Internet Based Prototyping of Micro-Electro-Mechanical Systems

N. M. Wilson[1], D. Yergeau[2], R. W. Dutton[2]
Mechanics & Computation[1] and TCAD group[2]
Mechanical Engineering[1] and Electrical Engineering[2]
CISX 332, Stanford University, Stanford, CA 94305-4075 USA

Abstract

This paper details preliminary work investigating Internet-based simulation of microsystems. It is in no means exhaustive, and focuses on using mainstream, current web-based technology while highlighting anticipated future directions in relevant internet technology. It proposes a paradigm of automation that limits the information that needs to pass back and forth between a browser-based client and the back-end server. In the present implementation, geometric models are generated from layouts and a process specification, then meshed automatically, and simulated using *hp*-adaptive finite element techniques. The web- and browser-based environment provides file transfer, simulation control, and visualization of simulated results.

Introduction

Without question, the Internet has revolutionized the world we live in. It has had a profound effect on many aspects of computing, yet has had less impact on computer simulation. Three key technical factors have delayed the success of internet-based simulation for products of engineering interest:

1. *Software*. Software has been written to run on a specific machine and platform (i.e. operating system) using application programming interfaces (APIs) that require local access to the machine.
2. *Bandwidth*. Engineering analysis typically requires the use and manipulation of large amounts of data.
3. *Hardware*. Simulation and visualizing of an engineering analysis often requires powerful computers with memory and graphics capabilities beyond that of the typical office desktop machine.

Currently available commercial simulation tools are designed for a "traditional" computing environment. A preliminary system is detailed in this paper that demonstrates an Internet-based prototyping environment for microsystems. The field of MEMS provides an ideal target for Internet-based simulation software for the following four important reasons:

1. *Emerging market*. Unlike the automotive and aerospace industries which have been around and utilizing simulation tools for decades, the MEMS market is relatively new and use of simulation in the design process has not yet become deeply entrenched using existing simulation paradigms.
2. *Cost.* The automotive and aerospace industry are driven by a few large companies that can devote significant manpower and financial resources to computational prototyping. In sharp contrast, MEMS is a field driven by smaller firms and universities with more financial constraints.
3. *Size of models*. A typical model for an aircraft involves millions of unknowns (i.e. degrees of freedom). However, many micromechanical devices of interest can be reasonably simulation with tens-of-thousands of degrees of freedom.
4. *Standardized processes*. Due to inherent difficulties in IC fabrication, several popular commercial processes (e.g. MUMPS) currently exist. Since fabrication runs take several months, significant time is lost between runs if one is iteratively designing a prototype. This provides opportunities for "virtual fabrication runs" which can be done using internet-based simulation tools in between fabrication of devices.

This paper details a MEMS computational prototyping system designed from the ground up to take full advantage of state-of-the-art technology in computer hardware, software, and the Internet. Key design decisions in the software architecture and engineering tradeoffs of computational efficiency for automation are made to enable internet-based prototyping.

Client-Server Overview

The system takes advantage of a client-server paradigm as shown schematically in Fig. 1. The light gray boxes (masks/layout, processing flow, boundary conditions, and material properties) indicate the user provided information. The left side of the diagram also indicates the steps that are carried out on the designer's machine (referred to as the "client"). The right side of the figure corresponds to the processes run on the "server." The server can be the same machine as the client, a different machine on the same local network, or a remote machine accessed via the Internet, even through a proxy. There are three main simulation steps carried out on the server:

1. *Geometry*. This step takes the user-defined process flow and masks for the layout and creates a solid model representing the three-dimensional device.
2. *Discretization*. This step automatically breaks the solid model into smaller non-overlapping discrete pieces (called a mesh) needed for simulation.
3. *Simulation*. This step simulates the device using multi-physics Finite Element Analysis techniques to obtain electrical and mechanical characteristics of the structure.

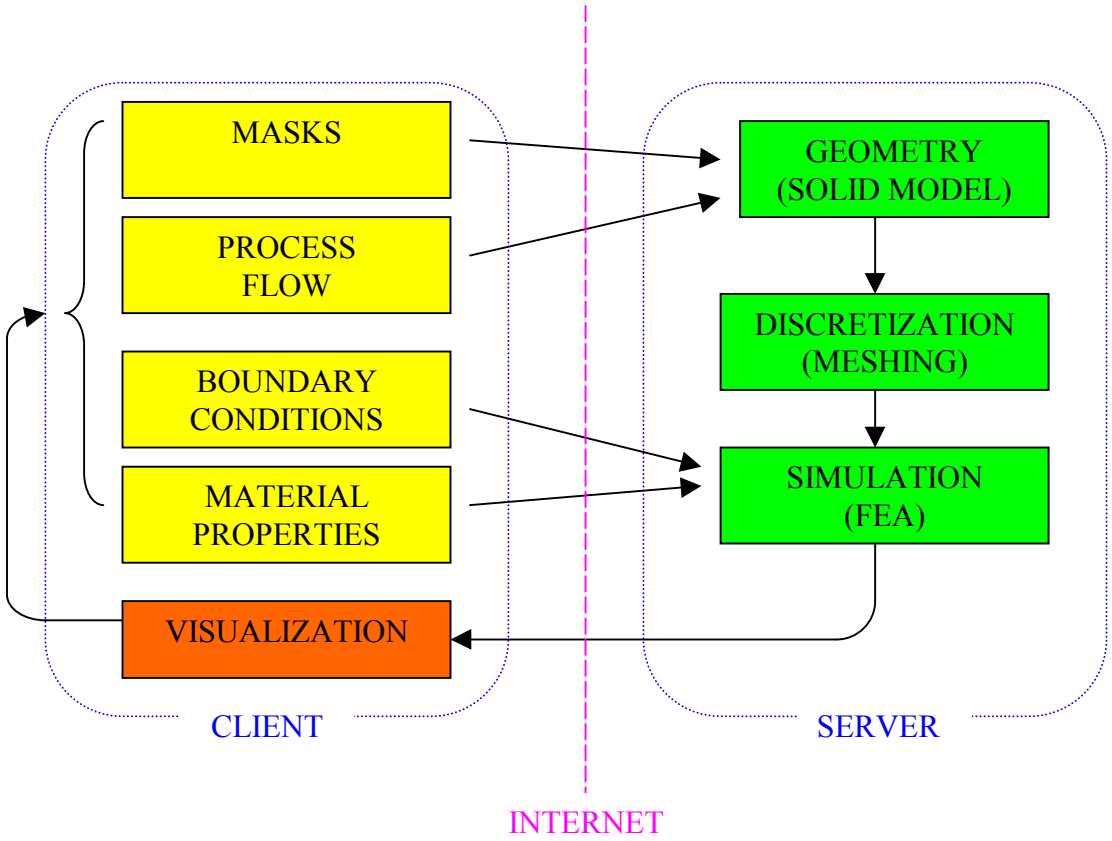

Figure 1: Schematic of client-server architecture.

In the overall system, the boxes on the left correspond to computational inexpensive tasks, while the boxes on the right can require significant computational resources. The communication between the left and the right side is transaction oriented (not interactive) and thus can be done efficiently by transmitting small- to medium-sized blocks of data.

Client Architecture

The client side application is handled through a standard web browser as seen in Figure 2. The browser-based user interface takes full advantage of frames and other HTML enhancements. The toolbar on the left allows selection of several submenus, which in turn update the toolbar and/or the main viewing window to guide the user interactively through the process of going from a design to simulation results. Most of the submenus consist of information entered directly using standard HTML form objects. The user interface was implemented with standard HTML, instead of Java, for greater portability among browsers and to avoid some of the security restrictions of Java (i.e. Java applets cannot access to the local filesystem which would have prohibited uploading of local files containing mask and process flow information). Also, all of the information is transferred using standard HTTP POST/GET methods and server-side CGI scripts. The use of these standard transaction mechanisms permits access to a server across a firewall, provided that an HTTP proxy is available. With the exception of the visualization step, all of the boxes on the left side represent simple data entry. For example, it is assumed the masks are created in a standard layout editor (e.g. L-Edit, Magic, etc.). Thus, this step represents little more than specifying a filename. The process flow is currently specified using the Composite CAD Process Definition Specification [1].

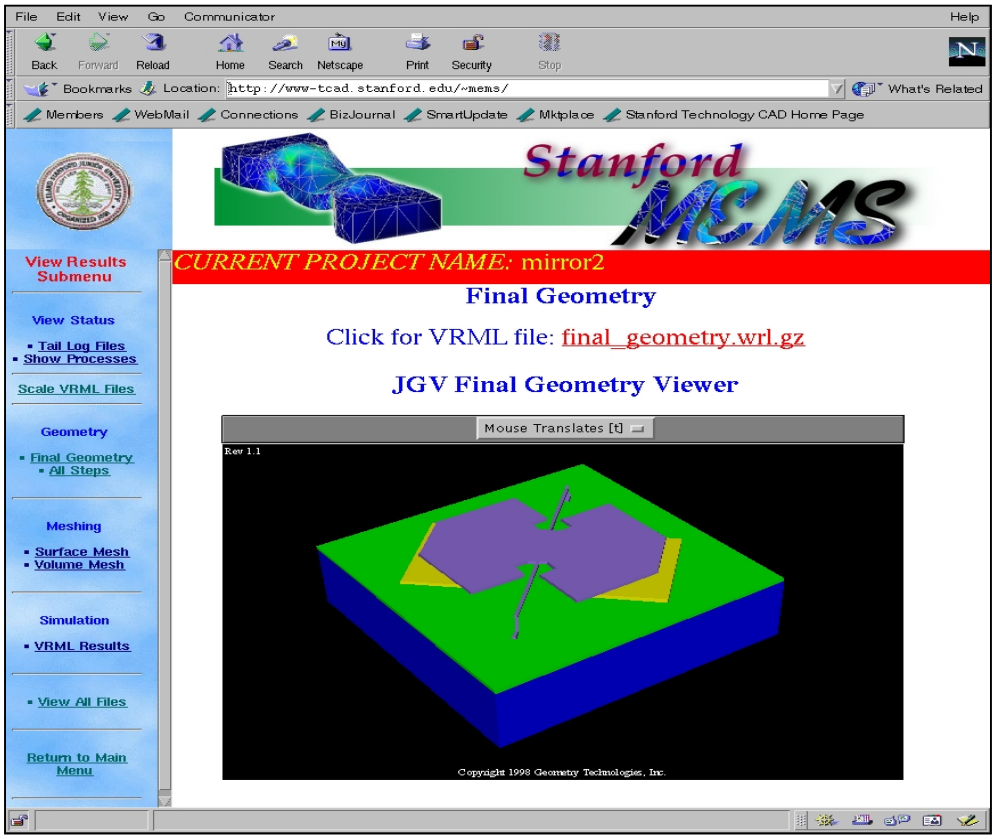

Figure 2: User interface inside of a web browser.

Unfortunately, the transaction-oriented nature of HTTP introduces latencies that are too high for highly interactive use. Tasks that require a high level of interactivity or do not map well into a forms-based interface are implemented using Java applets that are downloaded to the browser. In addition to JGV, there is a Java applet that aids in defining a correct process flow specification. This Java applet also uses an HTTP POST method to communicate with the CGI scripts on the server.

Visualization of simulation results on the local machine is an important step of the process. Several visualization techniques were tested. First, visualization of the geometry and the surface meshing was achieved using the Virtual Reality Markup Language (VRML) [2]. VRML was one of the first standard file formats proposed for 3-D internet-based visualization. Unfortunately, the standard was never universally adopted and VRML viewers are typically limited to only a subset of the functionality and are not available on all platforms. The VRML standard is no longer being developed, and the Web3D consortium has migrated its efforts into developing a new extensible 3-D format [3] for internet applications. The second attempt for portable visualization was to use a model viewed called JGV [4], an applet implemented in pure Java. The viewer was less elaborate than the VRML viewers, but did not require special plugins and is only 100 kilobytes in size, small enough to be transmitted with the geometry over low bandwidth links such as a 56 Kbps modem.

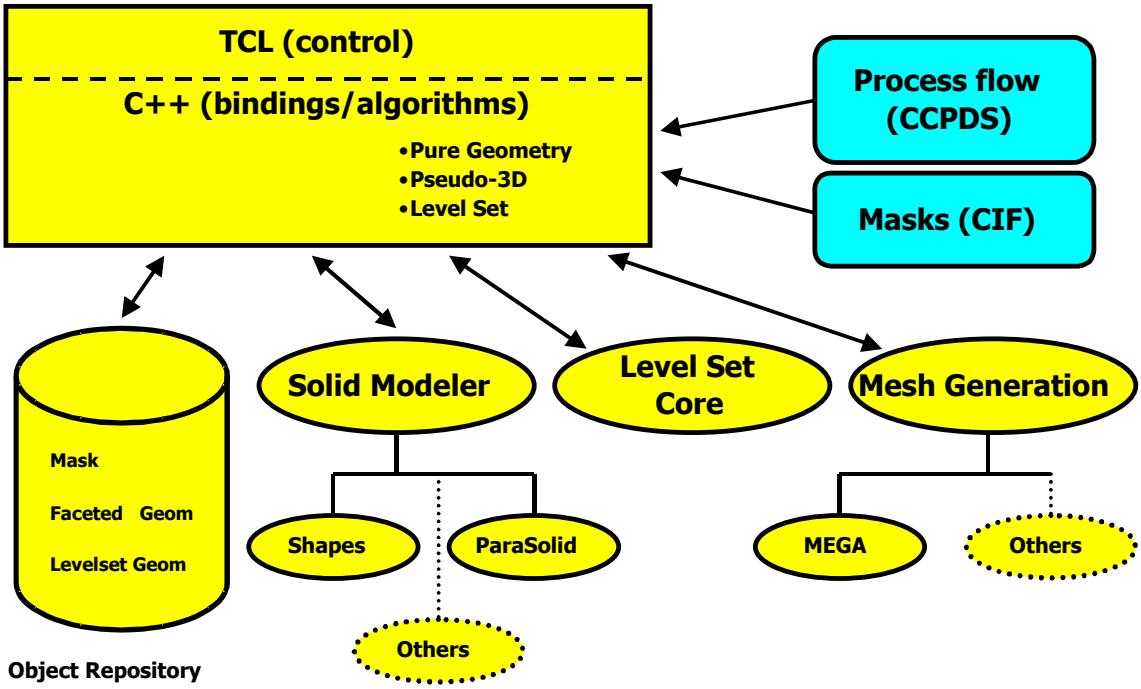

Figure 3: Geodesic Architecture.

Server Architecture

The server side framework is implemented as a collection of PERL CGI scripts that manage data and control other simulation tools on the server. The geometry generation tool, known as Geodesic, is detailed in [5]. Figure 3 depicts the architecture of Geodesic. Geodesic consists of an extensible framework to create

MEMS geometry of varying degrees of physical accuracy using multiple geometric simulation techniques. In addition, it has an integrated meshing layer that permits automatic tetrahedral mesh generation. Finally, it creates an input file for a finite-element code, based on the ProPHLEX finite-element kernel, which permits *hp*-adaptive finite element simulation of the coupled electromechanical problem.

Discussion and Conclusions

Table 1 shows examples of file sizes for various micro-electro-mechanical devices. Both VRML and OOGL (the file format used by JGV) are ASCII file formats. OOGL does have a complementary binary format, unfortunately it is not supported by JGV. Most VRML browsers permit the input files to be compressed using the gzip compression algorithm. This greatly reduces the size of the actual file transmitted, as can be seen in Table 1. As the table indicates, the geometry can be transmitted in a reasonable amount of time even over a 56 Kbps modem. Also shown are example file sizes for coarse meshes for each of the devices. Note that even though a volume mesh is being generated for each device by the server, only the surface mesh is being transmitted back to the client for viewing.

Table 1: File size (in bytes) of four example MEMS devices.

Device	Geometry		Surface Mesh	
	VRML*	OOGL	VRML*	OOGL
Dual Electrode Switch	6116	43119	9926	61510
Comb Drive	52257	479440	87137	580514
Torsional Micromirror	4215	28743	11827	149934
Simple Switch	1429	8012	2762	30909

* gzipped file size

An internet-based simulation environment has been prototyped using a browser-based client and a back-end server. Established internet standards were utilized to provide universal access and portability between browsers. Evolving new standards such as X3D and XML hold great promise for additional capabilities useful for internet-based MEMS simulation provided they become universally accepted and implemented standards.

Acknowledgements

This work was supported under Air Force/DARPA F30602-96-2-0308.

References

1. "Composite CAD Process Definition Specification version 1.0," DARPA internal document.
2. http://www.web3d.org/vrml/spv2.htm "the VRML repository"
3. http://www.web3d.org/x3d.html "Extensible 3D (X3D) Task Group"
4. http://www.geom.umn.edu/java/JGV/ "JGV: 3D Viewing in Java"
5. Wilson, N. M., Wang, K., Yergeau, D., and Dutton, R.W. (2000): "GEODESIC: A New and Extensible Geometry Tool and Framework with Application to MEMS," Proceedings of Modeling and Simulation of Microsystems, March 27-29, pp. 716-719.