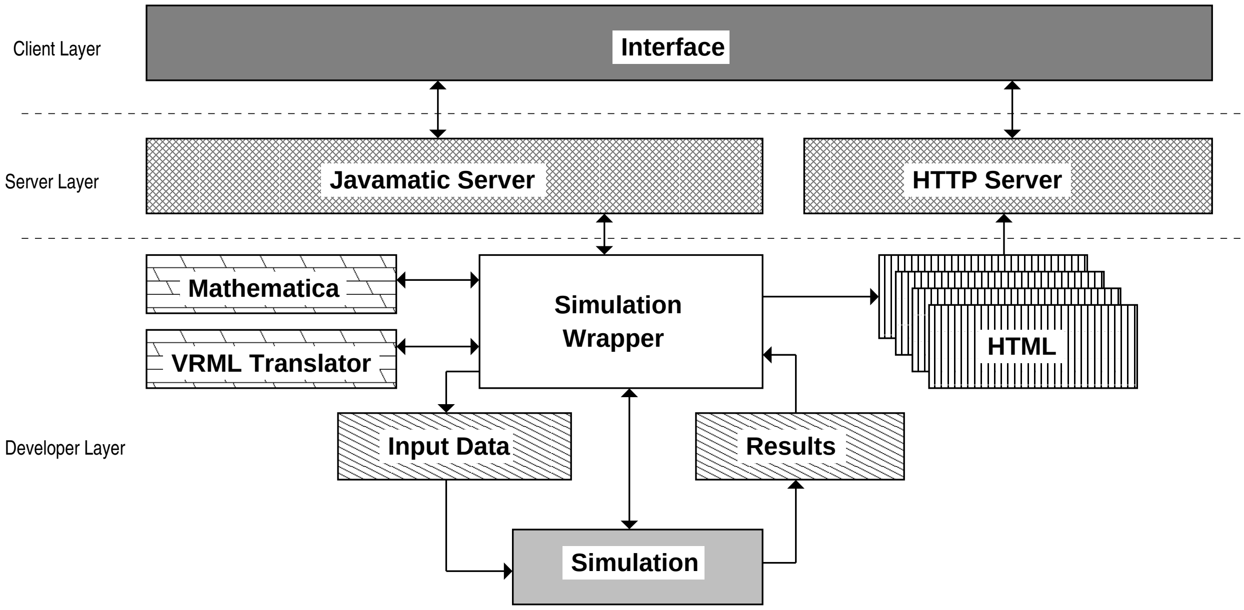

Figure 7. WBCSim architecture overview.

5.2. Client Layer

The client layer is responsible for the user interface. It also handles communication with the server layer. This is the only layer that is visible to end-users, and typically will be the only layer running on the user's local machine.

The client layer consists of the Java applets described in the previous section. After the user enters all the necessary parameters to control execution of the simulation, the client communicates these parameters along with a request to execute the corresponding program to the server layer. The server layer returns the URL for the HTML page generated from the simulation's output. The client layer directs this page to the user's HTML browser.

The client layer also contains viewers for the various visualization tools found in the developer layer. WBCSim requires a VRML 2.0 viewer for the RFP model, a VRML 1.0 viewer for the CMA model, and a 3D visualization Java applet. The user is responsible for installing a VRML viewer on the local machine, but the 3D visualization applet is automatically downloaded from the HTTP server.

5.3. Server Layer

The server layer is the core of WBCSim as a system distinct from its legacy code simulations and associated data viewers. The server layer is responsible for managing execution of the simulations and for communicating with the user interface contained in the client layer. The main part of the server layer is the Javamatic server [30]. The Javamatic server is written in the Java programming language. The Javamatic server can direct execution of multiple simulations and accept multiple requests from clients concurrently. The results from the simulations are communicated to the clients using an HTTP server.

For security reasons, Java applets that have been downloaded from a network are not allowed to read, write, or execute files on the client's (local) file system. Since WBCSim takes input data from users via Java applets, this means that such applets must forward requests to execute the legacy application through the WBCSim server for processing. The server processes these requests and returns the results of the transaction to the client.

WBCSim uses the Java *Socket* class to communicate between the client and the server. Each time a user issues a command through a Java applet (e.g., he/she clicks on the "Run Simulation" button), a new client request is sent to the Javamatic server. The Javamatic server then goes through the following steps:

1. The main thread of the server creates a new thread to service the request and then continues to listen for new requests.
2. This new thread then receives the request from the client.
3. If the request is to execute a simulation, the server then validates the identifier in the request using a dictionary file. The dictionary file

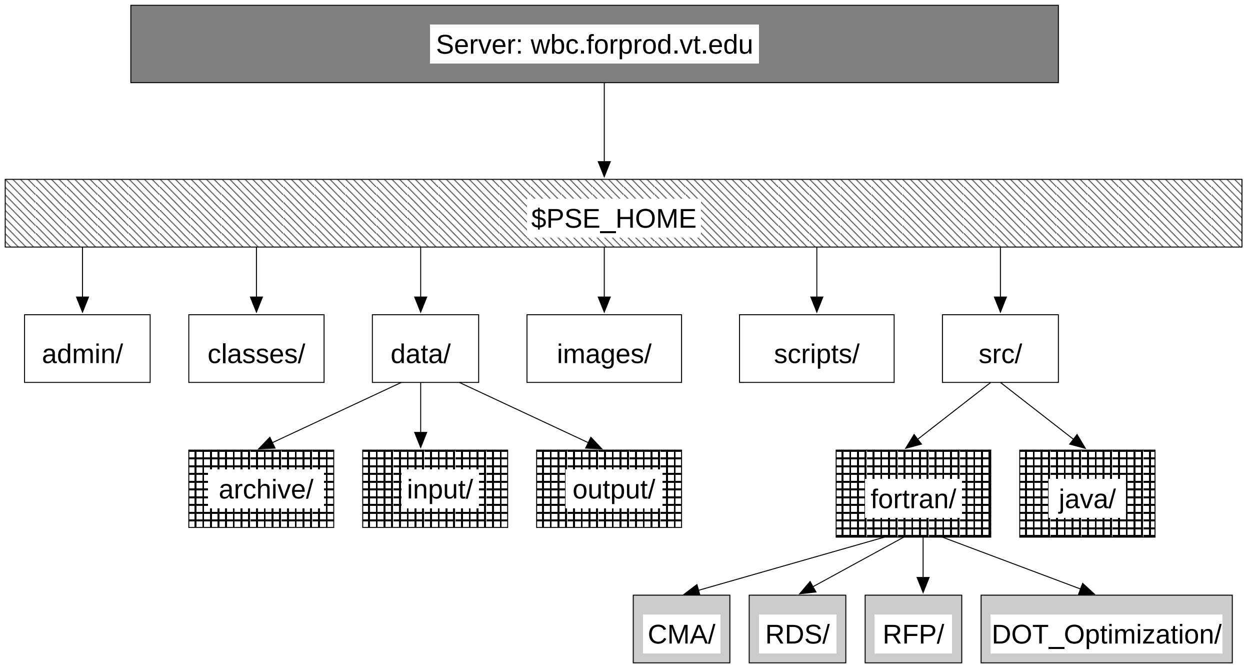

Figure 8. Directory structure of WBCSim.

contains a list of known application identifiers along with the path for the executable file for each. If the validation fails, the server responds with an error message and closes the connection.

4. If the validation succeeds, the server then receives the arguments for the simulation and returns a unique URL. This URL points to the HTML file that will contain the results after the execution.
5. The server creates a new process and takes control of the input, output, and error streams. However, only the error stream is used to provide feedback to the client. The Java Virtual Machine buffers the output stream and the output file(s) are available only after the process terminates. They are not used by WBCSim for real-time feedback.
6. The server then executes the application in the new process. While the process is executing, the current thread sends to the client any information coming from the error stream.
7. The client can request termination of the execution of the simulation, or query the status of the server.
8. When the simulation terminates, the server closes its socket to the client and stops the two threads.
9. The client then contacts the HTTP server (which runs on the same machine as the Javamatic server), and gets the content of the HTML page generated for the simulation's output.

6. Directory Structure of WBCSim

WBCSim simulations can be accessed from

`http://wbc.forprod.vt.edu/pse/`

All source and data files required for running WBC simulations are placed in the WBCSim home directory. The directory structure of WBCSim is shown in Figure 8.

A brief description of the contents of each directory in WBCSim is as follows:

1. `admin/`: contains files required for maintaining the server, i.e., starting the server, setting environment variables prior to starting the server, maintaining a log of client transactions, and the WBCSim dictionary of recognized commands.
2. `classes/`: contains the Java bytecode for WBCSim.
3. `data/archive/`: contains a log of old simulation results that a user may have stored. The log files are stored under a separate subdirectory for each user.
4. `data/input/`: contains the simulation input files generated by a user while performing a simulation. These files are temporary and may be deleted.
5. `data/output/`: contains the simulation output files generated by a user while performing a simulation. All the results have unique names and are stored here temporarily until they are

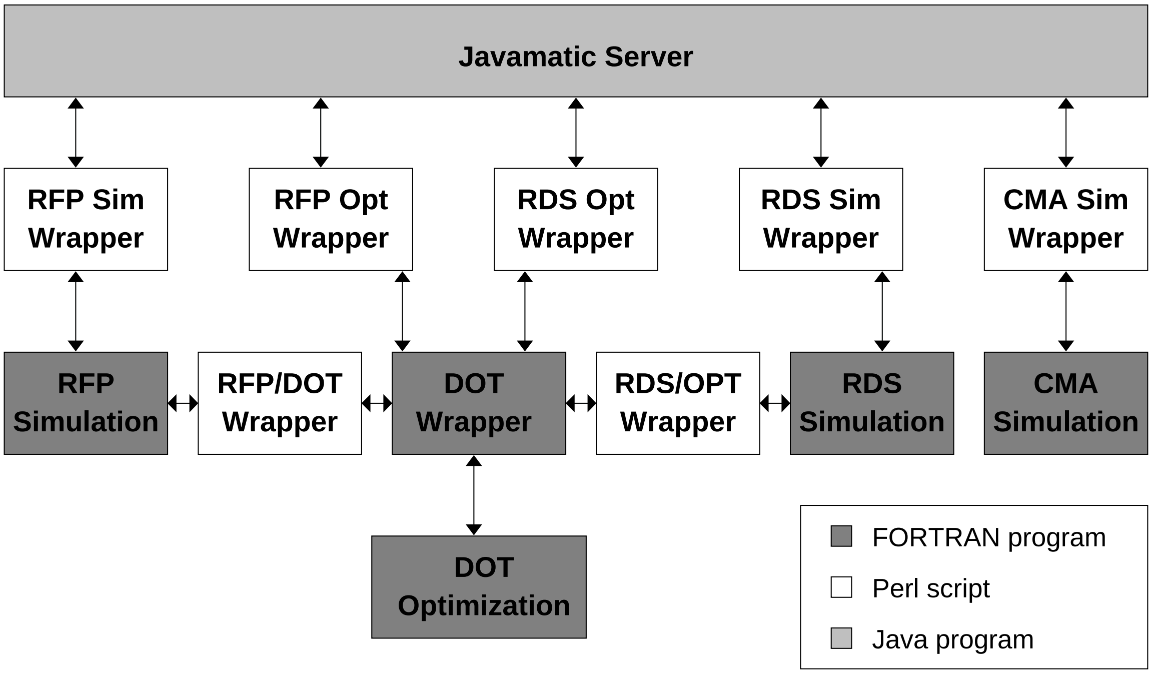

Figure 9. Legacy Fortran 77 simulation code wrappers.

permanently archived. These files may be deleted.

6. `images/`: contains images that are displayed on the WBCSim Web pages. Images generated by simulations are not stored here.
7. `scripts/`: contains Perl scripts for WBCSim.
8. `src/fortran/`: contains Fortran code that runs the simulations.
9. `src/java/`: contains Java source code for WBCSim.

7. Simulation Scenario

WBCSim incorporates the legacy Fortran 77 programs that implement its models without any modifications to the code. This has the advantage that if the programmer decides to make changes in the legacy program, such as bug fixes, recompilation with new libraries or newer compilers, or implementing a different algorithm, the new program can be installed in WBCSim without additional work. Figure 9 shows how this is possible.

Consider the RFP simulation. A Perl script (*RFP Sim Wrapper*) acts as a proxy between the Javamatic server and the simulation. This script is responsible for converting data from the Javamatic server to a format the RFP simulation can recognize.

In a typical scenario, the server will execute the RFP Sim Wrapper and give it all the input parameters from the command line plus an additional argument. The input parameters come directly from the client and they are a sequence of strings derived from the user's selection. The Javamatic server treats all parameters as strings regardless of whether they are boolean, numeric, or alphanumeric. The additional argument is a filename, which is accessible from the HTTP server. The RFP Sim Wrapper does not return the results to the Javamatic server, but outputs them in HTML format to that file. The Javamatic server then returns to the client the URL pointing to that file.

When executed, the RFP Sim Wrapper packages all the input parameters into a file and executes the RFP simulation. The parameters are recognized by position, thus all parameters must be present and have a value even if they are not visible on the interface or the user did not provide a value. The client has a list of default values for all the parameters that are not visible on the interface and so it fills the blanks before sending the parameters.

The RFP simulation has its own text-based user interface and requires input from the standard input stream (e.g., keyboard). The RFP Sim Wrapper takes control of the standard input stream of the RFP simulation and generates the appropriate keystrokes to read the input file and generate the results. It does this by generating a new file with all the appropriate commands and redirecting that file at the command-line when it executes the simulation. This is a temporary file and is deleted when the simulation completes execution.

While the simulation executes, RFP Sim Wrapper takes control of the standard output stream of the simulation and listens for specific string patterns. These string patterns indicate major milestones in the execution of the simulation (i.e., successful completion of a simulation step or the computation of an intermediate value). RFP Sim Wrapper then generates a message and outputs it to the standard error stream. The Javamatic server captures that message and propagates it to the client. Eventually the message gets to the user. The standard error stream is used, instead of the standard output stream, because Java buffers the standard output stream and makes it available only after the process terminates. In contrast, the contents of the standard error stream are sent continuously to the parent of this process. Arbitrary network delays can cause a group of messages to be delivered simultaneously, even though they were generated at different times. The client always displays the latest message and discards any old messages.

When the simulation terminates, the wrapper takes the results and creates a file, in HTML format, with the filename given by the server. This file contains a list of links that point to the results in various formats. The RFP Sim Wrapper then runs Mathematica to read the results, which are in textual format, and generates a 3D Plot, which is a Mathematica internal data structure. RFP Sim Wrapper then calls the appropriate Mathematica function to output the 3D Plot to a file in encapsulated postscript format. It also calls the VRML translator to generate a VRML wireframe representation of the results. RFP Sim Wrapper executes a series of filters to convert the results to GIF images. RFP Sim Wrapper also creates text and HTML-table versions of the results. All these different views are stored in individual files on the server. An archive tool manages the files from different simulation runs.

A benefit of using the simulation code in a PSE environment is that other software tools, such as the DOT optimizer, can be used to provide more functionality to the end-user. The user can provide a range instead of a fixed value for any of the input parameters and specify which component in the output to maximize or minimize. RFP Opt Wrapper, a Perl script, receives the input parameters from the server plus the variables to optimize and a filename. The wrapper then packages all the input parameters in a file and executes DOT Wrapper, which is a Fortran 90 program. RFP Opt Wrapper gives DOT Wrapper the filename that contains the data plus the name of the program to call every time the optimizer asks for an evaluation of the objective function. RFP/DOT Wrapper executes the RFP simulation once and returns the objective function value. When the optimizer finishes, DOT Wrapper returns the optimal parameter values and objective function value to RFP Opt Wrapper. RFP Opt Wrapper then executes the RFP simulation one more time and packages the results in the same format that RFP Sim Wrapper uses. The advantage is that neither the simulation code nor the optimization code needs to be changed. To add a new optimizer with a different optimization algorithm, the only thing that needs to be done is to modify the wrapper script DOT Wrapper. The simulation code and RFP/DOT Wrapper remain the same.

8. Visualization

VRML was chosen as the primary viewing environment for simulation output. Our approach is to convert output generated by Fortran into a VRML description so that the output of the simulations can be visualized interactively. A custom-built translator provides greater control over the description of 3D models that third-party translators cannot provide.

VRML was chosen for the following reasons:

1. VRML is a recognized standard for visualizing 3D worlds.
2. VRML viewers are available for a wide variety of platforms, and most of them are easy to use.
3. VRML syntax is simple, and VRML code can be easily generated by programs [27].

VRML models were generated for the radio-frequency pressing model and the composite material analysis model.

Radio-frequency pressing model: The output data generated by RFP is a matrix of numbers where each number represents the value of a dependent variable with respect to two independent variables. Since this is a two-dimensional data matrix, it is conveniently modeled as a VRML `ElevationGrid` as shown in Figure 10. The dependent variable here is "Pressure" (y axis), which is now represented by variations in the height of the elevation grid, and the independent variables are "Position" (x axis) and "Time" (z axis). Each element in the matrix becomes a colored point on the grid, and the color is smoothly varied between the grid points. WBCSim also provides a variant of this model where elements in the matrix are represented by squares rather than points, and a different color is assigned to each square, resulting in a checkered elevation grid.

To ensure that the top and bottom views of the elevation grid are able to indicate the true height of the grid points, each grid point is colored based on its

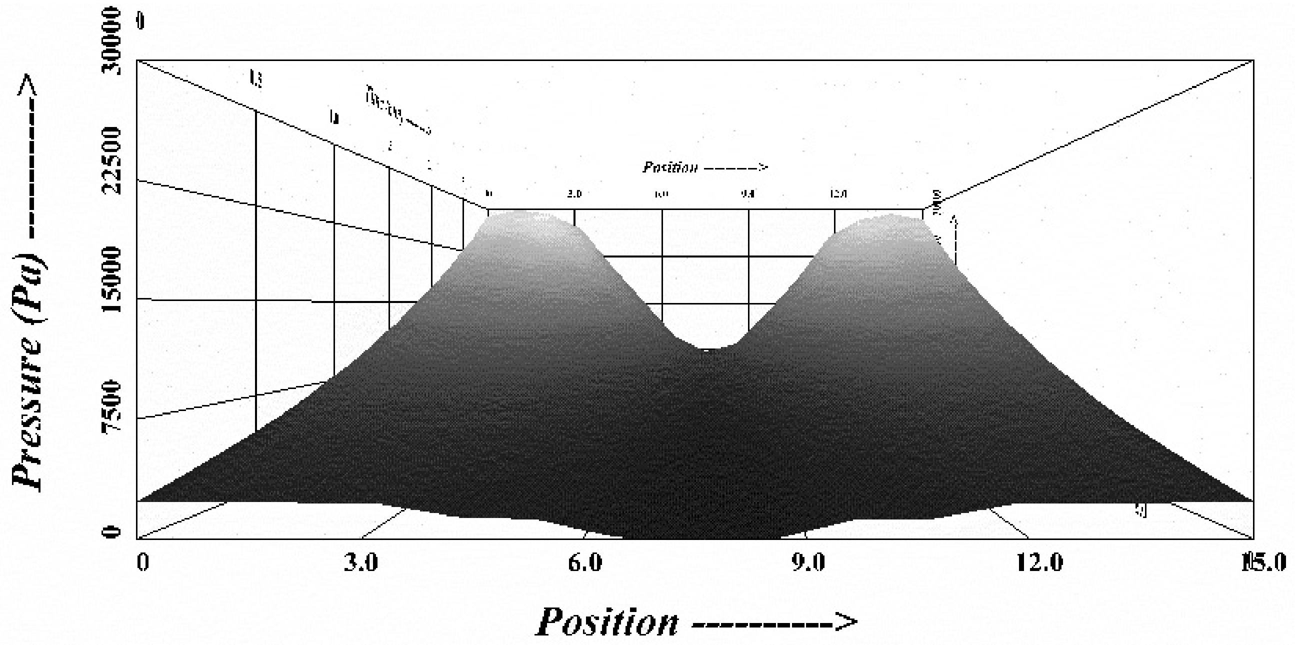

Figure 10. Pressure graph showing the variation of pressure with increasing distance from the laminate surface and with time (the receding axis).

height, with blue representing the lowest point on the grid, and red the highest. All other grid points are colored by linearly interpolating between blue and red color values.

Composite material analysis model: In this simulation, the output contains information about the forces being applied on various layers of a composite when one or more of the layers fail due to an applied load. The composites are tested for strength based on their material property, thickness of layers, and orientation of the fibers in each layer. Material properties may be selected from a menu of various wood species. Five testing methods have been implemented: analysis, design, tensile strength, bending strength, and shear strength. In the analysis method the program calculates the deformed shape of a multi-layered composite laminate (e.g., plywood) caused by user-defined loads and moments. In the design method the program calculates the applicable loads and moments which cause a user-defined deformation of the multi-layered composite. The other three methods calculate the magnitude of the load (tensile, bending, or shear) which causes the multi-layer composite laminate to fail. In the analysis and design methods, one or more layers may or may not fail, whereas in the other three methods, the simulation continues until all the layers fail. It is the job of the VRML translator to decide which layers have failed by looking at the Fortran-generated output files. Each time a layer fails, a new VRML model is generated containing information about the forces on each layer at that instant. Failed layers are displayed in red (gray in the figure), whereas active layers are displayed in black, as shown in Figure 11.

9. Conclusion

By providing a Web-based graphical interface to command-line Fortran applications, WBCSim provides the following benefits:

1. It enables a wider class of users to access these tools away from their workplaces.
2. Since these simulations take input via Java applets, scientists and engineers working from a variety of platforms are able to access them.
3. All processing is done on the server-end, so users of WBCSim need not worry about software installation issues at the client—anybody with a Java-enabled Web browser and an Internet connection can perform these simulations.
4. Output from the simulations is visualized graphically as compared to the earlier method of reading text files generated by Fortran. This has the advantage that the user need not know what subprograms are being invoked, and what data conversions are being done to satisfy a request. In other words, the user can concentrate on the intellectual aspects of the problem solution [14], leaving data manipulation, management, and presentation to WBCSim.

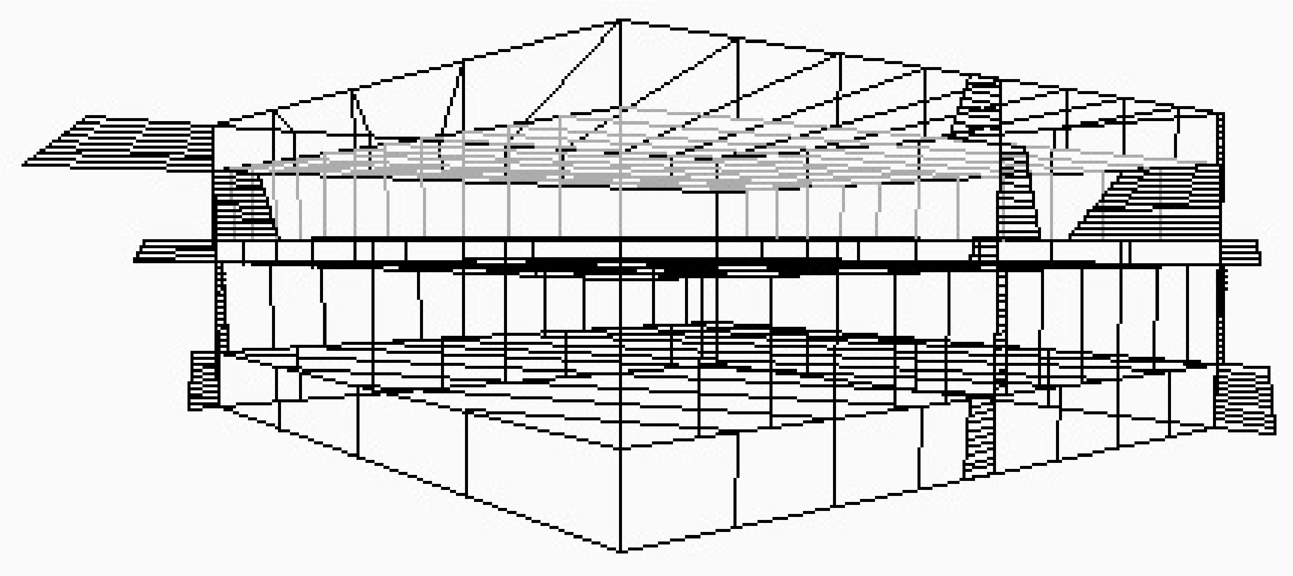

Figure 11. Wireframe model of a composite showing failed layers (gray) and active layers (black), and the orientation of fibers in each layer. In this figure, the second layer has failed. The horizontal protrusions are proportional in length to the magnitude of forces being applied to the layers at the time of failure. The direction of the protrusions is unimportant.

In short, WBCSim provides an integrated set of high-level facilities for solving problems in the wood-based composites domain. It allows users to define, record, and modify problems, and to visualize and analyze simulation results, which is the very essence of a PSE.

The original motivation for creating WBCSim was the more obvious effects of improved usability provided by the graphical user interface and Web-based access. These effects in themselves would be enough to make WBCSim a success for the research group that depends on these WBC models. However, there are more significant effects of WBCSim on the group's productivity, which were not so clearly predictable in advance. These effects are related to the synergistic nature of combining the models, an optimizer tool, and a visualization tool. The resulting integrated package immediately led to greater use of the simulation models by the researchers. Not only did they use the models more, but they used them to examine the design space in new ways. This is borne out by the fact that the researchers started to use parameter settings that caused the simulations to fail, that is, they uncovered new bugs in the simulation codes. Once these bugs were corrected, the researchers were able to continue more extensive use of the models.

It is likely that this integration effect will arise in many application domains. In general, an increased ability to integrate simulation, optimization, and analysis tools will allow researchers to get more out of their tools as they find new ways to make use of them.

Acknowledgement

This work was supported in part by a Virginia Polytechnic Institute & State University ASPIRES grant, USDA Grant 97-35504-4697, and NASA Grant NAG-2-1180.

References

[1] Akers, R.; Kant, E.; Randall, C.J.; Steinberg, S.; Young, R.L. (1997), "SciNapse: a problem-solving environment for partial differential equations," *IEEE Computational Science and Engineering*, **4**(3), 32–42.

[2] Ames, A. L.; Nadeau, D. R.; Moreland, J. L. (1996), *VRML 2.0 Sourcebook, 2nd ed.*, John Wiley & Sons, Inc., 241–295.

[3] Balay, S.; Gropp, B.; Curfman McInnes, L.; Smith, B. (1998), "A microkernel design for component-based parallel numerical software systems," Technical Report ANL/MCS-P727-0998, Argonne National Laboratory.

[4] Bramley, R.; Gannon, D.; Stuckey, T.; Villacis, J.; Akman, E.; Balasubramanian, J.; Breg, F.; Diwan, S.; Govindaraju, M. (1998), "The linear system analyzer," Technical Report TR-511, Computer Science Dept, Indiana University.

[5] Beca, L.; Cheng, G.; Fox, G.C.; Jurga, T.; Olszewski, K.; Podgorny, M.; Sokolowski, P.; Walczak, K. (1997), "Java enabling collaborative education, health care and computing," *Concurrency Practice and Experience*, **9**, 521–534.

[6] Boisvert, R.F.; Rice, J.R. (1985), "Solving elliptic problems using Ellpack," Springer-Verlag, New York.

[7] Casanova, H.; Dongarra, J. (1997), "NetSolve: a network server for solving computational science problems,"

International Journal of Supercomputer Applications and High Performance Computing, **11**, 212–223.

[8] Catlin, A.C.; Chui, C.; Crabill, C.; Houstis, E.N.; Markus, S.; Rice, J.R.; Weerawana, S. (1994), "PDELab: an object-oriented framework for building problem solving environments for PDE based applications," 2nd Object-Oriented Numerics Conference, (A. Vermeulen, ed.), Rogue Wave Software, Corvallis, OR, 79–92.

[9] Chabert, A.; Grossman, E.; Jackson, L.; Pietrovicz, S. (1997), "NCSA Habanero: synchronous collaborative framework and environment," Conference Supplement of the Fifth European Conference on Computer-Supported Cooperative Work (ECSCW'97), Lancaster, UK, 7–8.

[10] Czyzyk, J.; Owen, J.H.; Wright, S.J. (1997), "NEOS: optimization on the Internet," Technical Report OTC-97/04, Argonne National Laboratory.

[11] Dabdub, D.; Manohar, R. (1997), "Performance and portability of an air quality model," *Parallel Computing*, **23**(14), 2187–2200.

[12] Disz T.L.; Evard R.; Henderson, M.W.; Nickless, W.; Olson, R.; Papka, M.E.; Stevens, R. (1995), "Designing the future of collaborative science: Argonne's Futures Laboratory," *IEEE Parallel & Distributed Technology*, **3**(2), 14–21.

[13] Dyksen, W.R.; Ribbens, C.J. (1987), "Interactive ELLPACK: an interactive problem solving environment for elliptic partial differential equations," *ACM Transactions on Mathematical Software*, **13**, 113–132.

[14] Ford, B.; Iles, R.M.J. (1985), *The What and Why of Problem Solving Environments for Scientific Computing*, in Problem Solving Environments for Scientific Computing, B. Ford and F. Chatelin (Eds.), Elsevier Science Publishers, North-Holland, 3–18.

[15] Gallopoulos, E.; Houstis, E.; Rice, J.R. (1994), "Computer as thinker/doer: problem-solving environments for computational science," *IEEE Computational Science and Engineering*, **1**, 11–23.

[16] Gannon, D.; Bramley, R.; Stuckey, T.; Villacis, J.; Balasubramanian, J.; Akman, E.; Breg, F.; Diwan, S.; Govindaraju, M. (1998), "Component architectures for distributed scientific problem solving," *IEEE Computational Science and Engineering*, **5**(2), 50–63.

[17] Goldin, D.S.; Venneri, S.L.; Noor, A.K. (1998), "Beyond incremental change," *IEEE Computer*, **31**(10), 31–39.

[18] Houstis, E.; Gallopoulos, E.; Bramley, R.; Rice, J.R. (1997), "Problem-solving environments for computational science," *IEEE Computational Science and Engineering*, **4**, 18–21.

[19] Houstis, E.N.; Rice, J.R.; Weerawarana, S.; Catlin, A.C.; Papachiou, P.; Wang, K.Y.; Gaitatzes, M. (1998), "PELLPACK: a problem solving environment for PDE-based applications on multicomputer platforms," *ACM Transactions on Mathematical Software*, **24**, 30–73.

[20] Isenhour, P.L.; Begole, J.; Heagy, W.S.; and Shaffer, C.A. (1997), "Sieve: A Java-based collaborative visualization environment," Late Breaking Hot Topics Proceedings, IEEE Visualization '97, Phoenix, AZ, 13–16.

[21] Joshi, A.; Weerawarana, S.; Ramakrishnan, N.; Houstis, E.N.; Rice, J.R. (1996), "Neuro-fuzzy support for problem-solving environments: a step toward automated solution of PDEs," *Special Joint Issue of IEEE Computer and IEEE Computational Science and Engineering*, **3**(1), 44–56.

[22] Kamke, F.A.; Wilson, J.B. (1985), "Computer simulation of a rotary dryer: retention time," *American Institute of Chemical Engineers J.*, **32**(2), 263–268.

[23] Kamke, F.A.; Wilson, J.B. (1985), "Computer simulation of a rotary dryer: heat and mass transfer," *American Institute of Chemical Engineers J.*, **32**(2), 269–275.

[24] Knox, R.G.; Kalb, V.L.; Levine, E.R; Kendig, D.J. (1997), "A problem-solving workbench for interactive simulation of ecosystems," *IEEE Computational Science and Engineering*, **4**(3), 52–60.

[25] Long, K.; Van Straalen, B. (1998), "PDESolve: an object-oriented PDE analysis environment," Proceedings of the SIAM Workshop on Object Oriented Methods for Inter-operable Scientific and Engineering Computing, Philadelphia, PA.

[26] Markus, S.; Weerawarana, S.; Houstis, E.N.; Rice, J.R. (1997), "Scientific computing via the Web: the Net Pellpack PSE server," *IEEE Computational Science and Engineering*, **4**(3), 43–51.

[27] Marti, J. (1997), "Viewing IGES files through VRML," Proceedings of Visualization'97, IEEE Computer Society Press, Los Alamitos, CA, 471–472.

[28] Mniszewski, S.M.; Beckman, P.H.; Fasel, P.K.; Humphrey, W.F. (1998), "Efficient coupling of parallel applications using PAWS," Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC7), Chicago, IL.

[29] Parker, S.G.; Weinstein, D.M.; Johnson, C.R. (1997), "The SCIRun computational steering software system," in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Birkhauser Press, 1–40.

[30] Phanouriou, C.; Abrams, M. (1997), "Transforming command-line driven systems to Web applications," Sixth International World Wide Web Conference, Santa Clara, CA, 1–3.

[31] Regli, W.C., "Internet-enabled computer-aided design," *IEEE Internet Computing*, **1**(1), 39–50.

[32] Resnik, J.; Kamke, F.A. (1998), "Modelling the cure of adhesive-wood bonds using high frequency energy," Final Report, U.S.-Slovene Joint Board on Scientific and Technological Cooperation. Project 95-AES10. University of Ljubljana, Ljubljana, Slovenia.

[33] Schiesser, W.E. (1994), *Computational Mathematics in Engineering and Applied Science: ODEs, DAEs, and PDEs*, CRC Press, Boca Raton.

[34] Shah, A. (1998), *Symphony: A Java-based Composition and Manipulation Framework for Distributed Legacy Resources*, M.S. Thesis, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA.

[35] Umetani, Y.; Tsuji, M.; Iwasawa, K.; Hirayama, H. (1987), *DEQSOL: A Numerical Simulation Language for Vector/Parallel Processors*, in Problem Solving Environments for Scientific Computing, B. Ford and F. Chatelin (Eds.), Elsevier Science Publishers, North-Holland, 147–162.

[36] Vanderplaats Research & Development, Inc. (1995), *DOT Users Manual, Version 4.20*, Colorado Springs, CO.

[37] Weerawarana, S.; Houstis, E.N.; Rice, J.R.; Joshi, A. (1996), "PYTHIA: a knowledge based system to select scientific algorithms," *ACM Transactions on Mathematical Software*, **22**, 447–468.

[38] Weggel, C.F. (1997), "Versatile 2-D analysis," *IEEE Spectrum*, **34**(8), 92–93.

[39] Wolfram, S. (1996), *The Mathematica Book, 3rd ed.*, Wolfram Media/Cambridge University Press, 148–155.