Combined Research and Curriculum Development of Web-Based Educational Modules on Mechanical Behavior of Materials†

Ronald D. Kriz[1], Diana Frakas[2], Romesh C. Batra[3], Randy T. Levensalor[4] and Sanjiv D. Parikh[5]
Virginia Polytechnic Institute and State University

[1]Associate Professor, Dept. of Engineering Science and Mechanics (ESM), Norris Hall, Blacksburg, VA 24061, rkriz@vt.edu; [2]Professor, Dept. of Materials Science and Engineering, Holden Hall, Blacksburg, VA 24061, diana@vt.edu; [3]Professor, ESM, rbatra@vt.edu; [4]Graduate Research Associate (GRA), Dept. of Computer Science, McBryde Hall, Blacksburg, VA 24061, randyl@vt.edu; [5]GRA, ESM, engineer@vt.edu

Abstract

We describe development of modules funded by the National Science Foundation (NSF) program for Combined Research and Curriculum Development (CRCD). Construction of CRCD modules was designed both for research collaboration and educational pedagogy. Modules described here were used for teaching senior level undergraduate and first-year graduate level courses on mechanical behavior of materials that incorporate results of physics-based simulation models previously used by class instructors in their research. These modules were developed using a Java-based Graphical User Interface (GUI) called the Network Program Interface Builder (NPIB) that runs on a standard Web-server. Emphasis was placed on using existing Web-based commercial software and development of new Web-based software that could be used by others to create and maintain physics-based simulation models, and archive results. Strong emphasis was also placed on providing well documented user's guides so educators and researchers could build their own modules that would extend beyond the CRCD objectives of this project. For the educators and students, Web-access to simulation results allow instructors to monitor students' progress and enhance class participation in sharing and discussing results. As for the researchers NPIB can be used for collaborative development of new simulation models. For students in this project CRCD modules teach the basics of materials mechanical behavior that reflects the experience of the authors. The simulation results span various length scales, starting at the atomistic level, using embedded atom method techniques, and reaching finite element simulations at the continuum level. The modules attempt to stress the way in which macroscopic properties are controlled by phenomena at the atomistic and micro-structural levels. Advanced computation and visualization techniques, including simulation results in a CAVE™, are used to convey structure-property relationships where complex three-dimensional structures could be observed in a fully immersive virtual environment. Both courses were taught by an interdisciplinary team of materials scientists and engineering mechanicians. We will discuss our experience in teaching both courses and the lessons learned.

1. Background

Modules were developed and distributed on our "Wave-Java" Web-server[1]. Early efforts to create a distributed, Web-based, visual computing environment were funded by SUN Microsystems Inc. (SUN), Visual Numerics Inc. (VNI), and Virginia Tech's Advanced Communications and Information Technology Center (ACITC) by the creation of the Scientific Modeling and Visualization Classroom (SMVC). Modules discussed here were largely motivated by a student project, "Educational Atomic Models Using PV-Wave and Java" by Arturo Falck, in ESM4714: Scientific Visual Data Analysis and Multimedia, spring semester 1996[2]. The purpose of this project was to use existing Web-based commercial software and create new user-friendly *Web-based Java interface* that interacts with larger computer simulation models of cracks and dislocations in crystal lattices that would execute on remote computers.

Students used this *Web-based Java interface* to: 1) enter information required by the simulation, 2) compile that information into a data file, 3) submit the file as a batch job to a remote computer, and finally to 4) send raw data of simulation back to server where images of data were generated for viewing on the client's Web browser. Unique to this project was the level of industrial participation by SUN and VNI in the creation of the *Java Web-based interface.* Early Java/PV-Wave applet prototypes developed at Virginia Tech have been replaced with the JWave interface developed by VNI. The Network Program Interface Builder (NPIB) replaced the original *Web-based Java interface*. The same functionality was maintained except simulation execution was limited to the Web-server. A more detailed discussion on JWave and NPIB follows.

Development of the Web-server continued with funding from the NSF Combined Research and Curriculum Development (CRCD) Program [3]. With NSF funding, the Web-server was upgraded to a SUN Sparc10-Ultra with 1GB of memory that was used to execute larger simulations and generate more representative results for analysis by students. Further development of NPIB extended the execution of simulations to larger remote site computers, which provided students access to more

† Paper submitted to the Journal of Materials Education, July, 2001.
CAVE ™ is a trademark name of the Electronic Visualization Laboratory of the University of Illinois.

realistic simulation results. For this project two SUN Sparc10-Ultras provided an environment for development and experimentation, where one workstation was configured as the Web-server and the other was configured as the remote site computation computer.

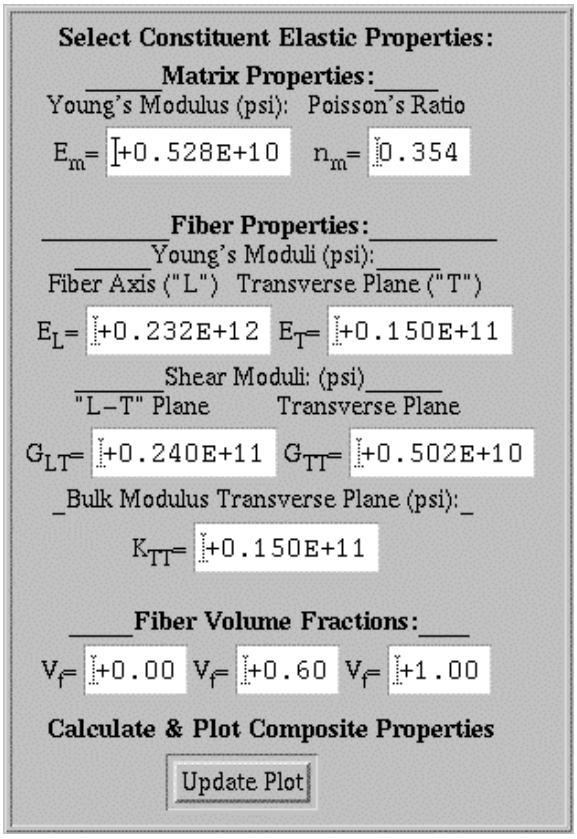

Figure 1(a). Module01 JWave-GUI: entry of micro-scale constituents (fiber/matrix) elastic properties.

The first course was organized on the Web server with hyperlinks to Web modules that were divided into lectures, assignments, and interactive modules [4,5]. The first course focused on atomistic and continuum mechanics models; the second course focused on micro-scale models that predict mechanical behavior at the scale between the atomistic and continuum. Details on module content development of the first course were given in an earlier publication [5]. Here we describe the development of the NPIB *Web-based Java Interface* and explain how the NPIB graphical user interface (GUI) was designed and how it facilitated students' efforts to parametrically study the mechanical behavior which was modeled and simulated by legacy computer programs written by the class instructors.

For comparison the same legacy code was used to develop CRCD micro-scale module01 (JWave-GUI) and module02 (NPIB-GUI). These two modules have been specifically documented as working examples that demonstrate how to setup JWave and NPIB GUI-interfaces[4]. These examples together with comprehensive user's guides for JWave and NPIB provide developers the necessary information to build modules that extend beyond the objectives of this CRCD project.

2. Creation of JWave and NPIB modules

Both JWave and NPIB modules are configured for execution on a Apache Web-server so that the user can place all modules in their ~<user-name>/public_html directory. Details for setting up JWave (module01) and NPIB (module02) in the user's public_html directory are available by studying the construction of these Web accessible modules located in the "Micro" section of the CRCD Web-server [4]. Setting up modules in a public_html directory facilitates module development, where multiple users can conveniently create and maintain their own Web-server modules without system administrator assistance. The systems administrator is only required to make one entry for each user in a file located in the cgi-bin directory.

2.1 JWave

Figure 1 shows the interactive JWave module01 applet GUI. This GUI form allows the student to fill in parameters, Figure 1(a), needed by the legacy code to calculate elastic property polar plots, Figure 1(b). When the "Update Plot" button is pressed the applet transfers parameters to a PV-Wave procedure which, plots results.

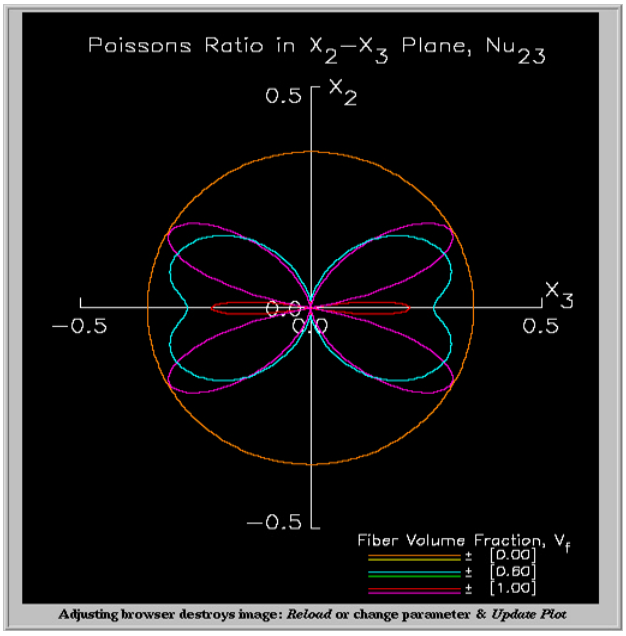

Figure 1(b). Module01 JWave-GUI: macro-scale elastic anisotropic polar plot of Poisson's ratio for a unidirectional fiber-reinforced composite material calculated from the micro-scale (fiber/matrix) properties shown in Figure 1(a).

Module01 was created for students to experiment with different mechanistic models that predict anisotropic elastic properties at the macro-scale which are caluclated from micro-scale constituent (fiber/matrix) elastic properties. Elastic anisotropy is fundamental to

understanding mechanical behavior of many engineering materials, which are highly anisotropic. Here only graphical results are returned through the JWave applet. The same micro-macro models were also used in module02 for comparison, where the NPIB GUI interface returns the same graphical results but along with all of the simulation model source code and data sets. This comparison demonstrates two different but equally useful GUIs and gives the developer a choice. In some cases only graphical results are needed in another case clients (students/researchers) may need more information for analysis and interpretation of simulation results.

Two files are required to draw the polar plots shown in Figure 1(b): the JWave applet embedded in an HTML document running on the client's Web-browser and the PV-Wave procedure file located on the JWave Web-server. Only graphical results are returned in this JWave module.

The JWave 3.0 applet was developed following a few simple easy-to-follow steps: 1) install JWave, 2) run the JWave server manager and configure the paths and directory structure for accessing the Java archive files (*.jar) and the compiled PV-Wave procedure files (*.cpr), 3) create an HTML forms applet, and 4) create PV-Wave procedure. The Java archive files contain all of the classes required by the JWave server that are also used to create java-script embedded HTML forms which act as the GUI interface for the data input to and the graphical output from PV-Wave procedures. Legacy PV-Wave procedure files need to be modified with a JWave API which communicates with the Java archive files.

Developing JWave applets requires a significant investment of time to learn how the javascript/JWave applet communicates with the PV-Wave procedure. This is done through JWave wrappers. JWave wrappers are VNI's in-house Java functions that are archived into the JWave<type>.jar files. These functions are used within the embedded HTML javascript to call and execute specific PV-Wave commands. The following is an example of how data from Figure 1(a) is passed from the JWave applet to a PV-Wave procedure which calculates and draws the polar plot of Poisson's ratio shown in Fig. 1(b).

There are a variety of these wrapper functions within JWave that allow the ease of coding simple yet effective HTML forms which access PV-Wave's massive data analysis and visualization toolkit. Since JWave is a commercially supported software package numerous examples and a comprehensive user's guide are available.

Beyond the JWave wrapper API, one only needs to be able to pick up basic javascript and the PV-Wave programming language. JWave was specifically designed to work with legacy PV-Wave procedure files. To generate web-based forms with text fields, buttons, check boxes, pull-down menus, etc., a basic understanding of simple javascript GUI generation is sufficient. The following code fragment is a sample HTML (javascript) code used to generate a text input field and return its value to the JWave applet's "setFormParam" function, which in this case sets values for all the variables 'em, num, efl, eft, *etc.*' displayed in Figure 1(a).

```
<!---------------------------- nu32polarplot.html -------------------------
  This file is located at
      ~rkriz/public_html/crcd/modules/module01/
  This file is called either by
      ~rkriz/public_html/crcd/modules/module01/show_all_three.html
   or ~rkriz/public_html/crcd/lectures/Geom_1.html
  This file executes nu32polar.cpr which is located at
      ~rkriz/public_html/crcd/bin
  The location of the *.cpr file is set by the JWave manager config
-->
<HTML><HEAD>
<TITLE>JWaveApplet Poisson's Ratio Polar Plot</TITLE>
<SCRIPT LANGUAGE=JavaScript>
.
// Application specific code
  var dataNeedsRecalc = true;
  function updatePlot() {
    with (document) {   // easier
      if (! FOOBAR.isStarted()) {
        setTimeout("updatePlot()", 250);
        return false;
      }
      // Set params for the plot from their form objects
      setFormParam(FOOBAR, JWavePlotForm.em);
      setFormParam(FOOBAR, JWavePlotForm.num);
      setFormParam(FOOBAR, JWavePlotForm.efl);
      setFormParam(FOOBAR, JWavePlotForm.eft);
      setFormParam(FOOBAR, JWavePlotForm.gflt);
      setFormParam(FOOBAR, JWavePlotForm.gftt);
      setFormParam(FOOBAR, JWavePlotForm.kftt);
      setFormParam(FOOBAR, JWavePlotForm.vf1);
      setFormParam(FOOBAR, JWavePlotForm.vf2);
      setFormParam(FOOBAR, JWavePlotForm.vf3);
      // make the plot
      if (dataNeedsRecalc) {
          FOOBAR.execute();
        dataNeedsRecalc = false;
      }
    }
    return false;   // prevent reload of page if called from FORM submit
  }
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="updatePlot()">
<FORM NAME=JWavePlotForm onSubmit="return updatePlot()">
<table border width=820 height=600>
<tr><th>
<APPLET NAME="FOOBAR"
        CODE="com.visualnumerics.jwave.JWaveApplet"
        CODEBASE="../../../jwave/classes"
        ARCHIVE="JWaveConnectInfo.jar, JWave.jar"
        WIDTH=480
        HEIGHT=568>
  <PARAM NAME="FUNCTION" VALUE="nu32polar">
  <PARAM NAME="EXECUTE_ON_START"   VALUE="NO">
</APPLET>
.
</BODY></HTML>
```

The next code fragment is the PV-Wave procedure with the JWave API wrapper, which transfer values for all variables, 'em, num, efl, eft, *etc.*' into the PV-Wave procedure file for execution.

```
;---------------------- filename: nu32polar.pro -------------------------------
; This file is called by                 | Compile nu32polar.pro & save as*.cpr
; nu32polarplot.html in same directory   | WAVE> .run, nu32polar
; This file is located at                | WAVE> compile, 'nu32polar'
;~home/public_html/crcd/modules/module01 | The compile creates nu32polar.cpr
;-------------------------------------------------------------------------------
; !! NOTE: file nu32polar.cpr must be located at ~rkriz/public_html/crcd/bin !!
;-------------------------------------------------------------------------------
 function NU32POLAR, client_data;!! JWave UNCOMMENT !!
;                                !! JWave UNCOMMENT !! p-array elements below
  p=fltarr(10)
  p(0) = getParam(client_data, 'em',   /Value, Default = +0.528E+10)
  p(1) = getParam(client_data, 'num',  /Value, Default = +0.354)
  p(2) = getParam(client_data, 'efl',  /Value, Default = +0.232E+12)
  p(3) = getParam(client_data, 'eft',  /Value, Default = +0.150E+11)
  p(4) = getParam(client_data, 'gflt', /Value, Default = +0.240E+11)
  p(5) = getParam(client_data, 'gftt', /Value, Default = +0.502E+10)
  p(6) = getParam(client_data, 'kftt', /Value, Default = +0.150E+11)
  p(7) = getParam(client_data, 'vf1',  /Value, Default = +0.00)
  p(8) = getParam(client_data, 'vf2',  /Value, Default = +0.60)
  p(9) = getParam(client_data, 'vf3',  /Value, Default = +1.00)
.
.
.
;--- Above this line is the "wrapper" need to interface with the JWave API ---
;-------------------------------------------------------------------------------
;------ Below this line is the legacy PV-Wave code previously written --------
;--------------------------- ORIGINAL: nu32.pro --------------------------------
.
```

```
.
.
return,0
end
```

To an experienced Java programmer coding these objects in javascript is relatively easy. The only coding needed is creating an HTML/javascript file and modifying a legacy PV-Wave procedure to include a JWave wrapper API. To make learning this task easier, VNI created a few demonstration JWave applets which have the buttons, pull-down, and input fields pre-coded into the HTML javascript forms. With a basic understanding of the outline of the applet, it is easy to generate web-based interactive data analysis and visualization toolkits via few modifications to VNI's JWave applet demos and existing PV-Wave procedures.

To aid in development of future JWave applets, a Web page with links to download the JWave software, download the sample demos, and download the tutorials as well as the JWave user's guide was set up in the "tools" section of the CRCD Web-server [4].

2.2 NPIB [6]

From the start NPIB was designed as a simple intermediary between students and complex legacy engineering programs developed by the class instructors. Use of existing tools, platform independence, and minimal system administration are major themes, which guided the development cycle [6]. Java using SUN's JDK 1.2 was chosen as the primary development platform. JDK 1.2 runs on all major Web-servers.

The major difference between JWave and NPIB is that JWave returns only graphical results and NPIB was designed to display and **archive all** simulation model **results** each time the client submits a unique parametric job for that module. Simulation results are archived as a Web accessible <unique-directory-name>, which can then be accessed by anyone interested in research collaboration or class participation.

NPIB can be divided into six distinct phases: 1) Creation of NPIB forms, 2) Form Display, 3) Data Delivery, 4) Local Execution, 5) Remote Execution, and 6) Results Delivery.

2.2.1 Creation of an NPIB form

NPIB form creation encompassed the majority of the development time. Instructors create the forms, which will collect the data from the students. The look and feel of the forms and the format of the files that will be used by the simulation legacy code are decided at this phase. NPIB versions 1.0 and 1.3 used a single GUI to create both the layout of the form and the legacy code file format. This approach was cumbersome and inflexible. Therefore, NPIB was improved in version 1.4, which used a scripting language to describe the layout of the forms and the legacy code file. This gave the class instructors more flexibility and control. Due to the strict control of the language syntax, modifying these forms with a text editor was a laborious task. Since a unique syntactical language was implemented there was a large learning curve associated with version 1.4 of NPIB.

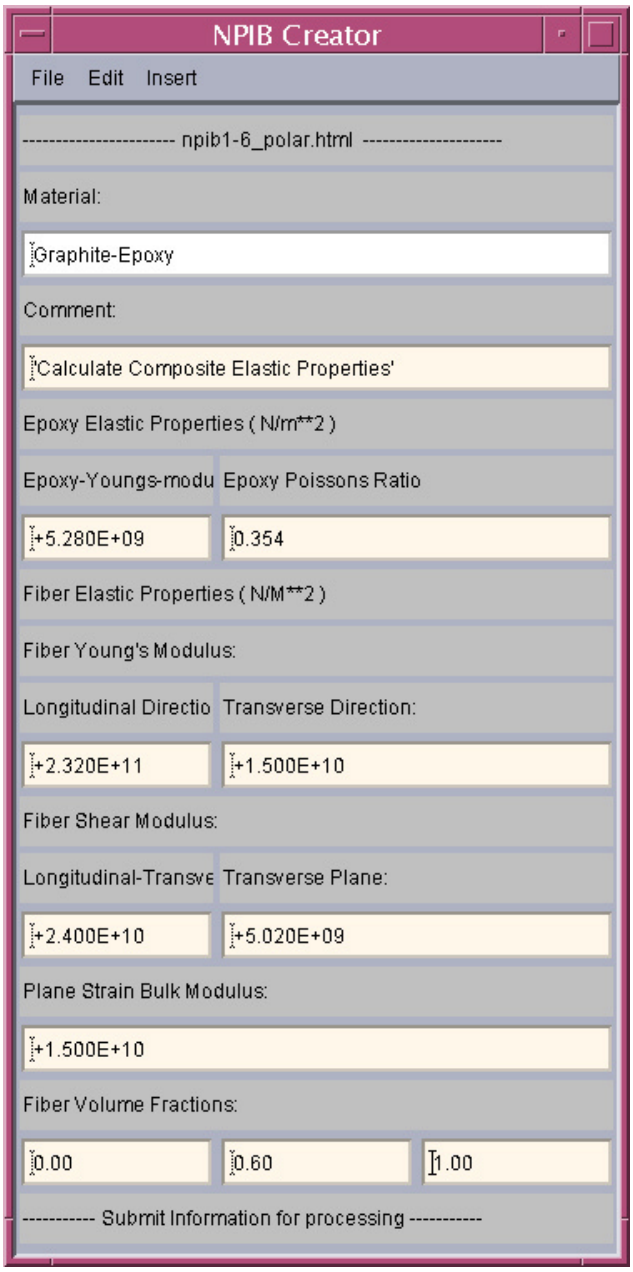

Figure 2. "NPIB Creator", version 1.6, creates a window that is used by class instructors for constructing NPIB forms so that the final Web-based form looks the same as the form viewed by the client on a Web browser except for information needed for remote processing, see Figure 5.

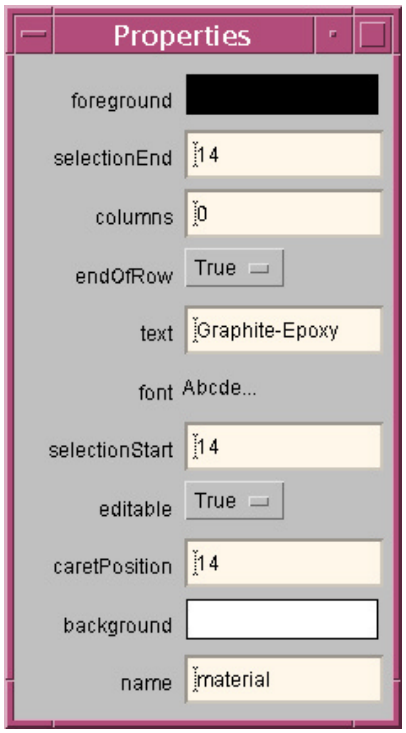

Figure 3. NPIB component properties are assigned here.

The current versions, NPIB 1.5 and 1.6, incorporate the strong points from previous designs, while attempting to eliminate their major drawbacks. A GUI is used to layout the forms through a WYSIWIG approach, see Figure 2. Instructors are able to directly manipulate the form components and view them, as the students view the form on the Web browser. Properties for the specific form elements are adjusted by changing values in a dialog similar to Java Beans [7], see Figure 3. The output is formatted using a simpler and more intuitive language, see Figure 4.

At present no formal usability tests have been conducted to verify these claims. However, direct feedback from users indicates that the goals laid out for the latest version of NPIB have been satisfactorily achieved. Although the latest version of NPIB is simpler and more intuitive, like JWave, development of NPIB modules requires a significant investment in time. Therefore most faculty will not develop their own modules. Instead modules will most likely be developed by graduate students or technical staff.

2.2.2 Form Display:

The same NPIB form file created by the class instructors, is displayed for the client (student/researcher) as an applet by using a standard web browser, see Figure 5. This approach was chosen because of the wide availability of web browsers. Note that the NPIB form shown in Figure 5, as viewed by the client, has additional information at the bottom of the form for submitting simulation to a specific remote site computer.

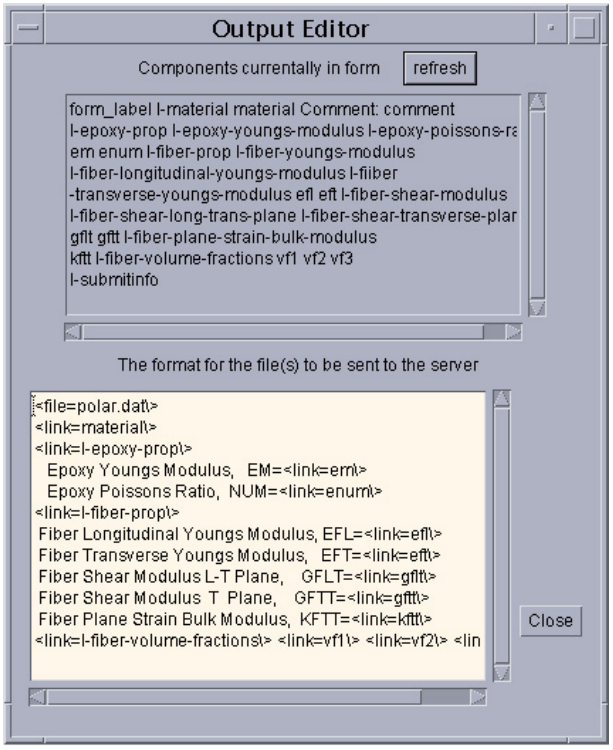

Figure 4. "Output Editor" controls format of the data file, <file=polar.dat>, used by legacy simulation code.

2.2.3 Data Delivery:

Deliver the data from the client to be processed on the Web-server. The data is submitted to the Web-server via a socket. Before the data is sent, it is broken into formatted files. The Web-server receives the text and dumps it into the appropriate files with the names specified from the form. The Web-server, which receives this data, is written in Java.

2.2.4 Local Execution:

Since several methods are in place to execute programs no additional programs were created to aid in the actual execution of the programs. Current implementations use basic shell scripts, which mirror commands as they would by typed if being run manually. The location of the data files and parameters passed from the form are passed as command line parameters in a script file. Others methods of execution can be used including c, c++, Fortran, Perl or any other executable languages may be used.

2.2.5 Remote execution ("Receiver"):

Initially instances of the receiver were run on machines other than the web server to execute programs remotely. This caused several complications at the implementation level, exercise level, and administrative level. This required the receiver be installed and always running as a daemon on any machine these programs needed to run on. Finally, we

looked to existing tools and found several solutions. The preferred method is to use "expect" script and secure shell (ssh) to move files and execute the simulation computer programs remotely, which is accomplished with command line script files.

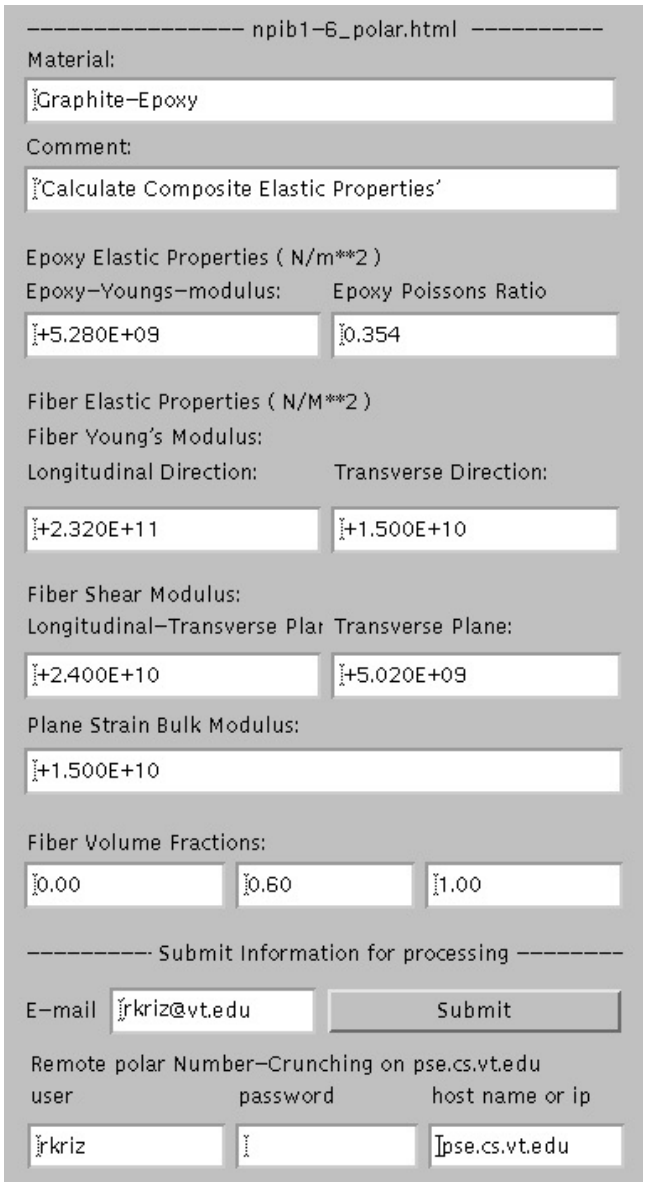

Figure 5. NPIB form as viewed on a client's Web browser, which is used to submit batch jobs to remote computers.

2.2.6 Results Displayed:

Simulation results are returned to the client (student/ researcher) by opening a new Web browser window and by sending the client email when simulation is completed. When the file transfer to the remote computer is complete, a new web browser window is opened and displays the directory where simulation results appear as they are returned to the Web-server. Within this directory the client can view simulation results in real-time, see Figure 6. When the simulation is completed an email is also sent to the client with a link to this Web address of simulation results. NPIB version 1.6 archives simulation results within a unique directory structure: "../user-name/application/client-email-date-time".

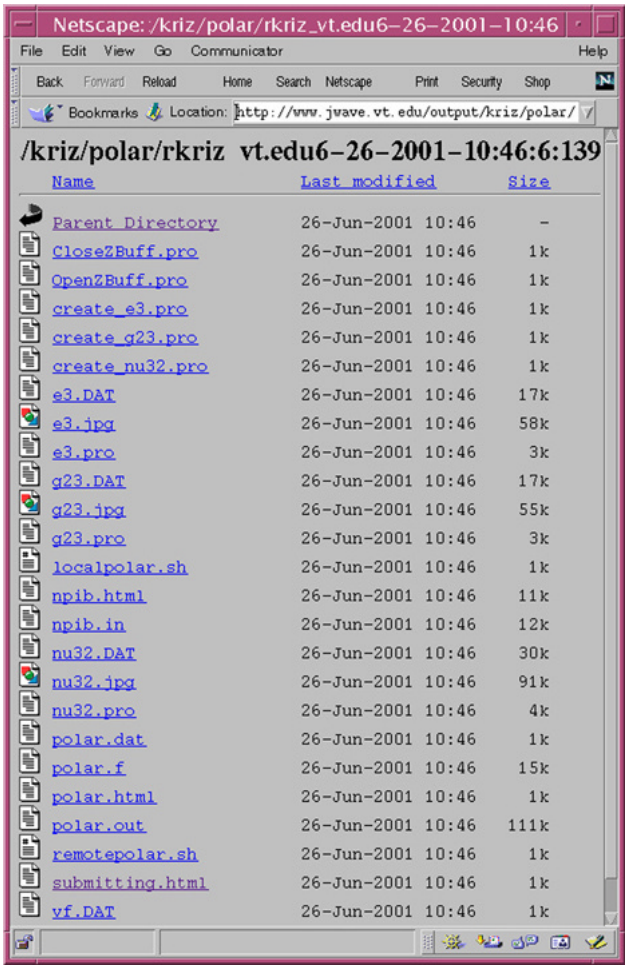

Figure 6: "Working" real-time archive of simulation results for calculating elastic property polar plots of a unidirectional fiber-reinforced composite. For comparison the same polar plot of Poisson's ratio shown in Figure 1(b) can be viewed by selecting nu32.jpg file in this directory.

3. Results and Lessons Learned

3.1 ESM-MSE4984 and ESM-MSE5984

The first NSF-CRCD senior-level course (ESM-MSE4984) was taught in the Fall semester, 1998 and a second class (ESM-MSE5984), a first-year graduate level class, was taught in the Fall Semester, 2000. Both classes are three-credit hour classes, meeting for one hour three times a week: Monday, Wednesday, and Friday. Mondays and Wednesdays were reserved for lectures. On Fridays students met with instructors in the Scientific Modeling and Visualization Classroom (SMVC) of the ACITC. The students used the SMVC more for the undergraduate class than for the graduate class. Because of the improvements

in the CRCD Web-server the students used the CRCD modules outside of the SMVC lab, which explains the decreased use of the SMVC for the graduate class. Results of the undergraduate class are posted in the "ESMMSE4984" section of the CRCD Web-server [4]. Student projects were also posted at this same location.

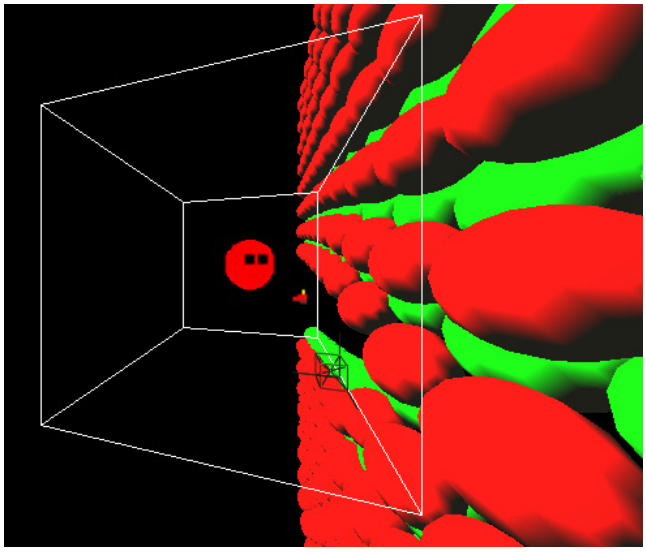

Figure 7. CAVE simulation view of a Mode-I crack propagation along Ni-Al grain boundary (nano-scale module01). White lines simulate the CAVE room boundaries. The head is simulated by a sphere near the center of the CAVE room with two black dots for eyes which can only simulate the fully immersive experience of being inside the CAVE.

For both classes, a list of lecture topics and interactive computer simulation modules is organized in Table 1 for each of the scales: nano-scale, mirco-scale, and macro-scale. Most of the interactive computer modules were associated with the lectures and homework assignments. Two nano-scale modules and one micro-scale module were designed for use in the CAVE. For these three modules students could open the Web URL address on the CAVE computer and directly load the inventor (*.iv) files into the CAVE using either the SARAnav or PFnav applications. The students could then navigate through these complex three-dimensional (3-D) structures. Figure 7 shows a student, red spherical head with two black eyes, observing a mode-I crack propagating along a Ni-Al grain boundary. From this simulation students can observe dislocation emission for the crack tip which results in a crack with a larger crack radius, hence higher fracture strength.

Figure 8 shows the elastic anisotropy of a special orthorhombic crystal class symmetry that results in a single connected stress wave surface glyph: longitudinal and both transverse wave surfaces connect into a single surface [8]. In both cases the fully immersive experience of the CAVE can be used to study and analyze these structures and their corresponding material properties. Several publications are referenced here that show how the CAVE was used as an effective research tool [10-14].

3.2 Things that worked well

Except for the occasional server downtimes, the NPIB and JWave interfaces worked well for both the undergraduate and graduate CRCD classes.

Results from the undergraduate class, revealed that the most productive time spent using the CRCD modules was when students and instructors met in the SMVC on Fridays. Friday classes resembled lab sessions where students could ask questions and try out their ideas with comments from the professors who also helped interpret the simulation results. Instructors also received valuable feedback on how the JWave and NPIB forms were working and what needed to be improved. Friday sessions also built student confidence for successful completion of their homework assignments. Results from the graduate class revealed that significant improvements on the CRCD Web-server modules contributed to decreased use of the SMVC by the students.

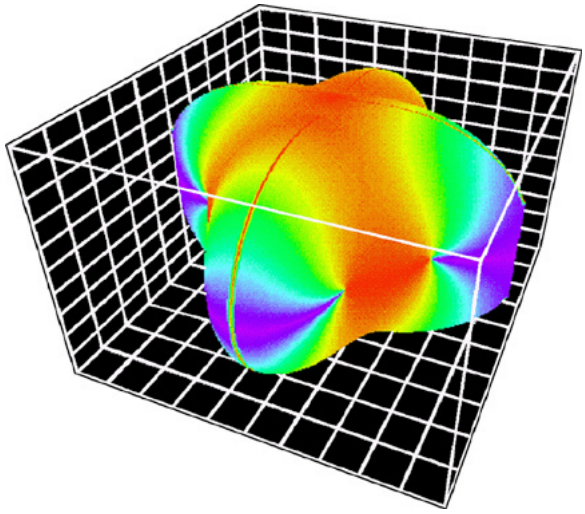

Figure 8. CAVE simulation view of a fourth-order stiffness tensor wave-surface glyph representation of a special crystal class symmetry for orthorhombic symmetry, where the longitudinal and both transverse wave surfaces are connected into a single surface [8] (micro-scale module04). For simplicity only the intermediate transverse wave surface is shown. The fully connected wave surface, not shown here, requires a fully immersive CAVE environment. Glyph data format is VRML 1.0 [9].

3.3 Things that need more work

For the first class, although the NPIB (version1.4) form worked well, the "builder" part of the NPIB was improved with more features; however it was not stable enough for the instructors to build their own forms. Consequently the technical support team members built all the NPIB version 1.4 forms using a scripting syntax that was difficult for the professors to learn. NPIB was upgraded to version 1.5

where the scripting language was replaced with a Java form builder. The Java form builder enables instructors, who are not Java literate, to build interactive NPIB forms. For the undergraduate class NPIB 1.4 forms only worked on the SMVC UNIX workstations with Netscape 4.5. Near the end of the semester we did get the NPIB 1.4 forms to work on Windows-NT Web browsers. This was largely due to the way Windows-NT handles screen refresh. For the first class the CAVE was located off campus and for the second class the CAVE was under reconstruction in the ACITC.

3.4 Lessons Learned

Java interface development is a difficult, if not impossible, for most engineering professors without backgrounds in computer science. Even using javascript with HTML is beyond the abilities of most professors. These same professors are also not experienced in routine systems administration needed for configuring and maintaining Web-servers. Hence there must be a commitment from the department or college to support a courseware Web-server and to train professors how to access and use systems such as the NPIB. Because of limited resources and reluctance to accept new technology, building and supporting courseware servers has been the most difficult aspect of this project. Because of these difficulties the CRCD Web-server was maintained by the Problem Solving Environment group by the Department of Computer Science[15].

In both classes we also discovered, based on previous experience, that programmers and system administrators need to work more closely. Typically all that was needed was to install a standard language compiler, with a users service group to answer any questions. With the advent of the network, professors and technical support staff can no longer afford to isolate themselves in the "new world" of computing where popular Web-based software applications are constantly changing. Successful projects now require that professors devote more time learning computing skills and how to work more closely in teams.

We also experienced first hand how Java needs to be maintained as a standard: early interfaces developed in Netscapes' IFC had to be rewritten.

Our experience in team-teaching this class was a rewarding but difficult task. In the first class more time was spent solving technical problems than was spent developing course content. This trend reversed in the second class because many of the modules were already developed and many of the technical problems solved. For the second class more modules were written for the "Micro" section.

To continue the learning experience, witnessed on Fridays in the first class that used the SMVC, students need access to the Java-Web server from outside the SMVC. Although convenient to manage, students should not be required to go to any particular workstation classroom environment. Some universities, because of security issues and convenience of management, prefer to isolate these resources from remote access. Such policies are counterproductive when every student is also required to own his/her own personal computer and where professors, who are located off-site, are expected to create courseware materials but cannot do so remotely. Improvements on the CRCD Web-server allowed students in the second class to work mostly from the CRCD Web-server. Remote access will aid in future development of these courses for distance learning.

3.5 Future Developments

Because the CAVE and SMVC were not fully operational, when we taught the second course, we hope to teach this courses again now that these resources are operational.

Since the second class was taught, there have been several improvements. The Java-Web server was upgraded to JWave 3.0, better security measures were implemented without restricting access to the anonymous ftp site, and NPIB was upgraded to version 1.6. Thus professors can build their own JWave and NPIB forms in their public_html directory. Other ideas for future development are: 1) validate and verify data before they are submitted to the server to check for data types: int, floats, etc., and 2) auto-generate NPIB forms given a specific input file (work with existing legacy data files).

Although our current Sparc10 Ultra server has been upgraded to 1GB of memory with 92GB of disk space, in the near future we will move the CRCD Web-server onto an SGI Origin 2000 server with 8 R10K CPUs, 8GB of memory, and 73GB disk space. With the current version of NPIB 1.6 we can now access remote site computers so that students can get their simulation results back quicker and submit larger simulations which will yield more realistic simulation results. With the NPIB 1.6 upgrade working we plan on accessing several new high performance computing systems at Virginia Tech: 1) Sun Enterprise 6500 computer with seventeen 400MHz (8MB cache) processors, 18GB memory, and 144GB of RAID disk, 2) SGI Origin 3400 rack with 8 R12K CPUs, 8GB of memory and 146GB of disk space, and 3) Beowulf cluster with 1GHz 200 CPUs and 1GB memory per CPU.

CRCD modules will be extensible to other classes taught in the Engineering Science and Mechanics (ESM) Department. We hope that this interest will grow to other ESM classes and that eventually the ESM Department will support their own JWave courseware Web-server.

Acknowledgements: Authors acknowledge the NSF grant "Combined Research and Curriculum Development: Computer Simulation of Material Behavior - From Atomistic to the Continuum Level" (EEC-9700815), and the foundation grant from SUN Microsystems Inc. and Visual Numerics Inc. to create the Scientific Modeling and Visualization Classroom of the ACITC.

4. References

1. Kriz, R.D., SUN-VNI Wave-Java Server: http://www.jwave.vt.edu.

2. Kriz, R.D., Scientific Visual Data Analysis and Multimedia: http://www.sv.vt.edu/classes/ESM4714/ESM4714.html.

3. Kriz, R.D. and Farkas, D. "Using Materials Resources on the World Wide Web for Introductory Materials Science Teaching," J. Materials Education, Vol. 19 No. (1&2), pp. 111-119, 1997.

4. Kriz, R.D., Farkas, D., and Batra, R.C., Computer Simulation of Behavior from the Atomistic to the Continuum Level: http://www.jwave.vt.edu/crcd/, 1997.

5. Kriz, R.D., Farkas, D., and Batra, R.C., "Integrating Simulation Research into Curriculum Modules on Mechanical Behavior for Materials: From the Atomistic to the Continuum", J. Materials Education, Vol. 21, No. (1&2), pp. 43-52, 1999.

6. Network Program Interface Builder (NPIB): http://www.jwave.vt.edu/npib/npib.html

7. SUN Microsystems Java Beans: http://java.sun.com/beans

8. Ledbetter, H.M. and Kriz, R.D., "Elastic-Wave Surfaces in Solids," Physica Status Solidi, Vol. 114, pp. 475-480, 1982.

9. Parikh, S.D., VRML 1.0 format necessary for viewing in both the CAVE and VRML Web-based viewer: http://www.sv.vt.edu/classes/vrml/exercise3.html.

10. Van Swygenhoven H, Farkas D, and Caro A, "Grain-boundary structures in polycrystalline metals at the nanoscal", *PHYS REV*, vol.62, p. 831, 2000

11. Farkas D, "Bulk and intergranular fracture behavior of NiAl" *MRS BULL*, vol.25, p. 35, 2000.

12. Farkas D, "Fracture mechanisms of symmetrical tilt grain boundaries", PHIL MAG LETT, vol.80, p. 229, 2000.

13. Farkas D, "Atomistic studies of intrinsic crack-tip plasticity", PHILOS MAG A, vol.80, p. 1425, 2000.

14. Van Swygenhoven H, Spaczer M, Farkas D, et al., "The role of grain size and the presence of low and high angle grain boundaries in the deformation mechanism of nanophase Ni: A molecular dynamics computer simulation", NANOSTRUCT MATER, vol.12, p. 323, 1999.

15. Shaffer, C., Problem Solving Environment: http://www.cs.vt.edu/~pse.

Table 1. CRCD Course Decomposition

Nano-scale	Micro-scale	Macro-scale
Lecture Topics: · Crystal bonding · Crystal structures · Crystal mechanical behavior · Dislocations · Fracture · Fracture at Interfaces	Lecture Topics: · Interface cracks · Anisotropy · Composite fiber-reinforce composites · Stress-free laminate edge problem · Laminate interface singularities · Laminate ply crack singularities · Cracks in homogeneous materials: · Isotropic · Anisotropic · Wave propagation: · Isotropic · Anisotropic	Lecture Topics: · Stress · Equilibrium · Strain · Material characterization · Boundary conditions · Work external forces · Minimum potential energy · Uniqueness theorem · Axial bar deformation · Beam bending terminal couples
Atomistic modules: · (01) Ni-Al grain boundary crack · (02) Vacancy in Iron	Heterogenous modules: · (01,02) Anisotropic polar plots · (03,04) Cijkl stiffness tensor glyph · (05,08) Linear elastic laminated plate analysis (LELPA) · Generalized plane strain edge stress laminate Finite Element Model · (06) Nonwoven $[0/\pm45/90]_s$ · (07) Woven $[0/90]_s$ · Generalized plane strain interior stress laminate Finite Element Model · (09) Without interior ply crack · (10) With interior ply crack · Singularity Stroh's method · (15a) stress free edge · (15b) laminate ply crack · Dynamic anisotropic mechanical behavior · (18) 1D FEM · (19) 2D FEM (30x60) mesh · (20) 2D FEM (45x180) mesh	Continuum modules: · (01) Stresses in thick walled cylinders · (02) Brittle-Ductile transition