JAVA APPLETS / SERVLETS FOR WEB-BASED CLIENT/SERVER PARAMETER STUDIES

Reinhard Finsterwalder

Universität der Bundeswehr, München
Wissenschaftliche Einrichtung Mathematik & Informatik
D-85579 Neubiberg, Germany

Reinhard.Finsterwalder@unibw-muenchen.de

Abstract: The Internet offers many new software technologies – the most notable is Java. Java not only specifies a computer language but serves also as a complete toolbox for building client/server solutions. We use this technology for developing a framework for remote computation. As application example serves the parametric multi-criteria robustness exploration of the dynamics of a high-fidelity aircraft model. *Copyright © 2000 IFAC*

Keywords: Remote Simulation, Client-Server Architecture, Java Applet, Java Servlet, Parametric experimenting

1. INTRODUCTION

Multi-objective design assessment and control law synthesis tuning relies on fast simulation evaluations of high-fidelity models. High fidelity models, in general, are the result of a multidisciplinary, collaborative effort. The simulation of such models usually require a lot of computation power. In addition, these models and their corresponding simulation code are frequently bound to a specific computer platform. Thus they are quite probably executed on a distant high performance computation server.

Client-server technology supports the remote exploration at the control designers computer. This technology enables several clients that are running on local machines to forward simulation requests to the remote computation server. The server accepts a client's request, performs the associated computation and returns the results to the client.

The Java language environment, World Wide Web, and Client-Server architecture are complementary software technologies, which, when used together provide a powerful set of tools for developing and deploying multi-user distributed applications (Evans and Rogers, 1997). In this article we describe an approach to build easy-to-use client software as WWW-downloadable Java applets, which use the World Wide Web to interact with servlets running on a remote server. We use this technology for developing a framework for remote computation. As application example serves the parametric multi-criteria robustness exploration of the dynamics of a high-fidelity aircraft model used in (Finsterwalder, *et al.*, 1999).

2. REMOTE SIMULATION

The client-server approach has several benefits. First, multiple users working on their local PCs can access shared simulation software and computation power concurrently. Second, the cost of supporting and maintaining one large simulation server is less than the cost of maintaining separate simulation programs on each client computer.

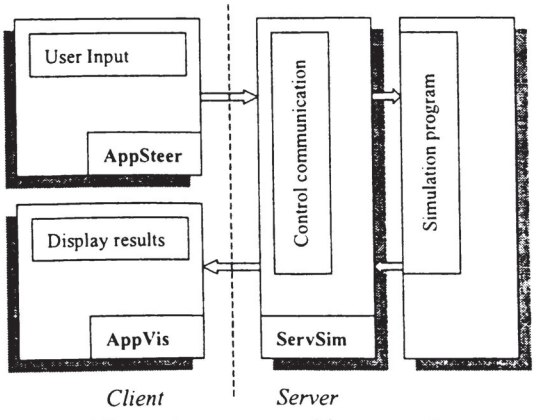

Fig. 1. Client / server architecture for remote simulation.

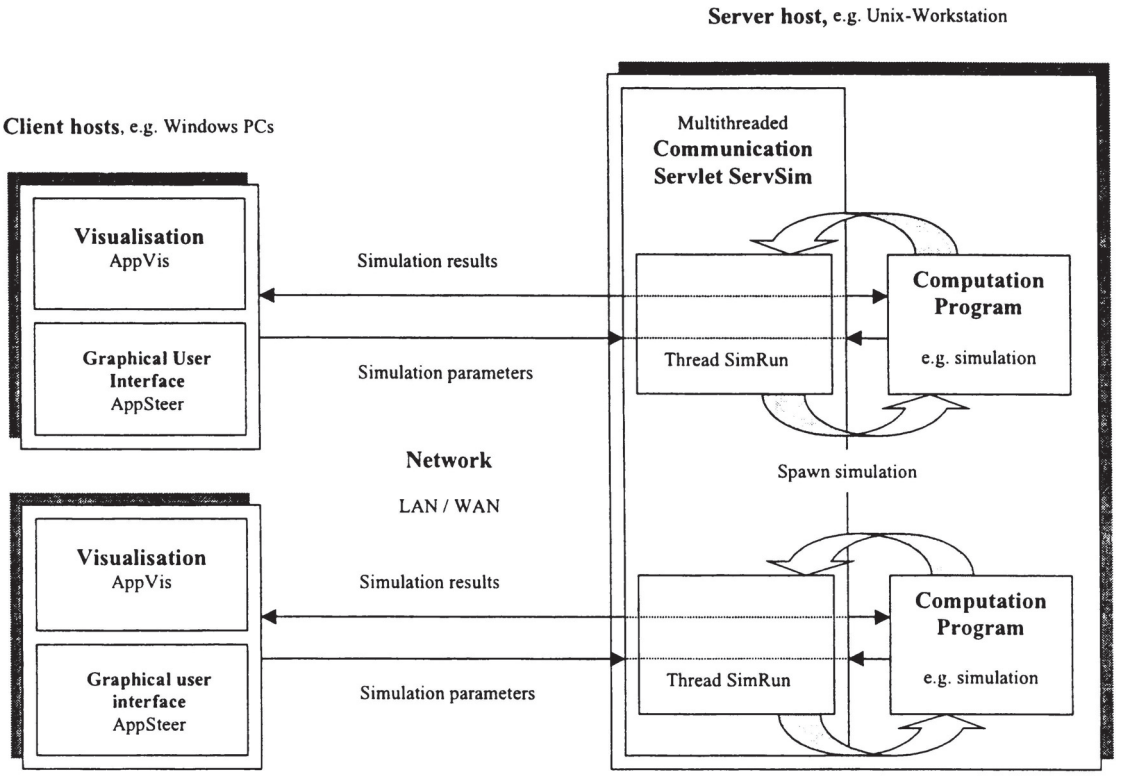

Fig. 2. Parallel execution of simulation runs.

Because data is exchanged across a network, the performance of remote simulations is generally lower than when sitting in front of the server. To maximise performance, data transfer has to be kept as small as possible. Minimum data transfer is guaranteed by splitting the simulation software into four components: 1) Graphical user interface *AppSteer* for steering the computation experiment, 2) application specific computation program, e.g. simulation program for performing numerical calculations, 3) visualisation system *AppVis* for runtime-monitoring and post-computational analysis, 4) communication servlet *ServSim* that controls the data flow between client and server computer, figure 1.

The multithreaded implementation of the communication servlet allows multiple access of clients and parallel execution of simulation runs, figure 2. *ServSim*, *AppSteer* and *AppVis* are realised as independent programs that communicate with each other via inter-process communication. This enables the distributed execution of the software in a heterogeneous computer network. In addition they are not tied to a specific simulation program. Thus they can be used in conjunction with any other simulation / computation executable. The interface to the simulation program also takes place on the process layer. For that we applied a generic process interface technique that we have presented in (Finsterwalder and Bals, 1996).

3. JAVA APPLETS AND SERVLETS

Both the clients *AppSteer* and *AppVis* and the server program *ServSim* are implemented in the object-oriented programming language Java (Gosling, *et al.*, 1996).

Among its many interesting features there are two making it attractive for use in distributed applications. First, Java programs can be compiled into a machine-independent format called the bytecode. Upon execution this bytecode is interpreted by a machine-specific Java interpreter. Since the Java interpreter is an integral part of any available Web-Browser, Java programs can be executed on any PC or UNIX workstation where a Web-Browser is installed. Second, Java programs can be incorporated as applets into HTML homepages. When a Web-Browser finds an applet tag in the current html page, it automatically downloads the corresponding bytecode from the Web-Server to the user's host computer and executes the applet there. This feature provides a simple mechanism to deploy software over the Internet without having to be concerned about installing, configuring or maintaining software on the client PC.

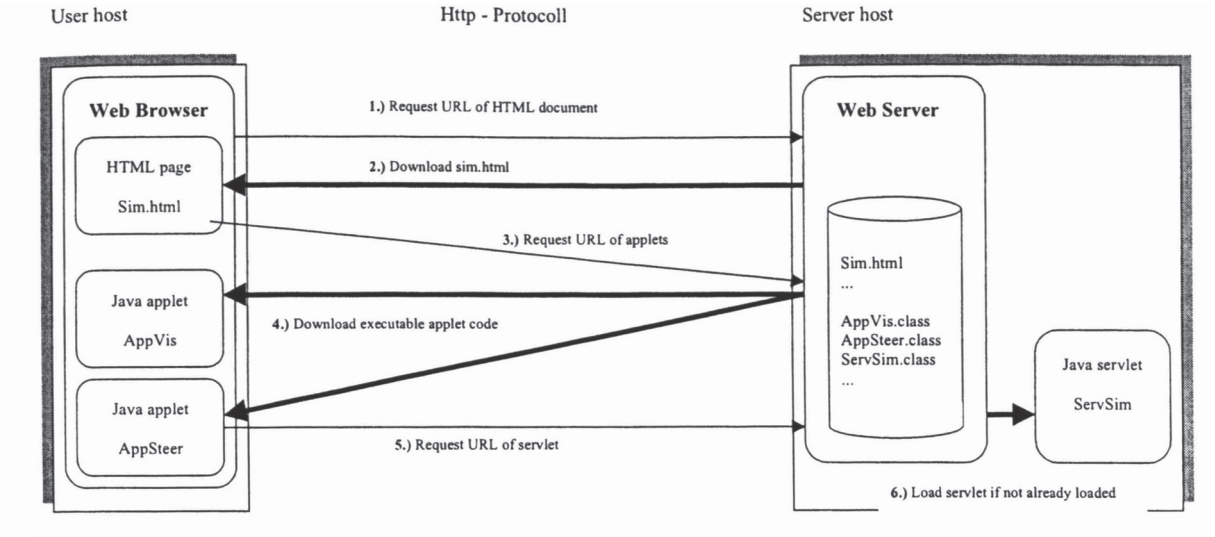

Fig. 3. Communication flow between Applets *AppSteer*, *AppVis* and Servlet *SimServ*.

A servlet is something like a server-side applet (Moss, 1998). Servlets are loaded an executed by a Web server in the same manner that applets are loaded and executed by a Web browser. The servlet technology is the missing brick for building client/server software in pure Java. The Java Servlet Development Kit (JSDK) is included as part of the Java Development Kit JDK 1.2.

Servlets are more and more replacing CGI[1] scripts, because CGI scripts are typically written in Perl or C and thus are tied to a particular Web server platform. Servlets are persistent, in contrary to CGI scripts which are transient. This means that servlets are loaded only once by the Web server and can maintain services between client requests. This makes servlets faster than CGI scripts which are removed from memory between sequential requests.

The clients *AppSteer* and *AppVis* are realised as Java applets The communication program *ServSim* is realised as Java servlet. AppSteer and AppVis make use of the *java.awt* (abstract window toolkit) class library that contains all usual components like push buttons, radio buttons, lists, menubars, etc. For process communication the *java.net* package is applied.

4. WEB-BASED PARAMETRIC EXPERIMENTATION

The proposed Internet techniques are applied to build a framework for performing parametric computation experiments on a remote computer host. Figure 3 illustrates the basic communication flow between the client applets and the servlet on the server computer.

To run a simulation experiment, the user simply has to starts its Web-browser and to load the simulation homepage *sim.html*. Since *sim.html* contains two applet tags, the browser makes further requests to download the machine-independent bytecode of the applets *AppSteer* and *AppVis*. The browser loads the applets in its JAVA virtual machine and executes them. Now the simulation program is ready to use. When the user clicks on the *Start Simulation* button of applet *AppSteer*, a request for a new simulation run is send to the Web-Server. The Web-Server forwards the request to the simulation servlet *SimServ* that spawns a new simulation thread *SimRun* and creates a TCP/IP socket connection between *SimRun* and the client applet *AppSim*. Then *SimRun* loads the actual parameter values, invokes a new simulation run and forwards the results to applet AppVis.

```
class ServSim extends HttpServlet
{

  public void run()
  {

    for(;;)
    {
      // waiting for work
      Socket client = server.accept();

      // spawn new simulation thread
      SimRun sim = new SimRun(client);
    }
  }
}
```

Fig. 4. Mulithreaded implementation of servlet SimServ

[1] CGI = Common Gateway Interface – server-side scripts written in Perl or C

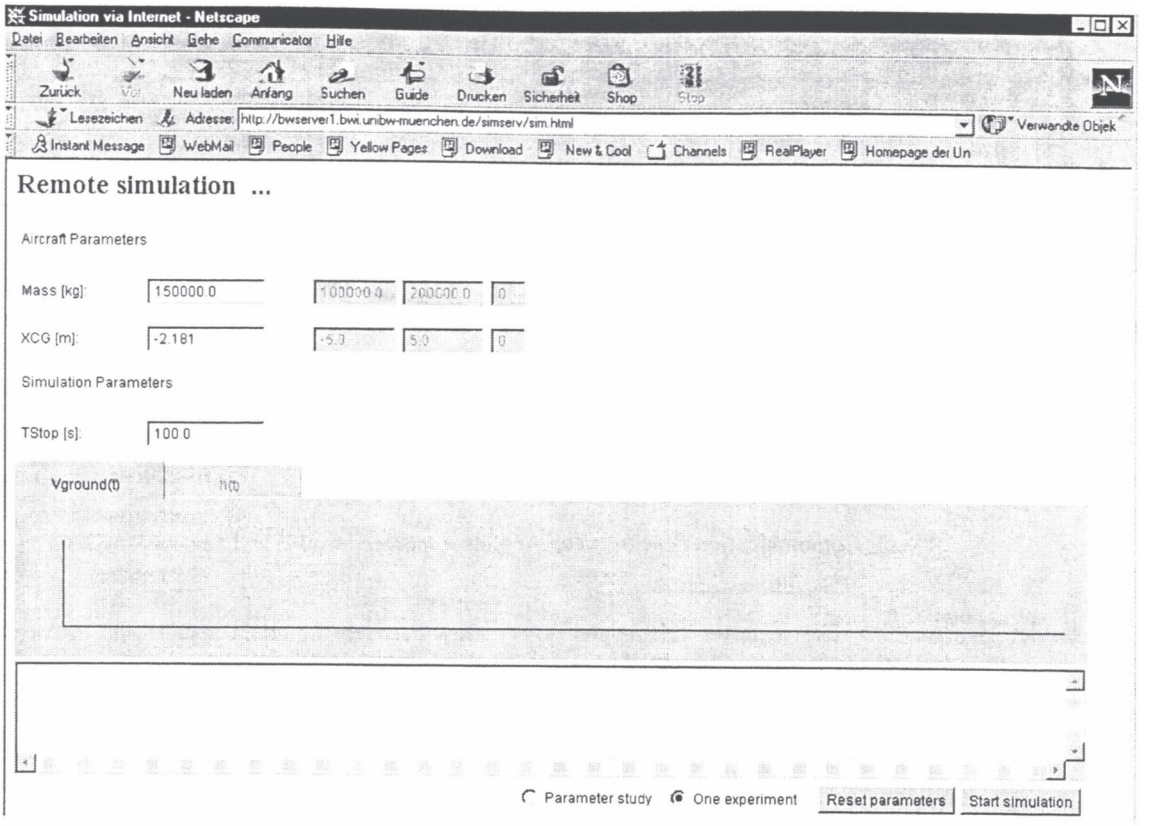

Fig. 5. Screenshot of applets *AppSteer* and *AppVis* in action.

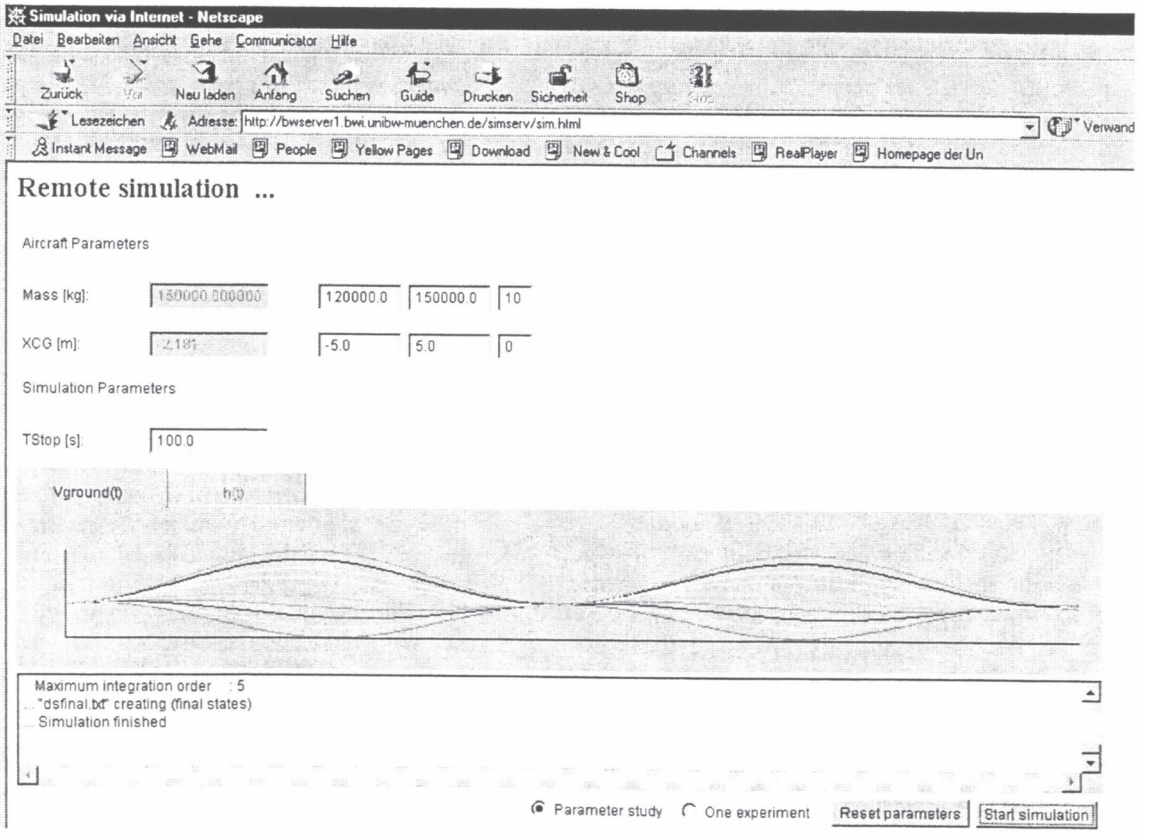

Fig. 6. Variation of parameter mass – 10 simulation runs.

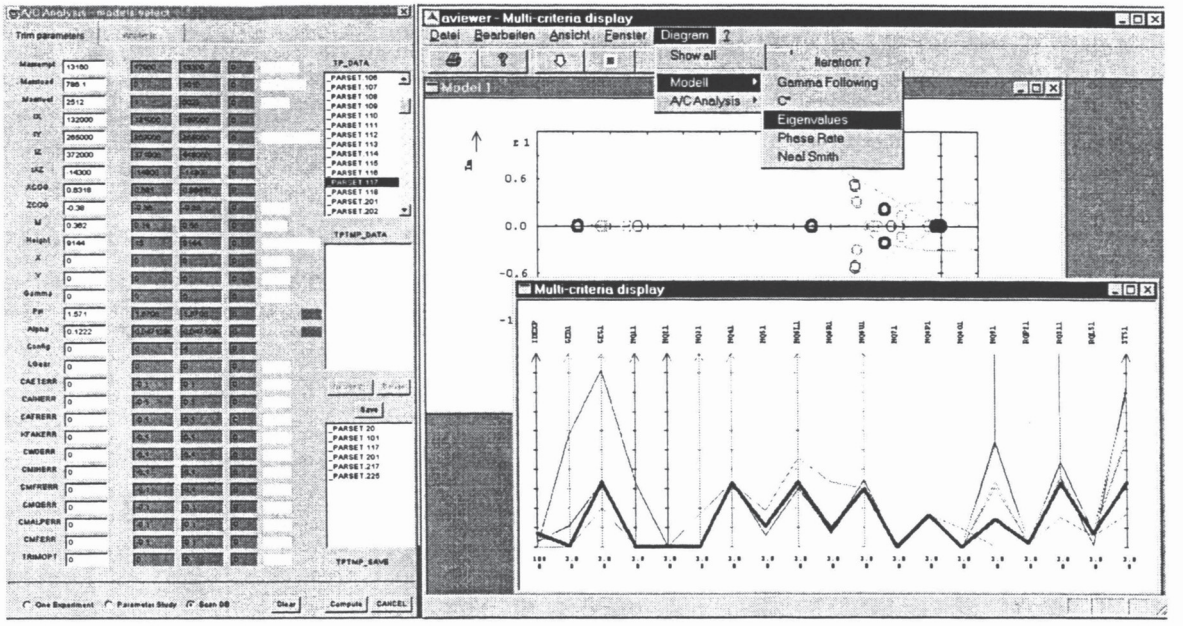

Fig. 7. Multi-objective parametric computation experimenting including visual decision support.

5. APPLICATION EXAMPLE

As demonstration example serves the simulation of a parametrised high-fidelity aircraft dynamics model with two variable parameters, aircraft mass and longitudinal center of gravity.

Two experiment modes are currently provided:

- Interactive experimentation: Single simulation run with user-given parameter values, figure 5.
- Parameter study: Automatic variation of one or more parameters in user-specified ranges, figure 6.

This example is available in the Internet. It can be loaded from http://bwserver1.bwi.unibw-muenchen.de/simserv/ sim.html.

We also applied this technology to build a first prototype for an integrated multidisciplinary environment for flight control development, that supports the user during all design phases: *aircraft-analysis, tuning & compromising and assessment* (Finsterwalder, *et al.*, 1999), figure 7.

6. SUMMARY AND CONCLUSIONS

In this paper, state-of-the-art Internet technology is reviewed to build a remote simulation environment for design assessment. For evaluation purpose, a first software prototype has been developed that is based on a client-server architecture written in Java. Although the current implementation is still a-proof-of-concept which lacks some of the features which are required for professional use, we believe that the combination of WWW-based applets and servlets is a practical approach for developing multi-user distributed computation experimentation frameworks.

REFERENCES

Evans, E. and Daniel Rogers (1997). Using Java Applets and Corba for Multi-User Distributed Applications. *IEEE Internet Computing*, May - June 1997, pp. 43-55.

Finsterwalder, R., H.-D. Joos, and A. Varga (1999). A Graphical User Interface for Flight Control Development. In: *Proceedings of IEEE International Symposium on Computer-Aided Control System Design* (Hapuna-Beach,. Hawaii, August 22-27).

Finsterwalder, R. and J. Bals (1996). A Generic Interface Technique for Interactive Use of File-Driven Software in a CACE Environment. In: *Proceedings of IEEE International Symposium on Computer-Aided Control System Design* (Dearborn, MI, September 15-18).

Gosling, J., B. Joy and G. Steele (1996). *The Java Language Specification*. Sun Microsystems Inc., Addision Wesley.

Moss, K. *Java Servlets (1998)*. McGraw-Hill.