Collaborative Toolkit for Crashworthiness Research

A. Bobrek, S. Simunovic, and G. Aramayo
Computer Science and Mathematics Division
Oak Ridge National Laboratory
simunovics@ornl.gov

Abstract

Ensuring the safety of vehicle's occupants during a crash is of foremost importance during the vehicle design process. Computer modeling of crashworthiness decreases the design cost and empowers the designers to explore new concepts with more confidence that their final product would be feasible and would perform to design specifications. To capture the complexity of vehicle deformations in computer simulations, detailed models are necessary and, in general, require full utilization of available supercomputing resources. Due to the limited availability of supercomputing resources, the development of collaborative methods for crashworthiness research would allow for effective utilization of the computational and human resources, as well as permit researchers with different backgrounds, interests, and roles in the design and production system to actively participate in crashworthiness research. Development of Collaborative Problem Solving Environments for the ongoing crashworthiness research at Oak Ridge National Laboratory is presented in this paper.

1. Introduction

Perhaps the most drastic automotive design verification comes from vehicle collision tests. Not only do they bring perspective to everyday driving but also make one more appreciative of challenges facing vehicle designers. The amounts of energy and deformation involved in collision are significant, and must be well understood to be harnessed into mechanisms that will protect vehicle occupants. In recent years considerable effort has been directed toward development of computational methodologies for simulating the mechanical response of automotive structures in collisions. It has become a standard practice in vehicle design to evaluate new and existing vehicles using computational simulations using large and complex models based on the Finite Element Method (FEM) [11].

Computational vehicle models have to capture complex deformation and interaction of vehicle parts and subsystems occurring during impact. The accuracy with which the crash behavior of the vehicle is simulated can be related in part to the density of the computational cells (finite elements) employed in the vehicle regions that experience significant permanent deformations. Therefore, a single vehicle crash model that would accurately simulate a wide range of impact conditions (frontal, side, offset, oblique impact, roll-over) would require a high density of finite elements over the entire vehicle. High finite element density is not the only requirement for an accurate model: the finite elements must join in a physically and computationally feasible way and materials properties of the vehicle must be approximated correctly. The problem then becomes one of generating a modeling environment that can be used for rapid generation and analysis of accurate crash situation-specific models. Creation of a modeling environment and other tools for generation of crashworthiness models is ongoing research at the Oak Ridge National Laboratory.

A Problem Solving Environment (PSE) is a set of tools dedicated to solving a target class of problems [1]. A PSE brings together computational resources, virtual libraries, visualization environments, databases, and expert systems in order to allow a scientist to concentrate on the science instead of the tools. To accomplish this task, a PSE needs to provide the expert assistance and software tools to the researcher in a user-friendly environment [2]. Recent developments in Internet networking technology have given rise to CPSEs that allow geographically separated researchers to share knowledge and ideas while solving a specific class of problems. In this paper, we will describe the development of the Collaborative Toolkit for Crashworthiness Research (CTCR), a CPSE that aims to meet the needs of crashworthiness model generation. The CTCR was created using well-established Internet technologies to reduce development time and to provide the user with a familiar user interface.

2. Requirements for crashworthiness CPSEs

When considering implementation of a Collaborative Problem Solving Environment (CPSE) in relation to crashworthiness, there are several important issues to be taken into account. Some of these issues are unique to the field of crashworthiness and material modeling, where some are common to all CPSEs. However, all these issues must be considered when designing a CPSE to be used in this area of research. Some of these considerations include:

1. Size and complexity of the crashworthiness models
2. Size and complexity of the data produced from the models
3. Need for visualization of the models, simulation results, and experimental data
4. Unifying computing and human resources
5. Meeting the needs of users with different backgrounds and skill levels.

The next four subsections will aim to address these issues and illustrate how they affect the design of CPSEs.

2.1 Size and complexity of the crashworthiness Models

Although crashworthiness tests have been conducted since 1930s, it was not until the 1970s and 1980s that the tests were analytically modeled using the FEM. Modeling beginnings were humble, simulating structural collapse of individual components whose properties were well known. As the computing power available to conduct these experiments increased, so did the complexity of the models. Where in 1988 a state-of-the- art FEM vehicle model would have about 10,000 elements, in 1998 a similar model would be composed of up to 160,000 elements. For the detailed description of the FEM, the reader is referred to a standard FEM textbook such as [11]. Vehicle components may be represented within the model by several different techniques. FEM modeling abstractions are assemblies of solid, shell, or beam finite elements. For example, a solid bar-shaped object such as a suspension link can be represented as one-dimensional object, i.e. beam. However, complex objects that are too irregular to be modeled as a one-dimensional beam, must be modeled as three-dimensional objects. This usually involves tediously generating a Computer Aided Drafting (CAD) representation of the object and then "shrink wrapping" a FEM around it. Most of the vehicle model is represented as an assembly of shells that are two-dimensional computational entities. Shell elements are used extensively because of their computational efficiency and because a large amount of the vehicle is made of sheet metal. Shells are generated much in the same way as solids: a CAD representation of a vehicle part must be first generated, and then FEM mesh projected onto the surface [4].

Properly applying these modeling techniques can significantly impact the performance of the model. For example, care must be taken that the objects do not penetrate each other because that would result in an incorrect FEM representation. Additionally, the individual parts must fit together in exact relation as in the real vehicle to simulate the crash time conditions. As a result, crashworthiness CPSE must be able to successfully handle models of high complexity and large size. Large model size makes sharing a full size model across current Internet bandwidths impractical. The collaborative aspect of a CPSE should allow for multiple users to be able to work on the vehicle model at the same time in order to share their expertise.

2.2 Size and Complexity of Data Produced from the Models

To monitor the vehicle's behavior during a crash test, many sensors are placed throughout the vehicle. This is true whether the test is conducted in a test facility or modeled computationally. Data from these sensors, such as acceleration, force, and displacement is essential to understand the behavior of the vehicle in a crash. In order to get an accurate idea of the events during a crash, a crashworthiness engineer may use 20 or more of such sensors per vehicle model. To complicate the picture, each model may be simulated using different material models, computational algorithms and crash test scenarios. This can lead to thousands of data sets that often need to be compared to each other. **Figure 1** illustrates the potential amount of data that a problem-solving environment has to deal with. The CPSE should be able to provide its users with a powerful and easy to use database system that can handle such large amount of diverse data. Accessibility is also an issue: the researchers without a computing background need to easily retrieve any combination of test data as well as view the deformation of the vehicle and its components as the crash progresses.

2.3 Need for Visualization of Models and Experimental Data

As seen above, large vehicle models for crashworthiness research can generate large amount of

data. Another problem that any proposed CPSE would face is the need to effectively visualize the model and the data generated from it. During an analysis of a crash, it is very important to properly view the state of the vehicle before, during, and after the crash. When visualizing a finite element based vehicle model, each shell finite element is generally represented by one polygon. Therefore, for a model with 160,000 finite elements, the computer will need to render 160,000 polygons. If a user wishes to smoothly move or rotate this model, the

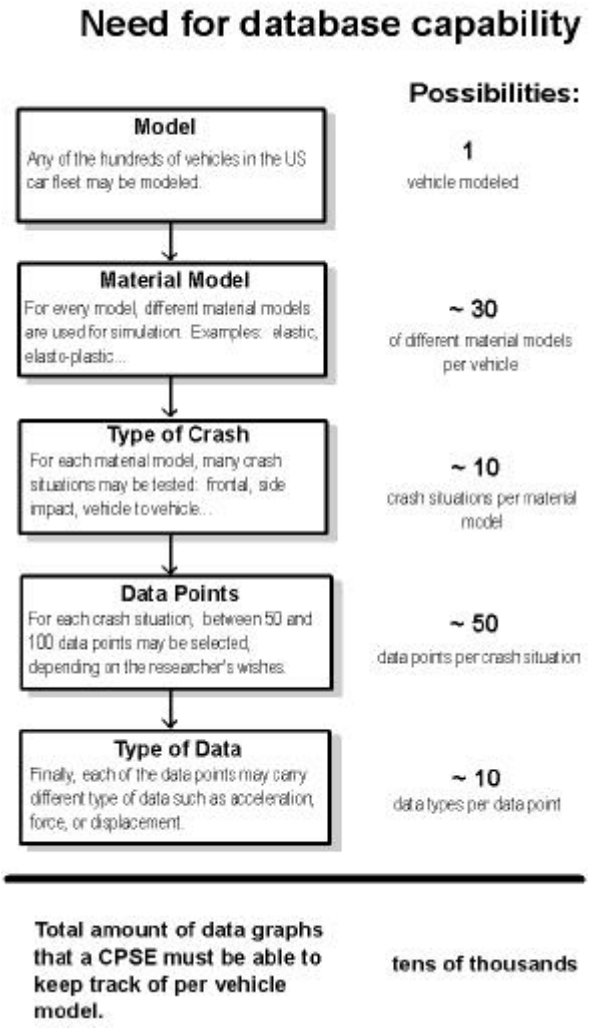


Figure 1. Amount of data potentially generated by crashworthiness research

computer will need to render about 20 frames per second or 3.2 million polygons per second. By rendering both sides of every polygon, this number would double again. It is easy to see how a crashworthiness model may present a significant drain on a workstation's rendering resources.

The data generated by the model also presents visualization challenges. Much of the data can depend on multiple factors requiring graphing in three or more dimensions. Additionally, the data for the simulation is stored in time intervals of 1ms over a duration of 100-150ms. In the case of displacement data, this would create up to 150 different representations of a vehicle as it proceeds through the crash simulation. Providing such support for every user regardless of what his or her computing platform is would be one of the major challenges for any problem solving environment.

2.4 Unifying computer and human resources

Due to their complexity, crashworthiness models require significant computing resources for development and simulation. Resources necessary for crashworthiness research as well as the researchers themselves are geographically separated. To allow researchers to pursue crashworthiness research from different geographic locations, a crashworthiness problem-solving environment must take advantage of the computer networking superstructure already in place (Internet or Internet2). A networked C`PSE will also have the effect of allowing researchers with different backgrounds and interests to participate in research, as well as allowing the industry leaders to easily collaborate on their accomplishments. Additionally, using the already established networking infrastructure would permit access to modeling data from a testing laboratory or a real crash site. A CPSE aimed at crashworthiness research should allow the researcher to access the computing resources and data he or she needs remotely.

3. Collaborative Toolkit for Crashworthiness Research

Collaborative Toolkit for Crashworthiness Research (CTCR) is a problem solving environment that aims to provide a researcher with a set of tools to solve crashworthiness specific problems. CTCR uses a web-based interface to bring together many useful tools for crashworthiness modeling. The web-based interface provides for easy remote access as well as for collaboration between researchers using well established protocols and familiar user interfaces. Functions of the CTCR can be divided into five separate areas:

1. Visualization of the model
2. Visualization of the simulation and test results
3. Collaboration and record keeping
4. Building Modeling Data
5. User Interface

Separate modules deal with each of the above needs. The web interface used already established Internet technologies for file transfer, security, visualization and collaboration allowing the design team to concentrate on developing the problem solving environment instead of the supporting technologies. Technologies like Virtual Reality Markup Language (VRML) [5] and Java3D [14] allowed for visualization of the models and the three-dimensional test results using the Internet. The Common Gateway Interface (CGI) [15] used in conjunction with Perl programming language provided for the communication between the user and the data server. Flash technology from Macromedia [16] allowed for test results to be requested by the user in a clear and attractive manner. Finally, Java and specifically its Swing GUI package [17], allowed for convenient internet-based user interface to be developed for the modeling data builder. All of these technologies are tied together using Hyper Text Markup Language (HTML) in a convenient and easy to use web page. Using such widespread technologies allow for the user to interact with the CTCR using a standard Internet browser. Additionally, this type of structure allows for mirroring of the CTCR web site to allow handling large number of users.

In the following sections, each of the modules will be discussed in detail, as well as their role in the CTCR and how they meet the needs of crashworthiness research.

3.1 VRML Model Visualization Module

Because the researcher has to work with a detailed vehicle model, it is necessary to provide him or her with a good visualization tool. CTCR's Model Visualization Module visualizes the model, empowering the user to gather information and collaborate on the model. Additionally, CTCR's Model Visualization Module allows for easy access of other tools within the CTCR package.

To effectively view the vehicle models through a network infrastructure, CTCR uses the Virtual Reality Modeling Language (VRML). VRML is a file format for describing interactive 3D objects and worlds [5]. VRML is especially well suited for a distributed environment like the World Wide Web by allowing hyperlinking to other files and supporting other Internet file formats. Due to VRML's ability to interact with the user, the network environment, and other tools within the CTCR, it was a good choice for CTCR's Visualization Module. The visualization tool had to provide certain basic services to the user through the VRML interface. User should be able to:

1. View the model remotely
2. Effectively gather information about the model
3. Provide feedback on the model

VRML, with the assistance of other web-based technologies such as JavaScript [8] and HTML was able to cover these requirements in the following ways:

3.1.1 Viewing the model. A VRML file format is a plain text format that describes the shapes and their properties within a 3D world. This file only specifies objects and shapes within a virtual environment; it does not handle any of the navigation. Moving through the world, rotating objects, and similar functions are handled through a VRML player, an application that displays the VRML file. Therefore, to represent a vehicle model in VRML, CTCR only has to include objects to be viewed and not an implementation to view them. Each object (or vehicle part in this case) is described in a node, an encapsulated entity containing information that defines that object. To define object shape, color, position, and other properties, other nodes containing that information are added to the original object node. For example, to create a sphere in a VRML world, user would first create a parent node named "Sphere". Then he or she would attach a shape ("child") node that contains sphere's geometrical information. To complete the task, other children can be attached specifying color, material, position, or rotation. However, the real strength of VRML lies in its ability to attach behavior nodes to objects. Using JavaScript, another well-established web-based technology, it is easy to describe simple behaviors for every object. For example, the sphere mentioned earlier can be made to light up every time it receives focus from a mouse pointer or to bring up a specific web page when clicked upon.

Vehicle geometry is converted into a VRML file by a Perl script that sorts the parts, assigns them positions and rotations, colors them, and assigns them appropriate behaviors. Due to this automated process it is possible to quickly export new versions of the vehicle model to VRML format and post them for viewing and collaboration on the web. The vehicle can then be viewed by VRML players such as Cosmo Software's Cosmo Player [12] or Sony's Community Place from a multitude of Internet browsers and computing platforms.

3.1.2 Gathering information about the model. In order to add interactivity to the model, CTCR creates a virtual control panel that constantly follows the user inside the VRML world. The panel allows the user to select individual vehicle parts, examine their mesh makeup,

move, rotate, and eventually call up an information page about the part. The "explode" feature separates the individual parts, allowing the user to get a good idea about the interconnectivity of the vehicle. The name of the part and the subsystem it belongs to are also displayed in the control panel **(Figure 2).**

By using this control panel, the user can easily navigate through the vehicle and gather information about it. The delete function allows the user to remove any parts that are obstructing the area of interest. This is a common occurrence, especially if user wants to examine a cramped area, such as the engine compartment. Additionally, the user is able to examine a part's mesh composition by selecting it. Best of all, these behaviors are easy to add, delete, or alter, due to the object-oriented design of the VRML interface.

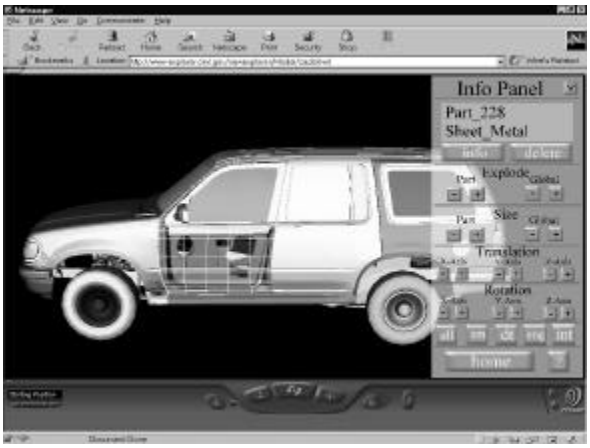


Figure 2. VRML control panel interface within Cosmo Player and Netscape 4.7

3.1.3 Providing feedback on the model. By selecting a part and the "info" button on the control panel, the user will bring up an information page about that specific vehicle part. Here, one will have an opportunity to read comments of other researchers about the part, or even enter one. This type of interface has several advantages. First, it allows the comments to be sorted by the area of interest. Additionally, by linking VRML to an HTML page, all the other tools that make up CTCR become available. In other words, the toolkit will provide the appropriate tools when a user needs them.

For example, the user may select one of the vehicle fenders, and bring up an information page about it. Here, user can read the comments left by his colleagues about the finite element model of the part and make some comments of his own. He will be provided with a link to the image server, in case he wishes to view the actual photographs of the part. Or he may bring up a graph containing the forces acting on the part during a crash. He may decide that this particular part's response is not satisfactory due to the selected material modeling technology and decide to pick another one using the Material Model Builder. CTCR can increase productivity by placing all the other tools in the package within the easy reach of the researcher.

3.2 Visualization of Simulation and Test Results

Crashworthiness research generates a large amount of data that a researcher needs to analyze. This data can be classified into two areas: vehicle deformation data, and the data collected by sensors placed throughout the vehicle. Vehicle deformation is viewed and analyzed through a multitude of crash movies. These movies are sorted and served so that the user may easily access and compare them. Sensor data is handled by the Graph Server Module (GSM). With GSM, CTCR provides the researcher with a tool that can access crashworthiness graph data in an easy-to-use and flexible fashion. The GSM is based on two internet technologies: Flash and CGI scripting.

3.2.1 Macromedia Flash technology within the GSM. Macromedia's Flash technology has been on the market since 1995, but has only recently (with release of Version 4 in 1999) been expanded to allow the amount of interactivity necessary to be a part of the CTCR. Flash is authoring software for creating scalable and interactive animation for the web [6]. It can be used to create everything from animated logos to complex Internet commerce applications. Flash's vector graphics make intricate animations compact, reducing transfer times over the network. Complex behavior can be incorporated into Flash applications, making it a perfect choice for easy and quick GUI on the web. Additionally, Flash can communicate with CGI, allowing server-side scripts to be incorporated with Flash applications. To operate, Flash requires a free browser plug-in that comes bundled with some versions of Netscape Communicator and Microsoft Internet Explorer.

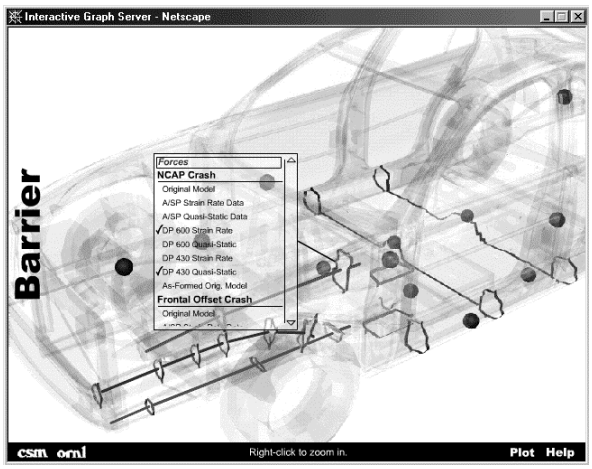


Figure 3. Flash implementation of the Graph

To serve crashworthiness data graphs, Flash application displays a rendering of the vehicle with each data point clearly marked **(Figure 3)**. If a user moves the mouse pointer over the data point, he or she will receive information about that data point. If a user selects the data point, then a menu with all the data sets attached to that particular data point is shown. Within the menu, data sets are sorted by the type of test they represent or by the material model or simulation technique that was used when generating the data. The user can then proceed to select the data set(s) he or she wishes to graph and plot them. This type of arrangement allows the user to view the data graphs for each data point as well as graphing multiple graphs onto one plotting area so that performances of different parts, material models, or simulation techniques can be compared.

The data served by the GSM is also very flexible and easily revisable. The Flash application has been designed to use a set of templates detailing which data sets should be displayed in the menu. Updates and changes to the system can be accomplished by template modification.

3.2.2 Common Gateway Interface (CGI) and server side scripting. Once the user decides which data sets to plot, the request is passed from the Flash interface through the Common Gateway Interface (CGI) to the server. This request is then processed by the Perl script, which fetches the desired data sets from the database and creates the desired plot **(Figure 4)**.

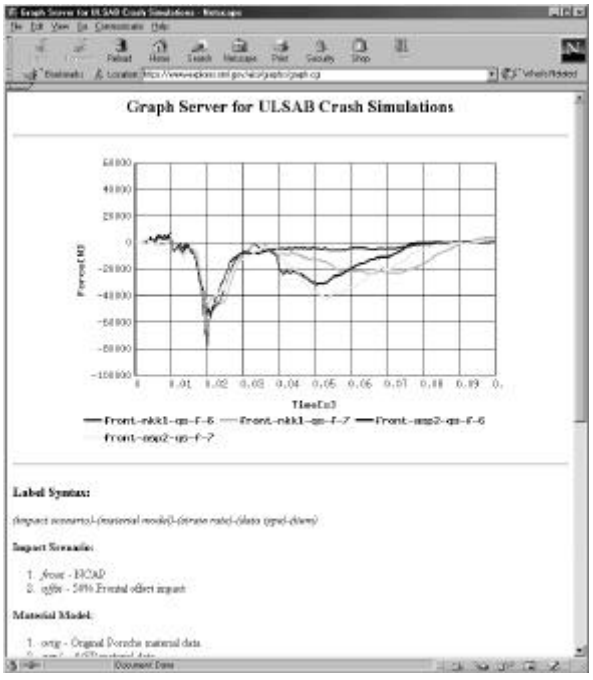


Figure 4. Plot generated by GSM's server side Perl script

Both CGI and Perl have been used to add interactivity to the World Wide Web since its early days. CGI, which is the gateway between a user's browser and any program running on the server, has been most commonly used for applications like counters and guest books on web pages. However, Perl, which runs a script on the server side of the CGI, is a very powerful programming environment. Because the libraries for generating image files and data graphs were already developed, GSM was fairly easy to implement. This met our goal to combine existing technologies providing a practical environment for collaboration on crashworthiness projects.

3.3 Electronic Notebook

Documentation is an essential component of every research endeavor. Due to the collaborative and web-based nature of CTCR, one of its essential components is an "electronic notebook", a tool which allows scientists to share their ideas, data, and events of their joint research. To fulfill this need, we chose to use the Oak Ridge National Laboratory (ORNL) Electronic Notebook, developed as a part of DOE 2000 Electronic Notebook Project.

The electronic notebook was developed in 1996 by Al Geist and Noel Nachtigal of the Oak Ridge National Laboratory [13]. It is now used by more than 100 research groups all over the world. By using CGI scripting, JavaScript and Java applets, the electronic notebook is an equivalent of a paper notebook allowing the researcher to write to its pages, add tables and sketches, and (digitally) sign and date all entries **(Figure 5)**. The electronic notebook is also more capable than its paper equivalent: it can store audio and video files as well as spreadsheet data. Other advantages of the Electronic Notebook include ease of use, security, and expandability.

The ORNL Electronic Notebook is designed to be used from a regular Internet browser which made it easy to integrate with CTCR. Unlike its paper equivalent, the electronic notebook can be searched with its built-in search engine. Sketches can be scanned in and imported as an image file, or made quickly by using the Java applet that allows sketching **(Figure 6)**.

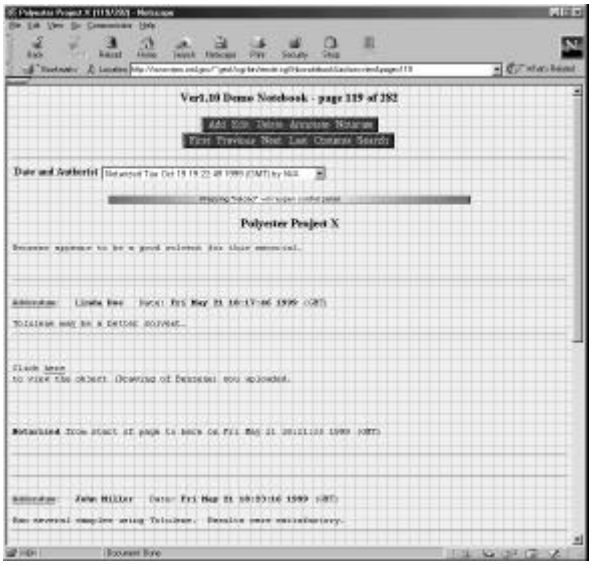


Figure 5. Example of a ORNL Electronic Notebook page

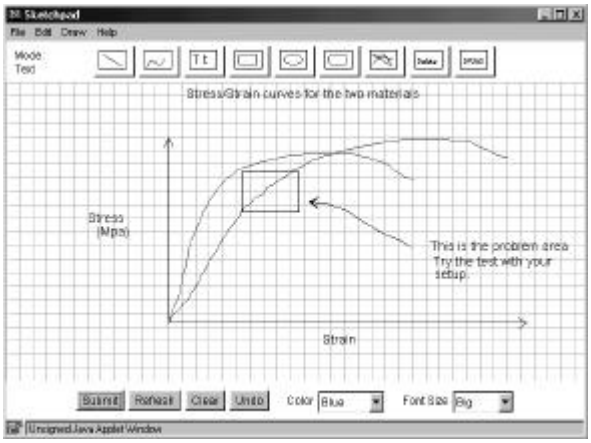


Figure 6. ORNL Electronic Notebook Java sketchpad

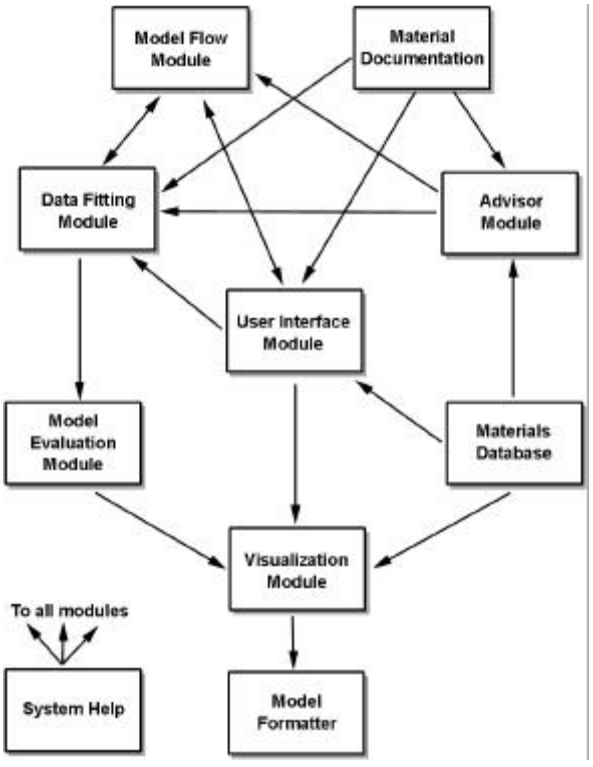
The notebook integrates several security mechanisms. It is possible to digitally sign (notarize) the notebook pages, making sure that no one can alter the information after it is entered. A password access can be specified for different groups of users. For example, some users may be allowed only to enter new information, where others could edit or delete pages. Data security on the network can be assured with SSL encryption, a feature supported by many web browsers.

By changing the MIME types of the files that the notebook can accept, the user can expand the ORNL Electronic Notebook to fit any uses. Additionally, third party plug-ins can be used within the Notebook to further expand its functionality.

The ORNL Electronic Notebook was included in the CTCR to provide a medium for discussion between the researchers who are often separated geographically. HTML make-up of the notebook allowed for easy integration into CTCR, putting it within easy reach of the researcher while he or she is using any of the other tools within the CTCR.

3.4 Material Model Builder

Even the FEM models with large numbers of elements will not be able to produce realistic results without a realistically modeled material behavior. The understanding of material behavior is far from complete. Even widely-used engineering materials (e.g., mild steel) lack reliable data for crashworthiness modeling. The CTCR should contain a set of tools that guide the user through a material models selection process, helping select the model that is most appropriate for the current situation. The CTCR should also provide the user with a database of commonly used materials models. Here the collaborative aspect of the CPSE is of extreme significance: researchers could share material models that have worked for them in the past, and thus save on development time. CTCR addresses above needs with its



Material Model Builder (MMB).

The objective of MMB is to provide interactive, user-friendly procedures for development of computational material models for engineering simulations. Addition of MMB to CTCR allows engineers with less experience in materials modeling to be as effective in crashworthiness research as those with more experience in the field. MMB does not need to be applied to crashworthiness only, it can be used in any field where selection of an appropriate model for a material is necessary. MMB can be divided into several building blocks:

1. User interface module
2. Data fitting module
3. Model evaluation module
4. Visualization module
5. Model flow chart builder
6. Advisor module
7. Model formatter
8. Materials database
9. Material documentation module
10. System Help

Their integration in the system is shown in **Figure 7**.

The *Advisor Module* is the key to guiding the user through a material model selection process. The purpose of the *Advisor Module* is to provide guidelines for the specification of required data, model types, and sequence of procedures that must be followed for material model development. The Advisor defines material-option flow charts (the *Model Flow Module*) and leads the user through series of procedures in order to develop a material model that suits the user's requirements. By pressing on the boxes of the flowchart, the user is taken to the model building subroutine associated with the box. Flow chart representation also visualizes the entire process in the user's mind, helping him or her to understand the formal theory behind the procedure, as well as providing a guide for the entire material model building process and thus reducing potential mistakes. The flowchart can be as long as the most complex material model available. However, the user can end the model building at the level of complexity suited for the simulation.

Throughout the material model selection process, the *Data Fitting Module* fits the information acquired from the user to the material model formulation. The main data fitting technique is based on Levenberg-Marquardt method [10], which has become de-facto standard of non-linear least square data fitting routines. The Model Evaluation Module uses measured levels from the fitting algorithms to evaluate the data fit to the specific formulation. Where appropriate, confidence intervals based on predetermined or user specified confidence levels are calculated and presented to the user in a graphical form so that a decision can be made on potential modification or restriction of data sets in order to develop a robust model estimate. In order to further increase user confidence in the model, data from the *Material Database* is continuously presented and updated so that the user can compare their models with characteristic properties of similar materials. The *Material Documentation Module* serves dynamic hypertext documents built from its own database that provide background information about the material and material model formulations, as well as hyperlinks and references to the key literature on the subject.

The *Visualization Module* continuously performs the visualization of the material model as the model building progresses. Visualization of the data sets, data fits, model formulations, error distributions, confidence intervals, and other entities for 1 dimension is performed in a bar graph form (picture), 2D data is displayed in a line graph form (picture), 3D data is presented as a surface (using VRML or Java3D), and higher dimensional data is reduced to sets of lower dimensional objects as appropriate following the standard procedure in the literature and engineering practice. The continuously updated visualization helps the user to understand the nature and the effect of the parameters as they are being added following the *Advisor Module's* procedure. Additionally, on-the-fly model visualization will illustrate the gains and drawbacks stemming from adding complexity to the model and assist the user in making a decision about when is the model "good enough". After a satisfactory model is reached, the user can use the *Model Formatter* to export the model into common formats used by simulation software.

These modules are tied together through the *User Interface Module*. Access to the information stored within the system can be easily accessible in a way that seems natural to the users. This module must recognize that there a numerous ways that a user would want to use the system. The data input can be verified by the system, and inconsistencies in the trends and values, confidence intervals, and data scatter characteristics, signaled to the user. Since Java is easily distributed, secure over networks, compatible with multiple platforms, and contains good GUI capabilities, it presents a good programming environment for the MMB. [9] Areas of the MMB which require performance that Java can not provide (such as the *Data Fitting Module*), are implemented in C and can be integrated into Java using the Java Native Interface (JNI). Finally, the object oriented implementation of Java allows for additional modules to easily be added as they become available.

4. Conclusions

CTCR has been developed over a period of two years with constant communication between the users and the CTCR designers. It has been used by mechanical engineers, design engineers, material scientists, program managers, and others whose comments and suggestions resulted in creation of a CPSE that can cater to the needs of users with different backgrounds and levels of knowledge. Because CTCR was used by variety of users, it was possible to identify problems within the modules early, and correct them while CTCR was still in development. Other lessons learned in the design of the CTCR can also be easily extended to other CPSE systems. CTCR's use of existing Internet technologies is a good example. By using well-established Internet technologies such as HTTP, JavaScript, or CGI, complex CPSEs can be built quickly and economically. These technologies also increase expandability of a software package because new tools for the Internet are becoming readily available. CTCR serves as a good example of how multiple tools, which differ greatly in implementation, can be brought together seamlessly using an ordinary Internet browser. Because most users are already familiar with how to use their Web browsers, using these technologies can decrease the learning curve users must face when using a new CPSE. Modules of the CTCR can be accessed through the web pages of the Computational Materials Science Group, Oak Ridge National Laboratory located at http://www-cms.ornl.gov. The problem of crashworthiness modeling environments is still in early stages of being solved. Although CTCR presents a significant help to a crashworthiness researcher, there are other improvements that can increase effectiveness. As computing power increases and becomes more accessible, designers may be able to generate models and get the results within a few minutes. Today it takes many hours just to simulate deformation of one subsystem in detail. Additionally, as Internet bandwidths and supporting technology continue to evolve, many other tools will become more commonly used within CPSEs. Collaboration technologies can be expected to break the geographical boundaries between researchers, making collaborative long-distance research practical and commonplace.

5. Acknowledgements

This research sponsored by the U.S. Department of Energy, Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Transportation Technologies, Lightweight Materials Program and by the American Iron and Steel Institute under Project Number ERD-97-XM001 with the U.S. Department of Energy. Work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under contract DE-AC05-00OR22725.

6. References

[1] E. Gallopoulos, E.N. Houstis, and J.R. Rice, "Computer as Thinker/Doer: Problem-Solving Environments for Computational Science," *IEEE Computational Science & Eng.*, Vol 1, No.2, Summer 1994, pp. 11-23.

[2] D. W. Walker, M. Li, O. F. Rana, M. S. Shields, and Y. Huang. "The software architecture of a distributed problem-solving environment." Technical Report TM/ORNL-1999/321, Oak Ridge National Laboratory, Oak Ridge TN 37831, USA. February 2000.

[3] A. Paluszny, "State-of-the-art review of automobile structural crashworthiness." Technical Report, American Iron and Steel Institute, June 1992.

[4] J.G. Thacker, S.W. Reagan, J.A. Pellettiere, W.D. Pilkey, J.R. Crandall, and E.M Sieveka. "Experiences during development of a dynamic crash response automobile model." *Finite Elements in Analysis and Desig*n, 30:279-295, 1998.

[5] The VRML Consortium. VRML International Standard ISO/IEC 14772-1:1997. Available at http://www.vrml.org.

[6] A. Ames, D. Nadeau, and J. Moreland. VRML 2.0 Sourcebook, Second Edition. John Wiley & Sons, Inc., New York, 1997.

[7] D. Emberton and J. Hamlin. Flash 4 Magic. New Riders, Indianapolis, 2000.

[8] Y. Shiran and T. Shiran. Advanced JavaScript Programming. Wordware Publishing, Plano, TX, 1998.

[9] D. Flanagan. Java in a Nutshell, Second Edition. O'Reilly & Associates, Inc., Sebastopol, CA, 1997.

[10] W. Press, S. Teukolsky, W. Vetterling, B. Flannery. Numerical Recipes in C: The Art of Scientific Computing, Second Edition. Cambridge University Press, 1993.

[11] T. Hughes. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Prentice-Hall, Inc., Englewood Cliffs, 1987.

[12] Silicon Graphics, Inc. Cosmo Player 2.1 Release Notes. Mountain View, CA, May 1998. Available at http://www.cosmosoftware.com

[13] Oak Ridge National Laboratory. DOE 2000 Electronic Notebook Project. Oak Ridge, TN, 1996. Available at http://www.epm.ornl.gov/enote

[14] Sun Microsystems, Inc. Java 3D 1.1 API Specification. Mountain View, CA, December 1998. Available at http://www.javasoft.com/products/java-media/3D/index.html

[15] National Center for Supercomputing Applications. The Common Gateway Interface. University of Illinois, Urbana-Champaign. Available at: http://hoohoo.ncsa.uiuc.edu/cgi

[16] Macromedia, Inc. Flash Software Documentation. Available at: http://www.macromedia.com/flash.[17] Sun Microsystems, Inc. Java 1.2.2 API Specification. Mountain View, CA, October, 1999.