2. *CGI Programming on the World-Wide Web* by Shishir Gunduvaram.
3. *Metalworking Simulation and Formability Using Finite Element Method* by J.P. Bandstra, presented at the SME AutoFact conference, November 1990.

5.0 Conclusion

This paper presented different techniques for integrating manufacturing process simulations to the Web. These techniques are applicable to many different kinds of simulations in addition to manufacturing processes. This paper discussed an implementation of a Web-based deformation process simulation, illustrating that performing manufacturing process simulations over the Web is possible. Nevertheless, several problems hinder performing large-scale simulations over the Web:

- VRML is fairly new and is a maturing technology.
- Presently, software tools for VRML development are limited.
- VRML, Java , JavaScript, and ActiveX need better integration.
- Distributed object technologies such as CORBA, IIOP, and Microsoft's Distributed Component Object Model (DCOM) need to mature.
- The software development tools for distributed object technologies need to improve.
- Network bandwidth needs to improve to support simulations over the Web.

In time, these problems will be resolved, and Web-based manufacturing process simulations will become commonplace.

Acknowledgments

The authors would like to acknowledge Murali Thirukkonda for helping to define the WebFrac simulation and Professor Jim Bandstra of the University of Pittsburgh at Johnstown for developing the ductile fracture simulation software.

References

1. *Java in a Nutshell* by David Flanagan.

- View nodal connectivity information
- Control the display parameters.

3.4 Implementation Discussion

As outlined in the scenario, the Java sockets method was used to implement visualization. A drawback with this method is its use of a fat server/thin client approach. Every time the user wants to visualize something different, the request is sent to the server and processed. Over the Internet, this process can be slow depending on network traffic and the number of simultaneous clients handled by the server. A faster implementation would have been to load the simulation results data into the applet on the client side. The applet could then process the user request, and create visualization information. For this approach, an API is required to create 3D visualization information. Such an API was not available to create VRML-formatted data during development of this project. VRML is fairly new, and VRML creators are actively involved in specifying APIs and external authoring interfaces for creating VRML content from external sources using scripts and applets. Software development tools and APIs will soon be available for the dynamic creation of VRML content.

This simulation could have been better implemented using IIOP instead of sockets. At the time of implementation, however, IIOP was just emerging and no tools were available. Therefore, the approach used was the best available solution. The WebFrac simulation will be made available to process modelers from the following website "http://www.ncemt.ctc.com"

4.0 Future Work

We will extend the deformation process simulation and defect prediction to support micro-structural parameter prediction. In addition, we will integrate other process simulations, e.g. casting and powder consolidation, with the Web. A simulation service will be established at the *CTC* Web site. This simulation service will enable the manufacturing industry to perform process simulations over the Web.

the WebFrac Web site. An initial HTML page will allow the user to upload an FEM simulation output file to the Web server and specify other input parameters.

2. The FEM results file is uploaded, via HTTP, to the Web server using a CGI script written in Perl.
3. The CGI script prepares an input file for the ductile fracture prediction.
4. The CGI script invokes the ductile fracture prediction, which is a stand-alone program.
5. DuctFrac performs ductile fracture prediction analysis on the FEM simulation output data and calculates defect parameters for all nodal points using several different defect criteria. This simulation is repeated for all the time steps specified by the user.
6. DuctFrac outputs simulation results to a file.
7. The CGI script prepares an HTML page housing the applet, which displays a GUI for the user to perform interactive visualization.
8. The CGI script sends the HTML page to the client.
9. Browser loads applet and displays the GUI to the user.
10. Browser requests socket connection with the application server.
11. Application server spawns a thread, which acts as the server for this client. The thread then loads simulation data from a file.
12. Application server thread indicates to browser that it is ready to serve the user request for visualization.
13. Applet tracks user input.
14. Applet sends request to application server through the socket.
15. Application server thread creates a VRML file 'on the fly' based on the user request.
16. Application server thread sends VRML file to the Web browser.
17. Web browser displays VRML file to the user.

3.3 Functional Capabilities

WebFrac allows the user to:

- View an animation of a deformation process simulation and defect prediction,
- View simulation results for a desired time step,

- **Significant Cost Reduction:** Because the simulation is platform-independent, special hardware is not needed to gain access to the simulation.

3.2 Scenario

Figure 5 illustrates a typical scenario for using the WebFrac simulation, with an explanation of each step following.

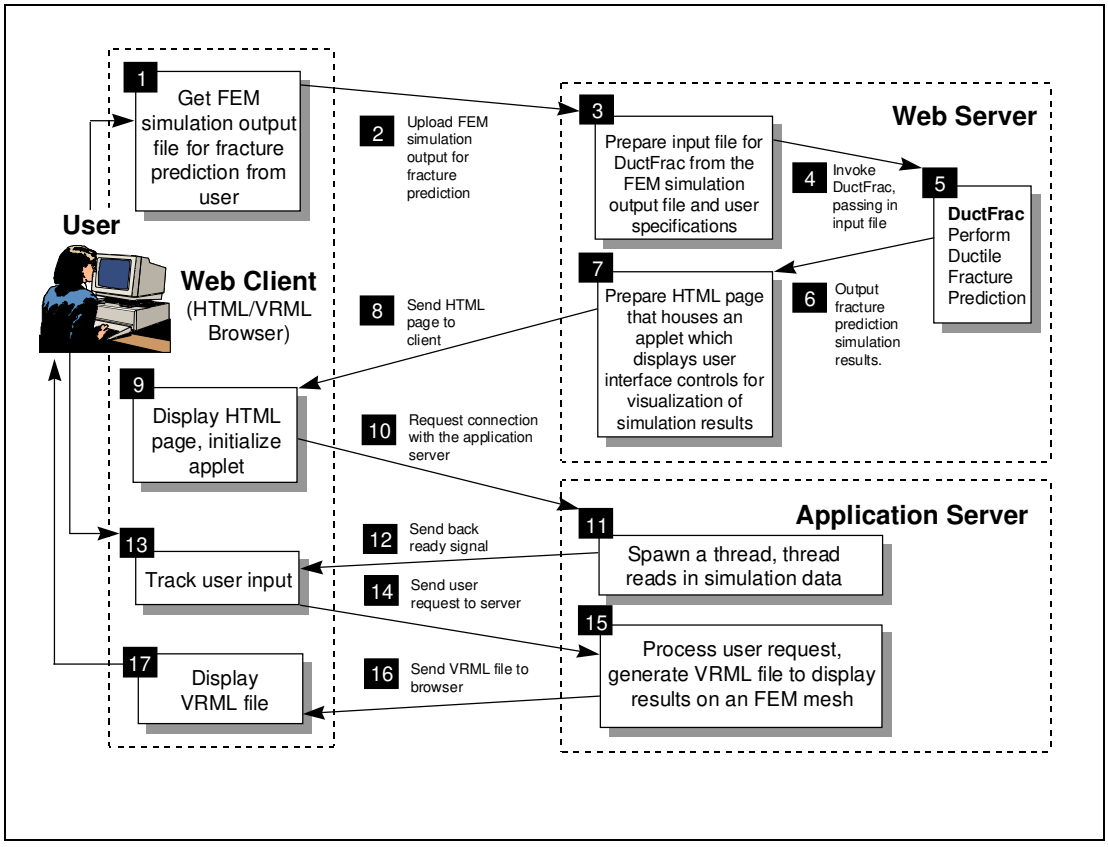

Figure 5: Event Flow Diagram for the WebFrac Simulation

1. A process modeler, with a need to solve a deformation process defect problem, would perform the FEM simulation using the tool of choice. An FEM simulation output file essentially consists of stress, strain rate, strain, and temperature predictions at various nodal points in the FEM model, with the data set repeating for various simulation time steps. To perform a ductile fracture prediction, the user would go to

3.0 Web-Based Deformation Process Simulation

This section discusses the implementation of a Web-based deformation process simulation with defect prediction capabilities, known as WebFrac. Ductile fracture occurs during a metal-forming process such as forging, wire drawing, and sheet metal rolling and is a major limitation in implementing such processes for modern, hard-to-work alloys. As an aid in design and contributing to solving a variety of manufacturing-related problems, the Finite Element Method (FEM) has been a tremendous asset to the engineering and manufacturing communities. [3] Most of the FEM software packages currently available for simulating deformation processes, however, do not support ductile fracture simulation. To address this need, a ductile fracture prediction simulation (called DuctFrac) was developed along with a visualization tool for easy interpretation of the results.

Both the simulation and visualization tools were developed as stand-alone applications running on Silicon Graphics platforms. These tools have three limitations: they are platform-dependent, porting to other platforms is a significant task, and they cannot be accessed remotely. The user needs to install or upgrade the software on-site, making maintenance cumbersome. These problems can be overcome by providing a Web interface to the simulation. WebFrac is an integration of DuctFrac to the Web. For process modelers, it is a highly interactive, remote-access, platform-independent tool for predicting ductile fracture in metalworking processes.

3.1 Rationale

The Web interface to the deformation process simulation and defect prediction tool provides the following benefits:

- **Remote access:** This simulation enables process modelers to perform deformation process simulations over the Internet from any location.
- **Platform Independence:** Because the GUI for this simulation is a Web browser, the simulation can be run from any platform that a Web browser can run on.

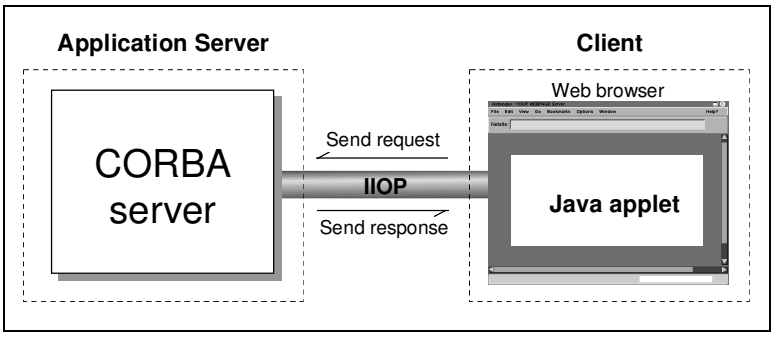

Figure 3: Client/Server Architecture for Manufacturing Process Simulations Based on IIOP

CORBA and IIOP provide a scaleable, distributed component architecture that enables the development of distributed manufacturing process simulations as shown in Figure 4.

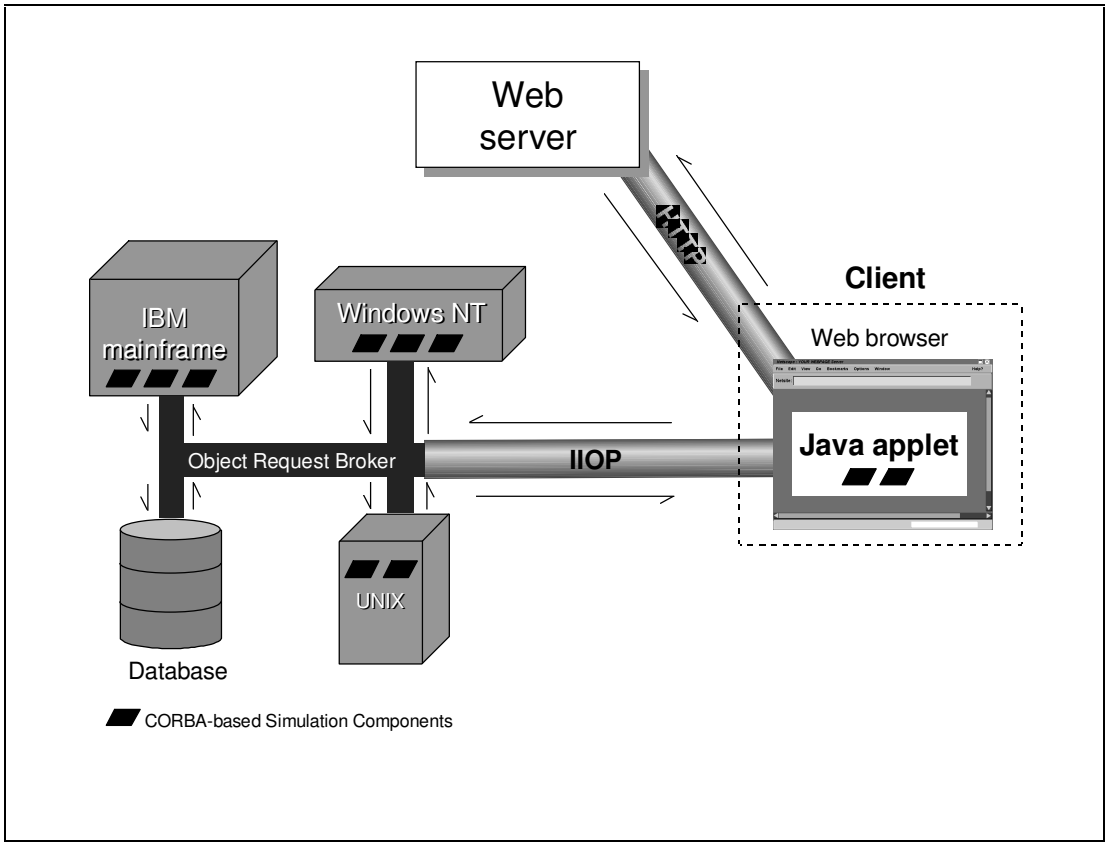

Figure 4: Distributed System Architecture for Manufacturing Process Simulations Based on IIOP

The method becomes cumbersome, however, when developing n-tier distributed simulations because sockets require the client and server to engage in application-level protocols to encode and decode messages for exchange, and the design of such protocols is cumbersome and error-prone. An additional restriction with this method is that the client and the server need to be implemented in Java.

2.3 Method 3: Internet Inter-ORB Protocol (IIOP)

In the third method (Figure 3) for developing Web-based simulations, the client hosts a Java applet that is connected to the application server through the Internet Inter Orb Protocol (IIOP). IIOP is based on the Common Object Request Broker Architecture (CORBA) and is implemented based on TCP/IP. CORBA is a distributed object computing specification developed by the Object Management Group (OMG), a software consortium with over 700 members. A Java applet embedded in an HTML page can be connected to a CORBA-based application server through IIOP. This method is the most suitable one for developing Web-based simulations for several reasons:

- CORBA-based IIOP provides language transparency, which enables development of the server-side software using the language of choice. This also allows for use of legacy code, in which many companies have invested considerable time, money, and effort to develop.
- CORBA and IIOP provide location transparency. The network-level programming details are hidden from the user. Objects are invoked by other objects in exactly the same manner, whether they are running locally or on a remote host. This enables tighter integration of client and server software.
- The roles of client and server can be interchanged. This provides a peer-to-peer architecture that scales well to support development of distributed simulations.
- CORBA provides a rich set of services that facilitate the development of Web-based distributed simulations.

2.2 Method 2: Java Sockets

In the second method (Figure 2) for developing Web-based simulations, the client, hosted by the Web browser, is a Java applet that resides in an HTML page. The applet is connected to the application server through a socket. The server is modeled as a perpetual process that waits for clients to connect (request a service). On each client connection, the server creates a new thread to handle the client's request. The server then goes back to waiting for new client requests. The threads approach is well suited for handling multiple clients because it prevents one client request from holding up others if that request enters a wait state or is performing a time-consuming task.

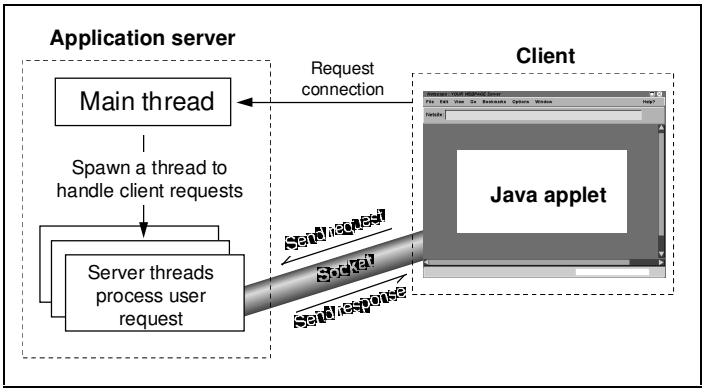

Figure 2: Client/Server Architecture for Manufacturing Process Simulations Based on Java Sockets

The Java Application Programming Interface (API) has a set of libraries for initiating and communicating over TCP/IP sockets. Previously UNIX programmers often resorted to writing C or C++ wrappers over the existing Berkeley Software Distribution (BSD) sockets API. The standard UNIX API provides a low-level interface and requires significant error checking and handling to be done by the programmer. It is also very architecture-specific. Java, with its "java.net.package," has removed a majority of the portability issues and most of the low-level programming details with respect to network I/O.

The Java sockets method is better than the CGI method because state is maintained by the application server. Also, each request does not create a new process. Because the server is connected to the client, there is no need for costly initialization and shutdown operations. This method is well suited for 2-tier or 3-tier client/server architectures.

- **Hidden fields:** By using hidden fields, information can be embedded into a form that the user won't see, but which will be sent back to the CGI program when the form is submitted. The CGI program will attach the hidden fields to the new form that it creates and displays to the user.
- **CGI Side Includes:** This mechanism embeds special codes into the HTML document, which are passed and used by a CGI program to maintain state information across several documents.
- **Netscape Persistent Cookies:** The Netscape browser can store and retrieve information via CGI. The information is stored on the client side as key-value pairs, called cookies. This technique only works for Netscape browsers, and the cookie size and number of cookies are restricted.
- **Magic Cookies:** This technique uses a special server that maintains state for CGI programs. The server runs continuously like any other server. CGI programs of all types and purposes can use this server to store information. With this technique, the CGI program knows how to retrieve data that a previous instance of the program sent to the server. [2]

These strategies are cryptic and are really only workaround solutions meant to overcome limitations with HTTP. In addition, as simulations become more complex, these strategies for software development quickly become cumbersome.

This first method is also extremely server-centric, causing a bottleneck at the server. The bottleneck is especially evident when several clients are trying to access the simulation simultaneously. This method cannot handle high traffic, for CGI programs are computationally expensive. Every time a user makes a new request, the Web server spawns a new process, executes the CGI program, and returns results. CGI programs need to also perform application-specific initialization and shutdown tasks for every request, which can make the simulations run extremely slow.

2.0 Web Integration Methods

Several different methods are available to integrate manufacturing process simulations to the Web. These methods, HTTP/CGI, Java sockets, and Internet Inter ORB Protocol (IIOP), are described and illustrated below.

2.1 Method 1: HTTP/CGI

The first method (Figure 1) involves a client/server architecture based on HTTP/CGI. This method uses CGI programs to process user requests.

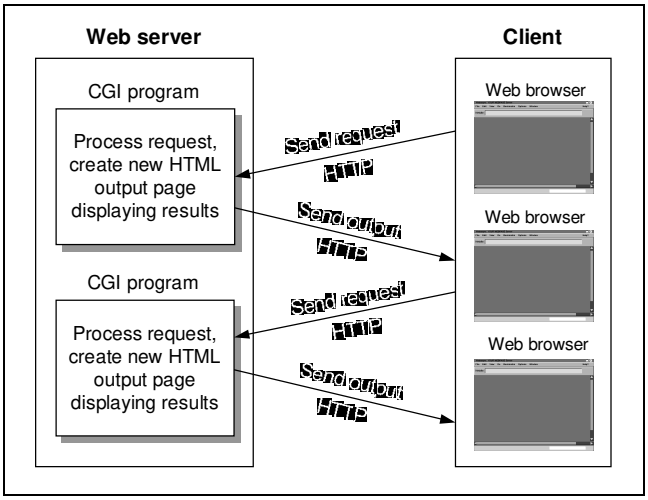

Figure 1: Client/Server Architecture for Manufacturing Process Simulations Based on HTTP/CGI

The graphical user interface (GUI) for the simulation is displayed on a HTML page on the Web browser. Every time the client makes a request, a CGI program is invoked to satisfy the request. The CGI program displays the simulation results in a new HTML form. This method has several problems though.

The first problem is it does not have good support for maintaining state, as HTTP is a stateless protocol. This problem becomes obvious if you have multiple forms wherein you need to exchange state information between the forms. To overcome this limitation with HTTP, different strategies were developed, among them:

Three dimensional graphics information can be represented over the Web via several different methods some of which are:

- image files (Tag Image File Format (TIFF), Graphical Interchange Format (GIF), Joint Photographic Experts Group (JPEG) image file formats)
- Moving Picture Expert Group (MPEG) movie files
- QuickTime Multimedia files
- Virtual Reality Modeling Language (VRML) files

The first three file formats are not well suited for process simulations because they do not support user interaction well and they cannot be created programmatically. VRML, however, is a scene description language, designed and developed to describe 3D environments over the Web. VRML has many features that make it the best 3D graphics format for manufacturing process simulations:

- VRML is a platform-independent file format, a fundamental requirement for the Web
- VRML allows for user navigation, so the user can move around freely in a VRML world.
- VRML supports user interaction, so the user can perform simple tasks, such as rotating, translating and scaling 3D objects, as well as complex tasks such as animating objects.
- VRML files are bandwidth-efficient; they are much smaller than other 3D file formats, and they can be transferred in a compressed format.
- VRML files can be created programmatically, which is very important for manufacturing process simulations.

Java and VRML are key Web technologies that enable the development of manufacturing process simulations over the Web. This paper discusses development issues for Web-based manufacturing process simulations and the various methods of integrating simulations to the Web. This paper also discusses the implementation of a pilot Web-based ductile fracture simulation. Although the focus of this paper is manufacturing process simulations, the techniques outlined here can be applied to simulations in general.

technologies to develop complex and highly interactive applications. To support manufacturing process simulations, the Web requires a robust software development environment with good programming language and interactive 3D graphics support.

The 1995 emergence of Java, an object-oriented programming language, provided the Web with the programming support it needed for developing complex manufacturing process simulations. Java is well suited for the Internet for several reasons:

- It is a platform-independent language, a primary requirement for the Internet. Java programs are compiled into architecture-neutral Java byte codes, which can run on any computer for which there is a Java Virtual Machine available.
- It is highly portable. For example, Java explicitly specifies the size of each primitive data type. Java programs can be embedded into HTML documents, and they execute within the confines of a Web browser. Such programs are called applets.
- It does not have a static link phase. Java classes are dynamically loaded at run-time. For example, if a Java applet encounters a new class that is not available locally, the class can be dynamically loaded over the Web.
- It supports automatic garbage collection, which helps the programmer, especially when developing distributed applications over the Web.
- It has built-in support for threads and sockets.
- It has several security mechanisms to protect the system from malicious code. Java does not support pointers, so a programmer cannot get behind the scenes and manipulate pointers in memory. More importantly, the Java compiler does not handle memory layout decisions, so a programmer cannot guess the actual memory layout of a class by looking at its declaration. Memory references in compiled Java code are resolved to real memory addresses at run-time by the Java interpreter. The Java run-time system uses a byte-code verification process to ensure that code loaded over the network does not violate any Java language restrictions. [1]

World-Wide Web Integration of Manufacturing Process Simulations

by Sridhar Adapalli and Krishna Addepalli
Concurrent Technologies Corporation
Johnstown, PA

Abstract

Manufacturing process simulations typically execute on a single platform, and are language and operating system-specific. With the growth of agile enterprises, however, the need is emerging to perform manufacturing process simulations in a distributed fashion from remote locations. In addition, individual businesses are becoming more decentralized, with geographically dispersed manufacturing and design teams. For effective concurrent engineering, product and process design teams that are in different locations need direct access to simulation tools. In addition, small- and medium-sized businesses cannot afford high-end computers and software for computationally intensive process simulations. Hence, these businesses could greatly benefit from easy access to high-end systems located remotely.

The World-Wide Web provides an excellent medium for remote collaboration and computing. Manufacturing process simulations can be deployed over the World-Wide Web for remote access to simulations. This paper discusses the different methods for integrating manufacturing process simulations with the Web. An implementation of a Web-based metalworking process simulation with defect prediction is also presented.[1]

1.0 Introduction

The World-Wide Web (commonly known simply as the Web) is quickly becoming an integral part of our lives, just like the telephone and television. The Web started out as a way to transmit static data, which is possible because of the hypertext transfer protocol (HTTP) and hypertext markup language (HTML). The communication, however, is only one-way; from the server to the browser. This is acceptable for many applications, however, there is a growing need for user interaction to enable applications such as online bookstores and access to customized data. The Common Gateway Interface, popularly known as CGI, enables the exchange of dynamic data and allows for two-way communication. Programming with CGI, however, becomes quite cumbersome for truly interactive applications, such as a multi-user computer game or a real-time simulation. The Web is driving the need for better

[1]This project was supported by the National Applied Software Engineering Center (NASEC), operated by Concurrent Technologies Corporation (*CTC*), under Grant # MDA972-93-1-0018, funded by the Defense Advance Research Projects Agency. The content of this information does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.