# Unification of Problem Solving Environment Implementation Layers with XML*


Jiang Shu† Layne T. Watson‡
Naren Ramakrishnan† Frederick A. Kamke¶
Chris North†



**Abstract**

This paper describes how XML is used to unify the implementation layers of a problem solving environment WBCSim. WBCSim is a Web based simulation system designed to increase the productivity of wood scientists conducting research on wood-based composites manufacturing processes. WBCSim serves as a prototypical example for the design, construction, and evaluation of small-scale problem solving environments. WBCSim supports five process models. A XML datasheet is tailored for each model. The WBCSim interface layer, server scripts, and database management system all use this XML datasheet to improve the usability and maintainability of the three layers—client, server, developer—comprising the WBCSim system. A detailed description of the WBCSim system architecture is presented, along with a typical scenario of usage.



Keywords: XML, Experiment Management, Problem Solving Environment, Computing Environment, Wood-based Composite Materials, Database Management.


## 1 Introduction

A problem solving environment (PSE) is a computational system that provides a complete and convenient set of high-level tools for solving problems from a specific domain [14]. A PSE commonly addresses many issues: Internet accessibility to legacy codes, visualization, experiment management, multidisciplinary support, recommender systems, collaboration support, optimization, high performance computing, preservation of expert knowledge, design extensibility, and pedagogical uses [23].

A PSE often has many development cycles. Features addressing the above issues are added step by step depending on various interests/feedback, priorities, and/or the dynamic nature of the problem. During the process of adding more features, much information could be duplicated at


* This work was supported in part by a Virginia Polytechnic Institute and State University 1997 ASPIRES grant, USDA Grant 97-35504-4697, and DOE contract DE-AC04-95AL97273-91830.



† Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106.

‡ Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106.

¶ Department of Wood Science and Forest Products, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0503.

various implementation layers. This situation obviously makes the system hard to maintain and change, and hinders the productivity of developers and scientists.

This paper addresses this problem of information duplication by describing how XML can be used to unify the implementation layers for the problem solving environment WBCSim. A XML datasheet is tailored for each model (WBCSim has five models). The WBCSim interface layer, server scripts, and database management system all use this XML datasheet to improve the usability and maintainability of the three layers—client, server, developer—comprising the WBCSim system. XML is the acronym for Extensible Markup Language, which allows a developer to create customized tags, supporting the definition, transmission, validation, and interpretation of data between applications and between organizations.

WBCSim is a problem solving environment that increases the productivity of wood scientists conducting research on wood-based composite (WBC) materials and manufacturing processes. It integrates Fortran 90 simulation codes with a Web-based graphical front end, an optimization tool, and various visualization tools. The WBCSim project was begun in 1997 with support from USDA, Department of Energy, and Virginia Polytechnic Institute & State University (VPI). It has since been used by students in several wood science classes, by graduate students and faculty, and by researchers at several forest products companies. User feedback has resulted in numerous changes to the interface and underlying models, and was the major impetus for using XML as a mechanism for unification. Replacing the batch file mode of use by the Web interface and supporting optimization for manufacturing process design have had a major impact on the productivity of wood scientists using the analysis codes in WBCSim.

Goel et al. [10], [9] described an early version of WBCSim. In 2002, Shu et al. [19] described how WBCSim had further evolved by taking different approaches to its architecture, adding more sophisticated models, and switching from an experiment-oriented to a manufacturer-oriented approach. In 2004, Shu et al. [18] described a WBCSim experimental management component, which integrates a Web-based graphical front end, server scripts, and a database management system to allow scientists to easily save, retrieve, and perform customized operations on experimental data. However, the application's original goals remain the same: (1) to increase the productivity of WBC research and manufacturing groups by improving their software environment, and (2) to continue serving as an example for the design, construction, and evaluation of small-scale problem solving environments.

WBCSim primarily serves as a test bed for the design, construction, and evaluation of small-scale PSEs. WBCSim qualifies as a PSE for the following reasons: it makes legacy simulation codes available via the World Wide Web; it is equipped with visualization and optimization tools; it is multidisciplinary; it will soon be augmented with experiment management, collaboration support, and high performance computing.

Concomitant with its multifunctional ability of running simulation experiments and generating and storing data, WBCSim is quite difficult to modify and maintain. Upon making a change such as adding an input parameter to a model, a developer needs to change the proper interface code, wrapper code, visualization code, and database schema. This series of changes can sometimes be tedious and tricky in order to maintain the integrity of the whole system. This paper describes efforts to use XML to resolve this problem and reduce redundancy. Succinctly, all implementation layers access the same XML datasheet for a particular model. A simple change in the XML can cause cascading changes at various implementation layers, thereby significantly simplifying the development effort and improving maintainability.

The paper is organized as follows. Section 2 reviews some related work in PSEs and XML usage in PSEs. Section 3 briefly describes WBCSim. Section 4 gives the motivation for applying XML in WBCSim. Section 5 elaborates on the WBCSim user interface, the model interface similarities, and also the unique aspects of each. Section 6 explains the architecture of WBCSim and how the XML fits in. Section 7 demonstrates typical scenarios of using and changing WBCSim. Section 8 outlines some future directions for WBCSim and draws conclusions.

## 2 Related Work

There are many problem specific PSEs developed for various application domains. Some of those PSEs utilize the XML technology.

### 2.1 Problem Solving Environments

A PSE is a relatively new computing paradigm that was first introduced in problem domains such as partial differential equations (ELLPACK [10] and its descendents [5] for solving two- and three-dimensional elliptic partial differential equations) and linear algebra (Linear System Analyzer [3] for manipulating and solving large-scale sparse linear systems of equations). There is also SCIRun [13], developed to interactively compose, execute, and control (computational steering) large-scale scientific computations.

Since then, many new PSEs have been introduced. Gismo [2], created at Washington University, is an object oriented Monte Carlo package for modeling all aspects of a satellite's design and performance. It has played a significant role in the design of the Gamma Ray Large Area Space Telescope, the successor to the Compton Gamma Ray Observatory that was launched into space in 1991 to explore the gamma ray portion of the electromagnetic spectrum for astrophysics research. VizCraft [8], developed at Virginia Polytechnic Institute and State University, provides a graphical user interface for a widely used suite of analysis and optimization codes that aid aircraft designers during the conceptual design of a high speed civil transport. VizCraft combines visualization and computation, encouraging the designer to think in terms of the overall problem solving task, not simply using the visualization to view the computational results. Also, a computing environment developed by Chen et al. [7], which combines particle systems, rigid body particle dynamics, computational fluid dynamics, rendering, and visualization techniques to simulate physically realistic, complex dust behavior, has been shown to be useful in interactive graphics applications for education, entertainment, or training.

Watson et al. [23] present a thorough summary of the key attributes of a PSE (described in Section 1), and also compare a PSE with other similar computing environments—a decision support system (DSS) and a geographic information system (GIS). Roughly, a DSS emphasizes the functionality (analysis, planning, and decision making), a GIS emphasizes the nature of data and information (spatial), while a PSE emphasizes the problem domain (such as wood-based composite materials).

## 2.2 XML in Problem Solving Environments

The XML technology is often applied in PSEs in two ways: 1) describing entities such as models, components, and services, and 2) describing relationships between entities such as calls, messages, and connectivity. Often XML is used in conjunction with other technologies such as SOAP, CORBA, and Java.

Rana et al. [17] describe a PSE for scientific computations and large-scale simulations. This environment is based on the Java programming language, with components being Java or CORBA objects defined with XML interfaces. Youn et al. [25] describe Application Web Service Toolkits, which consist of a set of core services built using the Web services model and application metadata services. A PSE can be built from these core services, which also support multiple versions of services. XML is used to describe, access, and discover those core services in a distributed computing paradigm. PROTEUS [6] is a grid based PSE for composing, compiling, and running bioinformatics applications on the grid. One of the main components—metadata repository—uses XML documents to represent installed software components and heterogeneous data sources. Beca [4] focuses on the collaboration aspect of a PSE by using a technology called Shared Place, which allows construction of virtual places on the Web by merging collaborative tools with Web page content. Shared Place Definition Language (SPDL), an XML based language, describes properties of a virtual place: Place appearance, behavior of collaboration components present in the Place, and access to the Place resources.

BSML [22] is a markup language proposed to formally specify data interchange assumptions underlying scientific codes. It addresses a problem similar to that studied in this paper but focused on a PSE for wireless system design [20]. In contrast to the tasks considered here, BSML is aimed at validation (ensuring that data is available in the appropriate format to execute the model), binding (associating a data stream with a computational model), and conversion (suitably massaging data if format conversions are required). There are numerous other efforts, primarily in the area of grid computing (see, e.g., [16]), aimed at using XML to declaratively specify problem solving tasks, scheduling runs, and managing scientific data.

# 3 WBCSim

WBCSim is intended to increase the productivity of wood scientists conducting research on wood-based composite materials by making legacy file based FORTRAN programs that solve scientific problems in the wood-based composites domain widely accessible and easy to use. WBCSim currently provides Internet access to command line driven simulations developed by the Wood-Based Composites Center at VPI. WBCSim leverages the accessibility of the World Wide Web to make simulations with legacy code available to scientists and engineers away from their laboratories.

WBCSim currently supports five simulation models that help wood scientists studying wood-based composite material manufacturing.

(1) Rotary dryer simulation (RDS). The rotary dryer simulation model assists in the design and operation of the most common type of system used for the drying of wood particles [11], [12].
(2) Radio-frequency pressing (RFP). The radio-frequency pressing model [15] was developed to simulate the consolidation of wood veneer into a laminated composite using high frequency energy.
(3) Oriented strand board mat formation (OSB). The mat formation model [19] creates a three-dimensional spatial structure of a layered wood-based composite (e.g., oriented strand board and waferboard); it also calculates certain mat properties by superimposing a mesh on the mat structure.

(4) Hot compression (HC). The hot compression model [19] simulates the mat consolidation and adhesive cure that occurs during industrial hot-pressing of wood-based panels.
(5) Composite material analysis (CMA). The composite material analysis model [19] was developed to assess the stress and strain behavior and strength properties of laminated materials (e.g., plywood and fiber-reinforced composites).

The current software architecture of WBCSim follows the three tier model described by Goel et al. [10], in which the tiers are (1) the client layer—user interface, (2) the server layer—Web server and a PHP module, and (3) the developer layer—legacy simulation codes and various optimization and visualization tools running on the server. Section 6 elaborates the details of the three layers while explaining how the XML technology fits into this architecture.

WBCSim is equipped with an optimization algorithm DOT (Design Optimization Tool) [21] and various visualization tools: VRML [1], Mathematica [24], and the UNIX utility WhirlGif. The reader is referred to [19] for an in-depth treatment of these tools.

## 4 Rationale for Applying XML Technology

Before the introduction of XML technology into WBCSim, the WBCSim Web interface (originally in Java, now in PHP) is hard-coded. At the beginning of each model interface code, there are sections to declare and initialize the model input parameters, which are sequentially used to construct the user interface, and store the input values. The simulation wrappers (in PERL) are used to envelop the FORTRAN simulations and connect various tools (such as the database, visualization tools, and optimization tools). These wrappers also have sections in their code for the input parameters, output parameters, and other corresponding scripts involved (such as Mathematica scripts for graphics). Moreover, the database (Postgresql) also has database table schemas for each model to define its input and output parameters and files.

This architecture results from adding features step by step. A typical PSE has many development cycles in which different features are added at different times depending on developer interest, user feedback, priorities, and the dynamic nature of the problem. WBCSim, for example, had its interface rewritten from Java into PHP; optimization tools, visualization tools, and the database were all added at different times and by different developers. The problem is that much information (such as input parameters) is duplicated at various implementation layers. When making a change at the interface layer, such as adding an input parameter to a model, a developer needs to locate the proper code to define the information about the parameter (such as name, physical units, minimum, maximum, and default), and then pinpoint the place to add the parameter (normally in a HTML table). Then, the corresponding simulation wrappers and the database table schemas need to be carefully changed as well, in order to maintain the integrity of the entire system. Therefore, it is very hard for a developer to change and maintain such a system, especially when the developer needs to communicate often with wood scientists requesting changes. Misunderstanding and wasted programming effort can occur, making the process even more inefficient.

What is required is that the information about the model be stored at one place so that all the components at various implementation layers can easily access such information. This approach would permit changes at one place to cause cascading changes in the entire system. Toward this end, a XML datasheet is tailored for each model in WBCSim. The WBCSim interface, server scripts, and database management system all use this XML datasheet to improve the usability and maintainability of the three layers—client, server, and developer—comprising the WBCSim system. The details of XML usage are explained in Section 6—WBCSim Architecture. This approach significantly simplifies development time and improves maintainability in the following ways.

(1) The information about a model is stored in its XML datasheet. All components from various implementation layers refer to this same datasheet. This eliminates any duplicate information.
(2) Changing the XML datasheet will cause cascading effects, which will alter the interface, wrappers, and database accordingly.
(3) The XML datasheet is flexible and simple enough to allow wood scientists to easily define and directly modify a model by themselves.
(4) The XML datasheet can be verified via its XML Schema (a standard to describe and validate data in an XML environment) to ensure its integrity and (syntactic) correctness.
(5) A set of XML datasheets can be labeled as one version of the system to support disaster recovery. A scientist could recover an old model if a newer process model is lost (from hardware crashes).

## 5 User Interface

In 2002, Shu et al. [19] described a Java applet interface for WBCSim. That Java applet interface (Browser plug-in technology) has since been replaced by a PHP (server scripting technology) interface. Java applets require a user to have a Java-enabled browser. Since different browsers may have different versions of the Java plug-in, the developers stuck to a very early version of Java (Java 1.0) to ensure that the applets worked on all user computers (especially old computers at industrial manufacturing sites). This old version of Java not only has a very limited set of graphical user interface APIs, but also is very buggy. Moreover, this old Java applet interface has many pop-up windows in a very strict hierarchy (parent and child relationship). If a user accidentally closes the wrong window, the whole application could crash. It was apparent there was a need to consolidate the Web interface while simultaneously getting away from the plug-in technology (browser/computer dependent). After carefully consideration and assessment, PHP was chosen to replace Java applets in WBCSim. PHP is the acronym for Hypertext Preprocessor, which is an open source, server-side, HTML embedded scripting language used to create dynamic Web pages. Other similar server scripting technologies are JSP (Sun) and ASP (Microsoft). Upon any HTTP request, the server runs PHP scripts and sends plain HTML code back to the client, where no plug-in is needed. While implementing this new PHP interface, most of the old pop-up windows were incorporated into frame-based Web pages. Studies with student users prove the new interface significantly improves WBCSim's usability and is much more user friendly.

Today a user launches the WBCSim Web page from a browser window, which contains a list of the process simulation models that WBCSim supports. Under each model, a user can either choose to enter (define parameters for) a simulation model by clicking "Use This Model as Guest" or read more information about the model by clicking "Detailed Description". Upon choosing a particular simulation model, a user has the choice of creating a new simulation, or investigating the stored simulation runs. There are additional choices at the right top corner of the page to allow a user to go back to the home of the current model, switch models, refresh a page, enable frames, submit comments, and logout; there is also a question mark icon that leads to help. These additional choices are always at the top and the bottom (in case the Web page is too long) of the page to help a user. There are two kinds of stored simulation runs—incomplete ones and complete ones. For incomplete simulation runs, a user can resume and finish off the simulation. For complete simulation runs, a user can view the result of that simulation, change the input values of the simulation to run it again, and perform an "Advanced Option". In the "Advanced

Option", a user is allowed to filter the simulation runs displayed. By selecting either "Title" or "Date" radio buttons, a user can narrow the filter process to either field. Then a user can input a regular expression (in UNIX grep style) in the text field, and press the "Filter" button to execute the filter process, which will reload this Web page with the simulation runs that meet the filter criteria. The "Advanced Option" also allows a user to apply comparison functions by selecting an item from the "Compare Option" dropdown list. Each process model in WBCSim has its own comparison functions. For example, the OSB model has three, which allow a user to compare "Void Fractions/Contact Area", "Density", and "Coefficient of Variation" among the simulation runs selected. If "Void Fractions/Contact Area" is selected from the "Compare Option" dropdown list, a new page is loaded with more options that allow comparisons among "space/mat", "lumen/mat", "void/mat", "lumen/flake", and "contact area". A user can also specify the interval of the y-axis in order to zoom the comparison graph. To the right of these stored simulation runs, there is information about model specific alerts and announcements and a user profile.

If a user decides to create a new simulation, he will actually see the specific model interface. There are some common features among all the model interfaces (see Fig. 1). Under the "WBC-Simulator" title, there is a series of names and arrows to show the current navigation stage. In the left frame, there are many links corresponding to the steps necessary to run a particular process simulation model. When these links are clicked, the corresponding Web pages will show up in the right frame. The first link is "Introduction" that explains the model behavior at an abstract level, and also gives general directions about how to use the model. The next link is "Set Name", which allows a user to name (describe) the simulation. Following "Set Name", there are model-dependent links to allow a user to specify necessary simulation parameters in the right frame with any number of text boxes, dropdown lists, radio buttons, or plain buttons. At the bottom of the left frame, there are links to "Run Simulation", "Save Inputs", "Save Results" and "Delete". The "Run Simulation" link will display a Web page to confirm and send the HTTP request to execute the proper PHP code with input values defined in the interface. This will cause a UNIX or PERL script to run compiled FORTRAN code, and any additional optimization or visualization codes. The "Save Inputs" link will only save the input values in the database as an incomplete simulation run, while the "Save Results" link will save both inputs and outputs in the database as a complete simulation run. The "Delete" link will delete the current simulation run from the database and any associated files. The "View Results" link will appear right under "Save Inputs" if a simulation is executed. The "View Results" will show the simulation results in both textual (normally tables of numbers) and graphical (VRML files, GIF files) forms.

The RDS (Fig. 1) and RFP models have the simplest user interfaces in WBCSim. There are only few model-dependent links such as "Edit Variables", "Edit Wood Layer Properties", and "View Constants". A user can simply enter values for various input parameters through text boxes and then run the simulation. The CMA model has evolved to support six calculations: "Design", "Analysis", "General Strength", "Tensile Strength", "Shear Strength", and "Bending Strength". A user can select a dynamically loaded calculation from a dropdown list in "Select Submodel". The CMA model includes two sets of defaults—structure and loading—in "Input Structure" and "Input Loading", which allow a user to cross reference different sets of defaults by setting one structure parameter set and then running any calculation with any loadings. The CMA model has rows of input data representing layers of the wood composite. A user can activate a layer by selecting the left most checkbox for that layer.

The OSB model (Fig. 2) is similar to the CMA model with input data representing layers of the wood composite. However, these layers in the OSB model are incorporated into the left frame, because there are too many properties for each layer. A user can enter the number of layers at "Edit Sim Variables." After clicking save, the links corresponding to each individual layer are displayed in the left frame menu. A user can further define the layer properties through these links. For example, clicking "Edit Layer 1", a user can select desired distributions for "Length", "Width", "Thickness", and "Density", along with "Orientation", "Flake Color", and "Number of Flakes to Display" for the first layer. Clicking the "Distribution" link, a user can further define the actual (probability density) distributions. For example, if selecting the "Empirical" distribution in the "Length Distribution" dropdown list of link "Edit Layer 1", a user will see there are forty-one text boxes under "Length Distribution" in the link "Distribution", allowing a user to define the number of intervals and each interval's starting and ending points. Also, as in the CMA model, the OSB model provides a set of defaults rather than a single default like most of the other models.

The HC model (Fig. 3) interface differs from the other models' interfaces by dividing the simulation parameters into three groups in the left frame menu: (1) model inputs, (2) model parameters (execution specification), and (3) model outputs. The first group contains links for specifying the parameters related to the material, initial condition, boundary condition, adhesive cure, and press schedule. The second group defines the HC model execution (numerical solution of a nonlinear partial differential equation) by specifying mat transport properties, boundary transport properties, and compression properties. The last group specifies the output, such as color or black and white, the time interval for generating frames for animation, and other parameters determining which data/image files are generated.

## 6 WBCSim Architecture

The current software architecture of WBCSim follows the three-tier model described by Goel et al. [10], in which the tiers are (1) the client layer—user interface, (2) the server layer—Apache Web server and a PHP module, and (3) the developer layer—legacy simulation codes and various optimization and visualization tools running on the server. These layers are shown in Fig. 4.

### 6.1 Client Layer

The client layer is the only layer visible to end-users and typically the only layer running on the local machine. The main part of the client layer is the user interface, which is described in the previous section.

The user interface is generated from PHP and XML. As described earlier, each model has a XML datasheet, which uses a set of customized tags to describe the model. The size of the XML datasheets in WBCSim varies from 109KB (around 7000 lines) to 18KB (800 lines) depending on how complicated the model input is. A typical reason for a large XML datasheet is that the XML datasheet contains information about many input/output parameters, and many sets of default values. Fig. 5 shows the actual XML datasheet for the RDS model. Some common tags among all the models are:

```
<simulation>
        <simulation_name></simulation_name>
        <simulation_name_abbreviation></simulation_name_abbreviation>
        <simulation_image></simulation_image>
        <short_description></short_description>
        <long_description></long_description>
        <announcement></announcement>
        <parameter_section>
                <parameter>
```

```
                        <name></name>
                        <unit></unit>
                        <min></min>
                        <max></max>
                        <default></default>
                        <help_text></help_text>
                </parameter>
        </parameter_section>
</simulation>
```

At the beginning, the PHP script reads this XML datasheet and uses a XML parser to parse the information into an array structure. Then different parts of the PHP script use this array to construct interfaces, set parameter defaults, store user inputs, and assemble the inputs into a string (the corresponding simulation wrapper will feed this string to the FORTRAN simulations). The first row of input parameters in the browser window shown in Fig. 1 is actually generated from a portion of the XML file shown in Fig. 5. Therefore, adding a parameter in the XML datasheet will simply add an element in the dynamic array. Then, the PHP code referring to this array will automatically update the interface when executed.

The client layer also contains viewers for one of the visualization tools, the VRML translator. WBCSim requires a VRML 2.0 viewer for the RFP model. The VRML viewer serves as a plug-in to the Web browsers. A user has other substitute visualization results if the VRML plug-in does not exist.

The client layer handles communication with the server layer by sending HTTP requests. When a user clicks a link, a URL appended with necessary action commands and parameters is sent to a Web sever in the server layer, where the corresponding PHP scripts, along with other possible PERL scripts, visualization tools, and database operations, are executed.

## 6.2 Server Layer

By separating the legacy simulation codes from the user interface, the server layer functions as the key to how WBCSim can run a text-only application from a Web browser. The server layer consists of two components: an Apache HTTP server and a PHP module.

In 2002, Shu et al. [19] described a telnet connection method, which replaced the old Javamatic server and socket communication paradigm described in [10], and facilitated setting up WBCSim guest accounts on the server machine that do not require full account privileges. The guest account and normal user accounts (for commercial, paying users) permit users to save and track their different simulation runs, and protect their data from being removed or overwritten by others. However, this telnet connection needs the client to use a special network port to communicate with the server. In today's world of highly tightened security, it is problematic to ask a user to open/monitor a special port on his computer in order to use WBCSim. A standard Apache HTTP server with a PHP module only requires the standard HTTP port 80, which is the standard and normally open/monitored port at the client computer for Web browsing.

The Apache HTTP Sever Project (http://httpd.apache.org/) is a collaborative software development effort aimed at creating a robust, commercial grade, featureful, and freely available source code implementation of an HTTP (Web) server. The project is jointly managed by a group of volunteers located around the world, using the Internet and the Web to communicate, plan, and develop the server and its related documentation. The October 2004 Netcraft Web Server Survey (http://news.netcraft.com/archives/web_server_survey.html) found that more than 67% of the Web sites on the Internet are using Apache, thus making it more widely used than all other Web servers combined.

The PHP module serves as a plug-in to the Apache HTTP server. When the HTTP server receives a request for a PHP file, the PHP module will parse and execute that PHP file and return HTML back to client. This PHP module supports sessions. A session is a series of related interactions between a single client and the Web server, which can take place over an extended period of time. By using sessions, the PHP module can concurrently accept multiple requests from different clients and direct executions of multiple simulations.

### 6.3 Developer Layer

As its name suggests, the developer layer contains legacy programs created by researchers to model WBC materials and manufacturing processes. These legacy programs are the heart of WBCSim. In general, WBCSim supports legacy programs written in any programming language as long as the program takes its input parameters from UNIX stdin. The input parameters can be a few numbers or a large data file. In particular, WBCSim supports five FORTRAN 77 and Fortran 90 simulation programs corresponding to the five models described in Section 3.

While each Fortran program has its own input format (stdin could be redirected to a file), the server layer communicates data with the developer layer via strings of parameters separated by white space (spaces, tabs, newlines). In order to cope with this string format, each legacy program is “wrapped” with a customized Perl script. The script receives this string of parameters from the PHP module at the server, and converts those parameters into an appropriate format for the Fortran program. Then the script calls the legacy program into action, feeding it the input, invoking any required optimization and visualization tools, packing all Fortran output in HTML files, and returning HTML files, first to the server layer, and then to the client layer. With this architecture, the developer layer is independent of the other layers, which makes the process of designing, and integrating new simulation codes relatively easy.

Originally, these wrappers had a set of input parameters hard coded in the code. Today these wrappers use the XML datasheet to convert the string of input parameters into the appropriate format for the Fortran program, similar to how the interface layer will read/parse the same XML file, and utilize array structures (described in section 6.1).

Shu et al. [18] described some additional operations in those wrappers to support the experiment management component. There are also other wrappers associated with the save, retrieve, and compare functions from the user interface at the client layer. Like the simulation wrapper, these wrappers receive data from the server layer, and convert those input values into an appropriate format for the database. Originally, this conversion process was also hard coded. However, work converting WBCSim to use the same XML sheet to construct those necessary queries and database tables is currently underway.

Other than wrappers, the developer layer also includes optimization and visualization tools to maximize the simulation’s value to the user [19]. Since the wrappers prepare the inputs for and read the outputs from these optimization and visualization tools, these tools do not interact with the XML datasheets directly.

## 7 Scenarios

## 7.1 Typical Usage Scenario

Describing a typical usage scenario of WBCSim is instructive. Consider research into the properties of oriented strand board products, which would use the OSB model. First, a scientist samples the face and core layer flakes from an industrial production line. The scientist measures the geometry and the weight of each flake. The flake property data sets are then used to estimate the statistical probability density functions of the flake properties. Then, the scientist opens a browser window and launches the WBCSim Web page, which is dynamically generated from PHP and XML. The guest account is the default login option, which stores all the input and output data in a temporary directory that is purged periodically. The scientist selects the OSB model from a list of all the available models by clicking "Use This Model as Guest". The scientist then launches the OSB model interface (shown in Fig. 2) by clicking "New Mat Formation". By specifying the probability density functions of the flake properties obtained offline earlier, the scientist can create a statistically valid model instance of the spatial structure of multiple layers of flakes, each with various parameters defining the number of flakes, length distribution, width distribution, thickness distribution, density distribution, orientation, flake color, and number of flakes shown. The latter two are solely for visualization.

Next, upon user requests, the input parameters (Boolean, numeric, or alphanumeric) are collected in the PHP code at the server as a long string (parameters are separated by white space). Then, the PHP module calls the OSB wrapper to convert this string into a SQL query, and checks if this set of input parameters is stored in the database.

If the output does not exist in the database, the OSB wrapper converts this input string into a data file destined for the OSB Fortran 90 simulation program by referencing the corresponding OSB XML datasheet. Since the OSB simulation code has its own text-based user interface taking input from stdin, a temporary file is generated to contain all of the appropriate commands. Stdin is then redirected to this temporary data file for the simulation. The OSB wrapper then invokes the OSB simulation code with this temporary file as stdin along with the properly formatted data file. When the simulation code is executing, the OSB wrapper monitors the simulation output stream for strings indicating execution milestones. The OSB wrapper uses the standard error stream for storing these messages into a log file, which a user can also access from the Web by clicking "Check Status".

If the output exists in the database (a simulation with this exact same input data has been run earlier, and the output saved), the OSB wrapper extracts the output from the database and places it into data files in the same format as it would have been written by the OSB Fortran 90 simulation code.

When those output data files are ready (no matter whether they came from the Fortran 90 simulation program or the database), the OSB wrapper calls Mathematica to read those data files and generate plots with various Mathematica commands such as ListContourPlot, ListPlot3D, MultipleListPlot, and Graphics3D. Mathematica also converts these internal graphics data structures into GIF format so that they can be viewed in a browser. Fig. 6 shows a three-dimensional visualization of a three-layer random flake mat created with Graphics3D by Mathematica. Finally, the OSB wrapper embeds these GIF files in HTML files and returns these HTML files to the PHP module. These HTML files are passed back to the client, where a user can click "View Results" to access them. Shu et al. [18] further described scenarios involving a comparison between results from multiple simulation runs.

### 7.2 Typical Change Scenario

Describing a typical change scenario for WBCSim is effective to demonstrate how XML unifies the components in the three implementation layers. Consider adding an input parameter—"Difference in Moisture Content at Face from Core"—in the "Initial Conditions" section for the HC model. A scientist initiates this change by adding this parameter in the Fortran simulation code (its input, its algorithm, and its output). In order to change the rest of the system, this scientist can download a copy of the HC XML datasheet from the Web (for example, http://.../xml/hot.xml). After locating the parameter section defining "Initial Conditions", the scientist can duplicate the tags and contents of an existing parameter as a template, and then edit and add this new parameter's full name, short name, physical units, and default value. This scientist can then submit this updated XML datasheet to a developer, who can verify the syntax of this datasheet by running it against a XML Schema. After properly backing up the old version of the XML datasheet, the new one can be plugged in. Then, the interface at the client layer and the wrappers at the developer layer will access this new XML datasheet instantly when a new session is started. The corresponding interface will have this new parameter as a textbox; the wrappers will prepare this new parameter as an additional input in the input data file for the HC simulation code. In the future, necessary scripts can even backup and alter the database tables as well, which is yet to be implemented in WBCSim.

The above scenario has many assumptions. Obviously, the XML datasheet only supports a set of customized tags (such as the ones listed in Section 6.1). If it is a simple change like above, a wood scientist can easily insert the new input parameter by using existing tags, and obey the XML structure. If a change requires extra tags (to define some special information related to a parameter or a set of parameters), certainly a wood scientist needs to consult with a developer to coordinate the modification of the XML schema (such as adding more tags and embracing more complex XML structures). The above scenario also assumes that the output stays the same after adding the new input parameter. That is seldom the case. Therefore, a wood scientist would also need to locate the output related sections within the XML datasheet to add possible additional output information, so that the wrappers can parse such information from the XML datasheet describing how to deal with the additional output.

How to perform such change management automatically is an active area of research in database systems and, needless to say, is quite critical in PSE research as well. By explicitly encoding modeling and execution assumptions in a declarative format such as XML, the ability is reserved to revisit WBCSim system design at a later stage and provide more sophisticated change management facilities. In summary, changes in XML datasheets can automatically propagate to the entire PSE system assuming:

(1) The change requires only existing tags.
(2) The change obeys existing XML structure—XML schema.

## 8 Future Work and Conclusions

Shu et al. [19] described planned additions of experiment management (EM), collaboration support, and high performance computing, which are the most appealing missing features given the current state of WBCSim development. While WBCSim is being developed on several different fronts, the development track for incorporating XML technology intends to support automatic database changes, be more flexible and dynamic, and improve XML storage.

## 8.1 Support Database Change

Currently, when a XML datasheet is changed, the behavior of the interface and wrappers accessing this XML datasheet will change accordingly. The database, however, has to be changed manually to store modified simulation runs. PERL scripts can be developed to backup the existing simulation runs, and use the XML to alter/create the new database tables automatically.

## 8.2 More Flexible

The existing set of the XML tags customized for WBCSim has yet to be tested for usability and flexibility. A wood scientist should be able to understand and change the XML datasheet easily. The goal is to make changing the XML datasheet a trivial task for a wood scientist who does not know much about XML.

## 8.3 More Dynamic

Upon submission of an updated XML datasheet, a developer is still needed to verify the syntax of this XML datasheet, and monitor whether the change can be realized in the WBCSim interface and wrappers (for example, a change should not affect WBCSim interface usability). Perhaps this process can be further automated with editing, submitting, verifying, and testing a XML datasheet being done by a wood scientist himself using a Web interface.

## 8.4 XML Storage

The XML files are currently stored as plain text files under a certain directory in the file system. When the XML files are updated, the old ones are renamed with a date. Certainly a better way of archiving/documenting these XML files is needed, especially when a set of XML files can be used to define a version of WBCSim.

## 8.5 Conclusions

WBCSim has evolved steadily from a prototype PSE intended as a tool for computer science PSE research and a Web-based interface for a few legacy computer programs, to a manufacture-oriented near commercial quality PSE that is seriously used by wood science researchers in industry and academia, which makes WBCSim models stable enough so that information about these models can be encoded in XML. Since interesting computational capabilities are still lacking [19], WBCSim will remain an object of computer science research for some time to come. Yet the program's interfaces, models, and output visualizations are now good enough to be used as production tools by wood scientists. The directions in which computer scientists would like to take WBCSim (collaboration, data mining, grid computing) are quite different from the directions that wood scientists would prefer for WBCSim (new models, refining existing models, more interface and visualization options). The present layered architecture of WBCSim supports these divergent development directions well.

With the addition of XML technology, WBCSim is now considerably easier to change and maintain. From the user perspective, using XML technology significantly reduces the development time so that users can enjoy new features more quickly. From the developer perspective, under the assumptions explained in Section 7.2, a wood scientist can edit the XML datasheet himself to make a change, instead of spending time explaining to a programmer how to make such a change. Furthermore, a developer doesn't have to hardcode such a change at various implementation layers, such as in the interface and wrappers.

The original stated goal [10] of WBCSim was to provide "an integrated set of facilities allowing wood scientists to concentrate on high-level problem solving rather than on low-level programming details and application scheduling/execution, allowing users to define, record, and modify problems, and visualize and optimize simulation results". Using XML technology to unify the WBCSim implementation layers has definitely brought this goal closer.

## References

[1] Ames AL, Nadeau DR, Moreland JL. VRML 2.0 Sourcebook, 2nd ed. New York: John Wiley & Sons, Inc., 1996.

[2] Burnett T, Chaput C, Arrighi H, Norris J, Suson DJ. Simulating the Glast satellite with Gismo. IEEE Computing in Science and Engineering 2000; 2:9–18.

[3] Bramley R, Gannon D, Stuckey T, Villacis J, Akman E, Balasubramanian J, Breg F, Diwan S, Govindaraju M. The linear system analyzer. Technical Report TR-511, Dept. of Computer Sci., Indiana Univn, Bloomington, IN, 1998.

[4] Beca L. Building Collaborative Problem-Solving Environments as Shared Places. 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9, Maui, Hawaii, January 2001.

[5] Boisvert RF, Rice JR. Solving elliptic problems using ELLPACK. New York: Springer-Verlag, 1985.

[6] Cannataro M, Comito C, Schiavo F, Veltri P. PROTEUS: a grid based problem solving environment for Bioinformatics. Workshop on DataMining Ontology for Grid Programming (KGGI 03), Halifax, Canada, 2003.

[7] Chen JX, Fu X. Integrating physics-based computing and visualization: modeling dust behavior. IEEE Computing in Science and Engineering 1999; 1:12–16.

[8] Goel A, Baker CA, Shaffer CA, Grossman B, Mason WH, Watson LT, Haftka RT. VizCraft: a problem-solving environment for aircraft configuration design. IEEE Computing in Science and Engineering 2001; 3:56–66.

[9] Goel A, Phanouriou C, Kamke FA, Ribbens CJ, Shaffer CA, Watson LT. WBCSim: a prototype problem solving environment for wood-based composites simulations. Technical Report TR98-25, Dept. of Computer Sci., Virginia Polytechnic Institute and State Univ., Blacksburg, VA, 1998.

[10] Goel A, Phanouriou C, Kamke FA, Ribbens CJ, Shaffer CA, Watson LT. WBCSim: a prototype problem solving environment for wood-based composites simulations. Engineering with Computers 1999; 15:198–210.

[11] Kamke FA, Wilson JB. Computer simulation of a rotary dryer: retention time. AIChE J. 1985; 32:263–268.

[12] Kamke FA, Wilson JB. Computer simulation of a rotary dryer: heat and mass transfer. AIChE J. 1985; 32:269–275.

[13] Parker SG, Weinstein DM, Johnson CR. The SCIRun computational steering software system. Modern Software Tools in Scientific Computing, Arge E, Bruaset AM, Langtangen HP (eds.), Birkhauser Press, 1997, 1–40.

[14] Rice JR, Boisvert RF. From scientific software libraries to problem-solving environments. IEEE Comput. Sci. Eng. 1996; 3(3):44–53.

[15] Resnik J, Kamke FA. Modeling 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, Ljubljana, Slovenia: University of Ljubljana, 1998.

[16] Ramakrishnan N, Watson LT, Kafura DG, Ribbens CJ, Shaffer CA. Programming environments for multidisciplinary grid communities. Concurrency and Computation: Practice and Experience 2002; 14(13–15):1241–1273.

[17] Rana OF, Li MZ, Walker DW, Shields MS. An XML based component model for generating scientific applications and performing large scale simulations in a meta-computing environment. GCSE 1999; 210–224.

[18] Shu J, Watson LT, Ramakrishnan N, Kamke FA, Zombori BG. An experiment management component for the WBCSim problem solving environment. Advances in Engineering Software 2004; 35:115–123.

[19] Shu J, Watson LT, Zombori BG, Kamke FA. WBCSim: an environment for modeling wood-based composites manufacture. Technical Report TR02-01, Dept. of Computer Sci., Virginia Polytechnic Institute and State Univ., Blacksburg, VA, 2002.

[20] Skidmore R, Verstak A, Ramakrishnan N, Rappaport TS, Watson LT, He J, Varadarajan S, Shaffer CA, Chen J, Bae KK, Jiang J, Tranter WH. Towards integrated PSEs for wireless communications: experiences with the $S^4W$ and projects. ACM SIGMOBILE Mobile Computing and Communications Review ($MC^2R$) 2004; 8(2):20–34.

[21] Vanderplaats Research & Development, Inc. DOT Users Manual, Version 4.20. Colorado Springs, CO: 1985.

[22] Verstak A, Ramakrishnan N, Watson LT, He J, Shaffer CA, Bae KK, Jiang J, Tranter WH, Rappaport TS. BSML: a binding schema markup language for data interchange in PSEs. Scientific Programming 2003; 11(3):199–224.

[23] Watson LT, Lohani VK, Kibler DF, Dymond RL, Ramakrishnan N, Shaffer CA. Integrated computing environments for watershed management. Journal of Computing in Civil Engineering 2002; 16(4):259–268.

[24] Wolfram S. The Mathematica Book, 3rd ed.. Champaign, IL: Wolfram Media/Cambridge University Press, 1996.

[25] Youn C, Pierce ME, and Fox GC. Building problem solving environments with application Web service toolkits. International Conference on Computational Science (ICCS03) 403–412, Melbourne, Australia, June 2003.

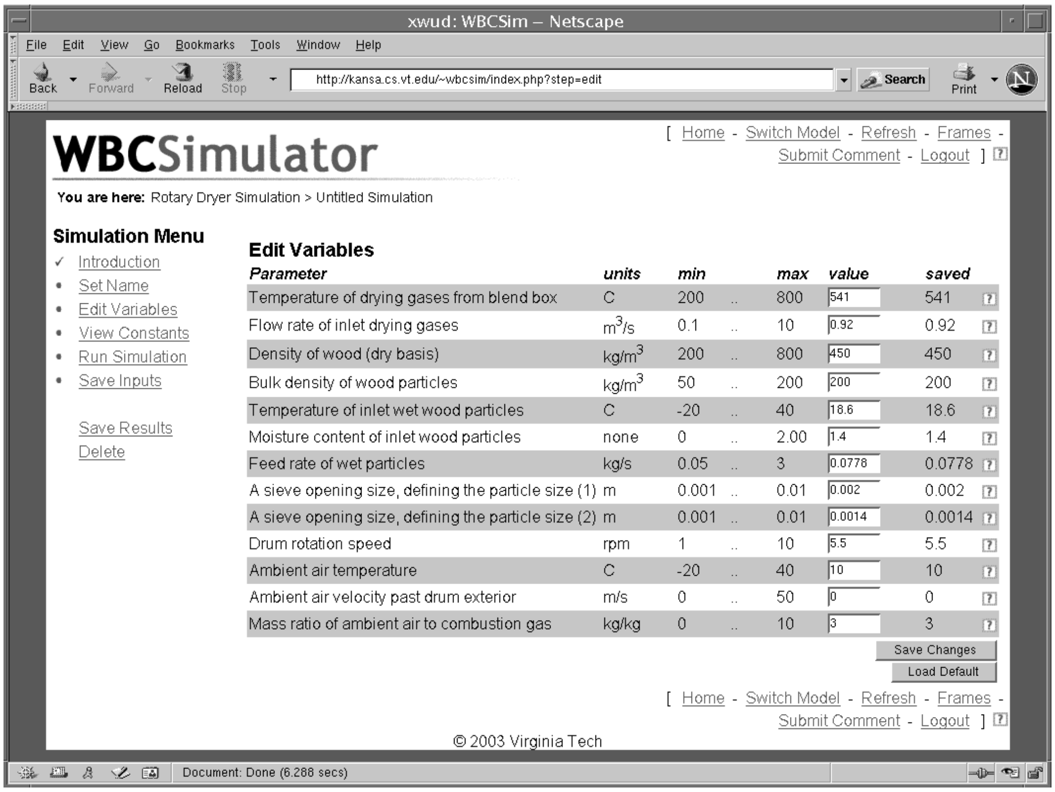


FIG. 1. *The RDS model user interface.*

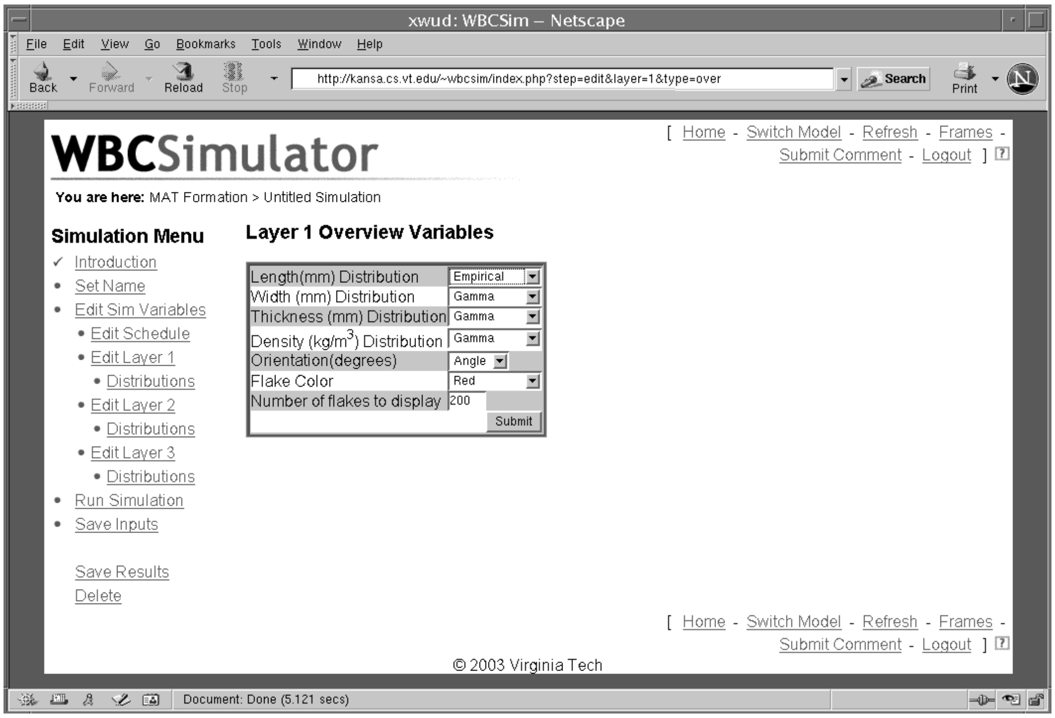


FIG. 2. *The OSB model user interface.*

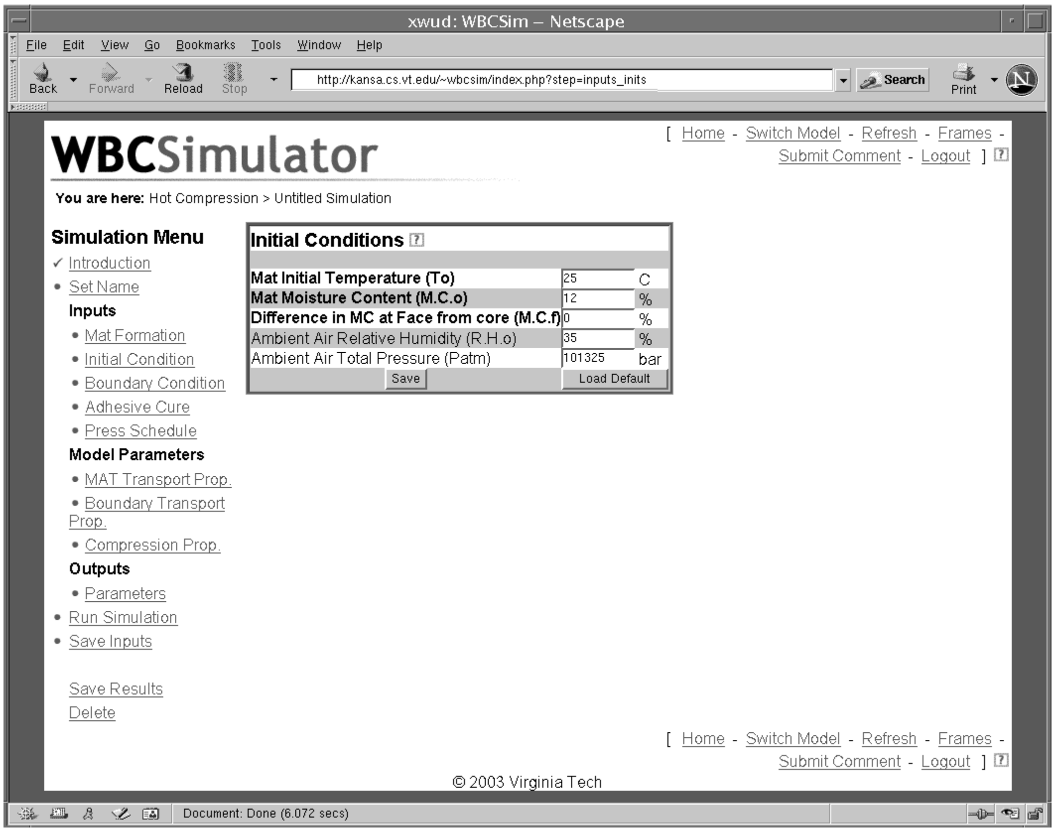


FIG. 3. *The HC model user interface.*

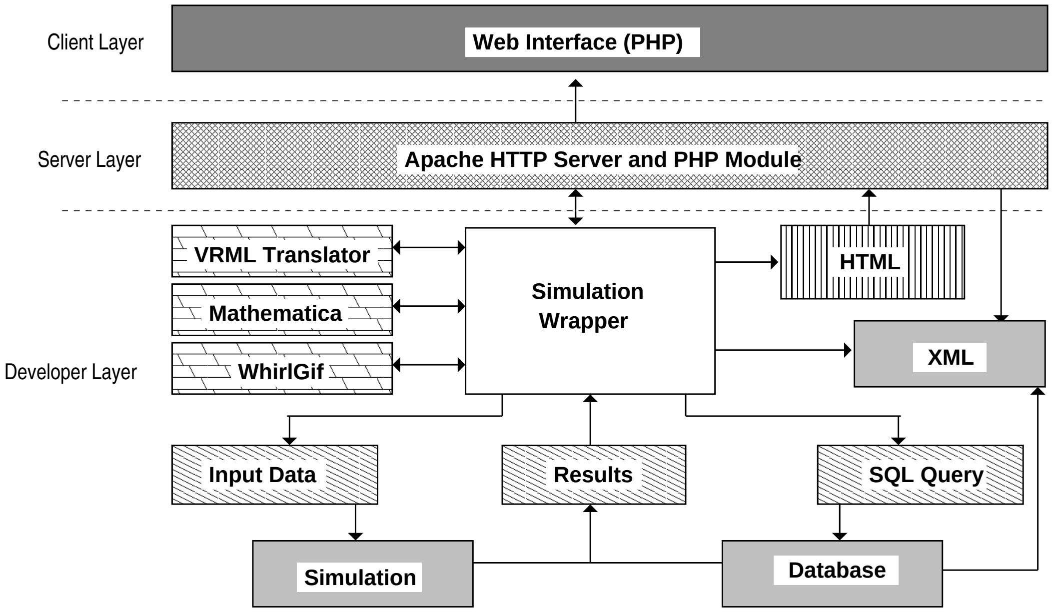


FIG. 4. *WBCSim architecture overview.*

```
<?xml version="1.0"?>
<simulation>
  <simulation_name>Rotary Dryer Simulation</simulation_name>
  <simulation_name_abbreviation>RDS</simulation_name_abbreviation>
  <simulation_image>../img/Rotary_Dryer.gif</simulation_image>
  <short_description>This model simulates the drying behavior of wood particles in a rotary
    dryer.</short_description>
  <long_description>The rotary dryer simulation model 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.<br><br>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.<br><br>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.<br><br>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.<br><br></long_description>
  <announcement></announcement>
  <parameter_section>
    <title>Variable</title>
    <parameter>
      <name>Temperature of drying gases from blend box</name>
      <unit>C</unit>
      <min>200</min>
      <max>800</max>
      <default>541</default>
      <help_text>Combustion gases from burner are mixed with ambient air, or recycled air, in the
        blend box. The airflow from the blend box is a mixture of these gases. This is the inlet
        air temperature to the rotating drum.<br><br><center><img src=../img/RDSHELP1.jpg
        alt=RDSHELP1.jpg border=0></center></help_text>
    </parameter>

                                    ⋮
```

FIG. 5. *The RDS model XML datasheet, an excerpt from the file* `rds.xml`*.*

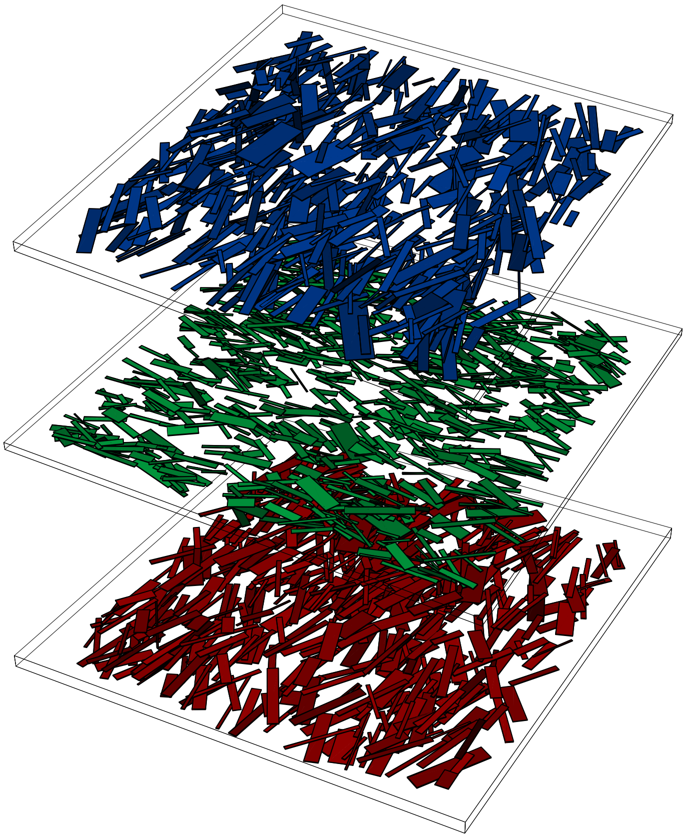

FIG. 6. *Visualization of the three layer random flake mat: the area of the mat is* $450 \times 450$ *mm. Only a fraction of the flakes are shown, and the shading designates the density of the flakes.*