

Web Computing Skeleton: A Case Study

N.B. Šerbedžija, L. Botha,* A. Abbott* and J. Bishop**

**GMD FIRST
Rudower Chaussee 5
D-12489 Berlin
nikola@first.gmd.de*

**Computer Science Department
University of Pretoria
Pretoria 0002, SA
louis@epiuse.co.za*

Abstract

In this paper an approach to provide Web-based framework for distributed execution of collaborative applications is presented. The work is a part of a wider on-going project whose aim is to make environmental simulation models publicly available to the Internet users. The Web computing skeleton has been constructed from prefabricated Web-enabled components with the ability to open and maintain Web connections and provide collaboration over the Internet. The case study illustrates how a single-user simulation system can be embedded into the skeleton, thus becoming widely available distributed application.

keywords: Web-Collaboration, Heterogeneous Systems, Environmental Simulation

1 Introduction

In the recent years much effort has been put into development of Web-technology, primarily providing global access to hyper-text. As the use of Internet spread, more complex applications became feasible [1]. With the appearance of Java programming language and Java run-time system [2, 3] the dream of global heterogeneous distributed computing merged with reality. This paper illustrates the design of a Web-based system using distributed object-oriented techniques to achieve heterogeneous, mobile and dynamically configured widely available software.

Web computing can be defined as a special kind of distributed computing that involves Internet-based collaboration of several remotely located applications. Contrary to the "classical" distributed components that are static and rely upon a fixed hardware configuration within a local area network, a new approach advocates existence of dynamic functionalities that can be easily migrated and executed on any machine connected to the Web. The new style of computing requires a uniform run-time environment, broad-band connections and flexible software structures.

In this paper an approach to provide a World Wide Web-based framework for the distributed execution of various programs is presented. Section two describes Web computing skeleton, a Java coded software template for Web collaboration. Section three presents a case study showing how a simulation system can be embedded into the skeleton. Section four discusses the results achieved and indicates the directions of further work.

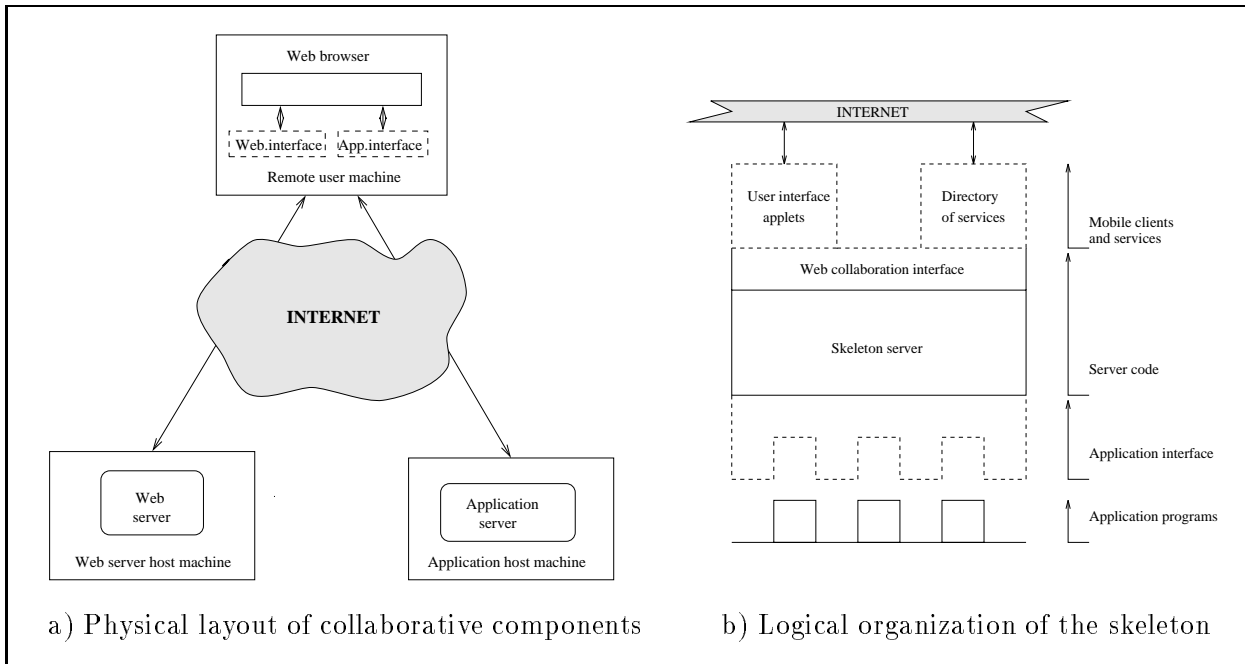


Figure 1: Web computing skeleton

2 Web Computing Skeleton

The purpose of the skeleton is to provide basic software structures for Web computing, i.e. it should establish necessary Internet connections and enable Web collaboration between different remotely located programs. The actual functionality is obtained by embedding application programs in the skeleton and providing appropriate interface to these programs.

2.1 Design Principles

The Web computing skeleton is a distributed and extendible software template for enabling existing software to run in the World Wide Web environment. It is designed to allow for robust, distributed collaboration, based on a client/server principle.

Figure 1 illustrate both physical layout and logical organization of the skeleton components. Physically, skeleton is distributed over at least three machines, each running different collaborative components: a Web browser, a Web server and an application server. Only after visiting the server sites, a remote Web user obtains the interface code needed for further collaboration (downloaded code is indicated with the dashed rectangles). The number of application servers within the skeleton is not restricted.

The logical structure of the skeleton is given in the second part of the figure 1. It contains the following four major parts:

Mobile code - a set of pre-programmed procedures which can be downloaded to an anonymous Web browser and play a role of the skeleton clients. The code contains procedures for the user interface and connection to other Web servers. The

presence of another Internet address allows for inclusion of the third collaborative part, enabling the use of new functionalities offered at the given Web address. It may be seen as a dynamic reference to a new server.

Server code - a daemon-like program that waits to be called by its mobile clients. Once called it performs the task requested, i.e. initiates application programs and interacts with the clients. The server also maintains a dynamic list of services with descriptions of the available functionalities.

Application interface - a set of procedures that provides the communication and synchronization with the local Web-enabled programs.

Application programs - a set of existing local programs that are offered for the Internet use.

The services offered by the skeleton are actually embedded application programs. Most of these programs are executed at the server site, but it is possible to offer functionalities which can be downloaded to the client and executed at its site. However, in this case an application program (or part of the program) has to be re-programmed in a form of applet that can be sent to the end user.

2.2 Skeleton Internals

The internal structure of the skeleton can be divided into three logical parts, namely the applet running in a remote user's Web browser virtual machine, the directory server and the application server. Each of these parts executes on physically separate machines.

The applets running in the remote user's Web browser present an interface to the processing that is taking place elsewhere. The skeleton has one pre-defined applet which the browser downloads from a Web server. This applet contacts the directory server to retrieve a list of available applications, and on selection of an application the applet will download the new user interface binary from a remote machine and run it.

The directory server maintains a database of available applications and their locations on the various application servers. It is able to control access to applications by means of an access control list mechanism. Each application can have its own access control list with user names and passwords protecting applications from unauthorized use.

The application server host is the machine that does the real processing. It contains a Java interface to the application being accessed, as well as some binary classes that can be run in the user's Web browser as a user interface to the application.

Figure 2 shows the typical flow of control within the skeleton. Initially the Web browser contacts the Web server(1) and downloads the initial applet(2). The Web page downloaded with the applet contains the host name of the directory server to contact. The applet then contacts the directory server(3) and downloads a list of applications(4). When the user chooses an application to execute and passes the access control system, the initial applet contacts the relevant application server(5) and downloads the new user interface classes(6), using the custom class loader built for that purpose. The new user

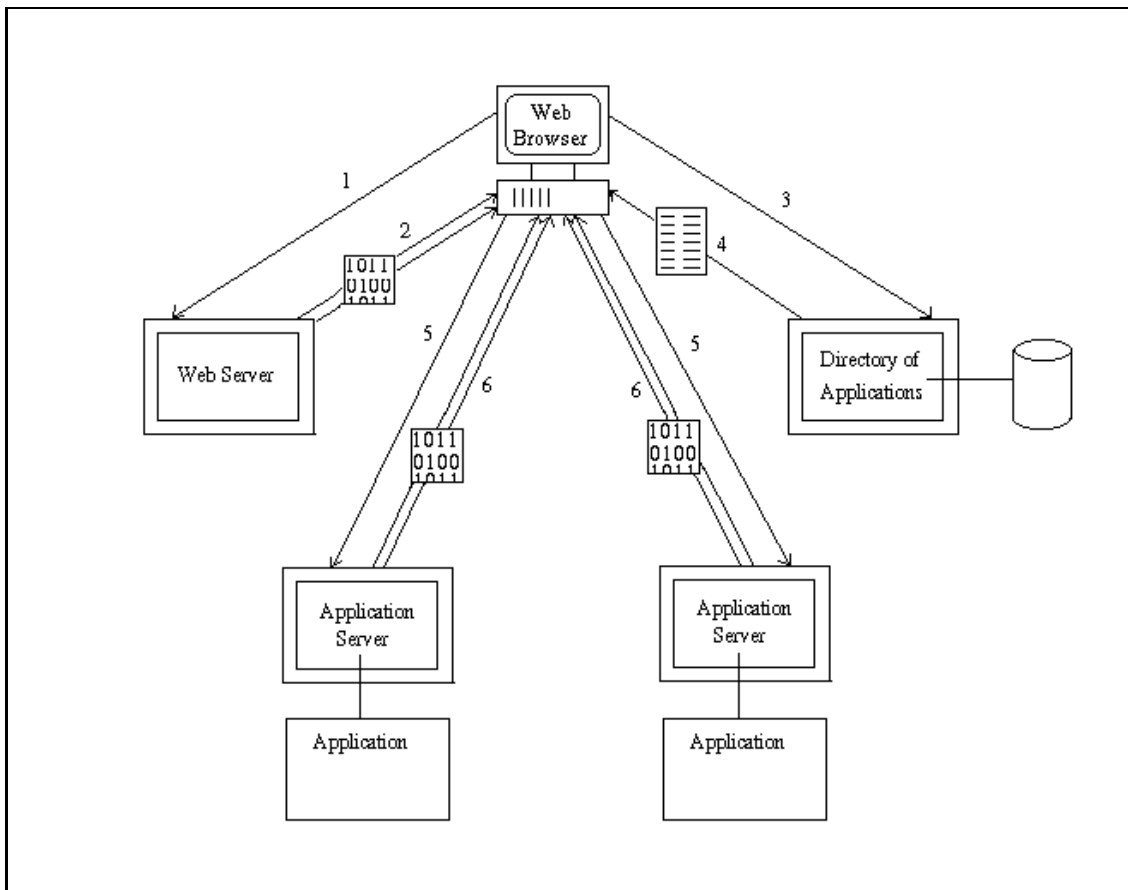


Figure 2: Web computing skeleton details

interface is then started up and control passes to it. The user interface contacts the application server and maintains remote collaboration.

2.3 Java-based Implementation Details

The Web computing skeleton is programmed in Java and contains two code types. One is a set of movable Java applets prepared to run on a remote machine. Another is non-movable code that resides on the server site and waits to be invoked from the Internet. The skeleton also contains a set of html-pages used as a data-driven coordinator for the user-interface.

Communication among skeleton collaborative components is implemented using the Java Remote Method Invocation (RMI) mechanism [4]. It is also the preferred way of communication between the application server and the user interface to the application, but it is not a necessity. Sockets, CORBA or any other communication method may be used without any change to the skeleton.

The Java Remote Method Invocation (RMI) system provides a mechanism for distributed communication on the method invocation level. The fact that Java RMI is designed specifically to operate in the Java environment means that it is less complex

than alternatives such as CORBA, which must cater for multiple source languages and heterogeneous environments. This single-language approach also allows RMI to take advantage of the Java object model and integrate more seamlessly into the Java environment.

A remote object, in the Java distributed object model, is an object whose methods may be invoked from another Virtual Machine (VM). These two virtual machines may or may not reside on the same host. A remote object implements one or more remote interfaces, which declare the methods that may be remotely invoked. Remote interfaces extend the interface `java.rmi.Remote`.

Invocation of a remote method with Java RMI is syntactically identical to a local method invocation. However, due to the added complexity of remote invocation and increased potential for failure, clients must handle additional exceptions that can occur. References to remote objects can be passed as arguments or return values in both local and remote method invocations. For bootstrapping a URL-based registry is provided, allowing URL's of the form “`//hostname/objectname`” to be bound to remote objects.

Remote objects are always passed by reference. Since a local object reference is valid only within its VM, local objects must be passed by copy rather than reference in remote method invocations. Pass by copy is implemented in Java RMI with the aid of the Java Object Serialization package. For security reasons Java objects are not serializable by default. Classes must implement the `java.io.Serializable` interface in order to be serializable [5].

3 A Case Study: Environmental Simulations

Environmental simulations belong to the category of applications with an acute need for a wider access. They are needed in both public and scientific domains, and contrary to commercial systems, there are practically no restrictions for their use. Actually, the only restriction is a technical one: they run on dedicated machines and are not widely available. This is exactly the point where Web-collaboration may fulfill its promise allowing wide-spread use of exclusive applications by distant connection.

3.1 The FAST Simulator

For a prototype development the groundwater simulation system FAST [6] has been selected. It is a collection of several simulation programs used for modelling in an irrigation field, prediction of pollutant concentrations, modelling saltwater intrusion etc. All simulation programs are based on discretisation of differential equations used to model hydrodynamical, geophysical, geochemical and microbiological processes.

The FAST programs are already equipped with a rich graphical user interface that allows for comfortable data-input, display of results and filed data output (which can be further post-processed by the same or different simulation programs). Nevertheless, programs are single-user, PC-based and are only available in the laboratory where the whole system is produced.

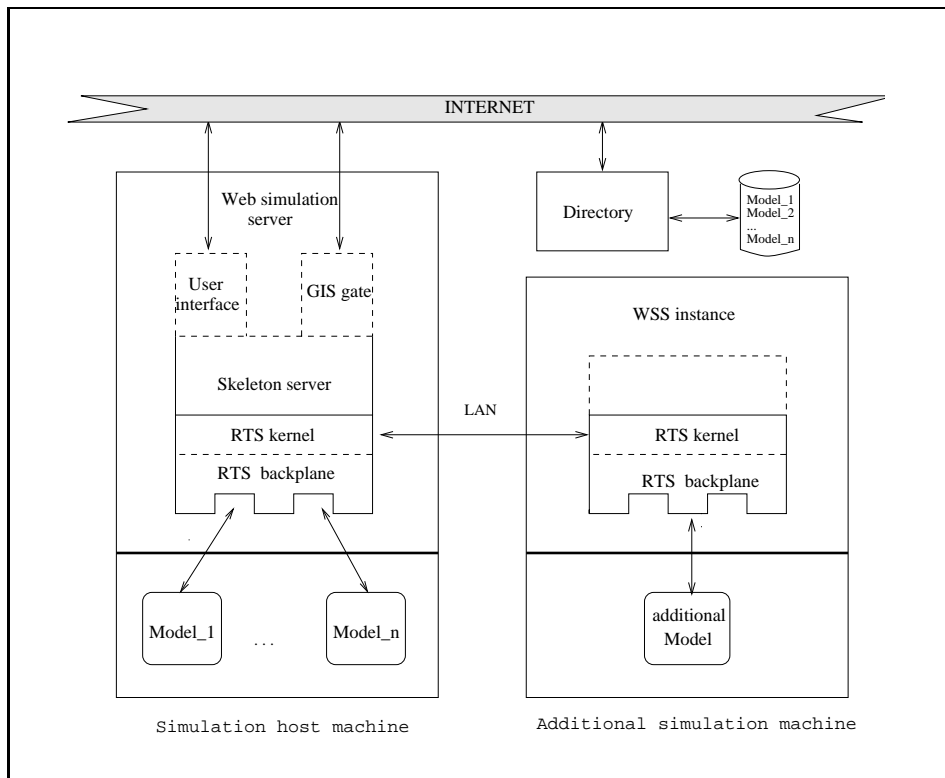


Figure 3: Web collaborative simulation platform

The FAST simulator is a perfect candidate for the Web enabling. Embedding the FAST system into the skeleton would provide a wider use of the simulator with low development costs. Furthermore, the user-interface functionalities may be improved by using versatile and familiar Web-browser style.

To embed the FAST system into the skeleton two major modifications have to be done: (1) development of collaborative simulation platform and (2) redesign of end-user interface. The first extension belongs to the server part of the skeleton (i.e. resident code), and the second to the user interface (i.e. mobile code).

3.2 Collaborative Simulation Platform

The Web computing skeleton sketched in the previous section may be used to enable simple programs to the World Wide Web. However, embedding more complex systems, such as the FAST simulator, requires further extensions. The main skeleton structure remains, but new software layers have to be added, that would provide distributed execution of different simulation models and access to a geographical information system (GIS) [8, 9].

The overall structure of the collaborative simulation platform is presented on the Figure 3. To allow for distributed execution of simulation models, the simulation kernel may be instantiated at the additional simulation computer, connected with the main simulation server via local area network (LAN).

3.2.1 Run-time simulation kernel

The run-time simulation kernel is additional software layer added to the Web computing skeleton to provide a common execution framework for different simulation models. Its objective is to support distributed collaboration among the FAST models. The actual distribution is hidden from the user (the user does not know on which computers the programs are actually running). During the execution of a simulation session, the kernel establishes and maintains physical links between different computers (using local area network) and supports actual data and control flow resolving possible differences between computers.

The distributed simulation kernel maintains a set of complex data objects, called simulation backplane. It is used to interface different simulation models providing a common data structure, common time strategy (dynamics) and common synchronization rules [10]. Each simulation module may have different internal data representation and its own dynamics. To synchronize these heterogeneous programs, common data types are maintained within the simulation backplane together with the conversion procedures. All data exchange among the simulation models and between the simulator and the skeleton is done via the backplane.

3.2.2 GIS interface

There are two types of information which environmental simulation systems require from a GIS: (1) numerical geographic data, needed as input data for a concrete simulation session and (2) map-styled interpretations, needed for displaying patterns, relationships and trends obtained as a result of the simulation session. The first type of data can be prepared in a form of regular queries to the geographical data base: a simulation server sends a request and receives the data from the GIS server. The second type of information that includes map-based interpretation requires massive conversions and transfer of map images in a form suitable for Web browsers.

The connection with a GIS system has yet to be integrated into the current version of collaborative simulation platform.

3.3 User Interface

Web computing skeleton interface consists of a few "html"-based descriptions and a number of Java-based applets that support data input for a concrete simulation, file transfer for bulky data and graphical display of simulation results.

The following functionalities are made available to a Web simulation browser:

- writing a scenario for a simulation session (defining which simulation programs are to be used, specifying input data, dynamics etc.)
- controlling the execution of a given scenario, through a number of commands for monitoring a running simulation (e.g. viewing intermediate results, changing the simulation dynamics etc.)
- calling GIS services

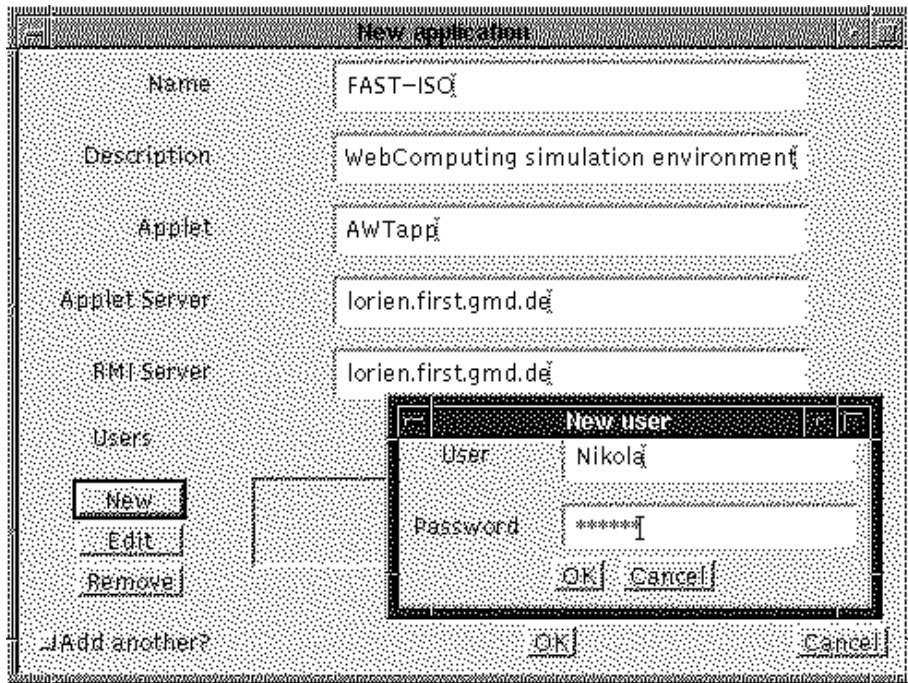


Figure 4: Skeleton directory: adding a new application

- viewing the simulation results
- archiving the simulation session (keeping log information about the simulation sessions)
- replaying the simulation session (re-running the same simulation session, which was previously prepared)

All these operations are pre-programmed in a typical Web browser-style to make the user interface familiar and easy to use.

In the text that follows two examples will be illustrated: (1) adding new application to the skeleton and (2) viewing the results.

3.3.1 The directory example

The directory contains all the data and descriptions of Web-enabled programs. The skeleton is actually driven by the data within directory. When a user starts a session, he sees the available programs from the directory and can decide which one to invoke.

The directory is maintained from the skeleton itself. A skeleton administrator may add/delete applications from the directory. The figure 4 illustrates the layout of the applet for adding/editing new simulation models. To perform these restricted operations a user must present his/her name and the password.

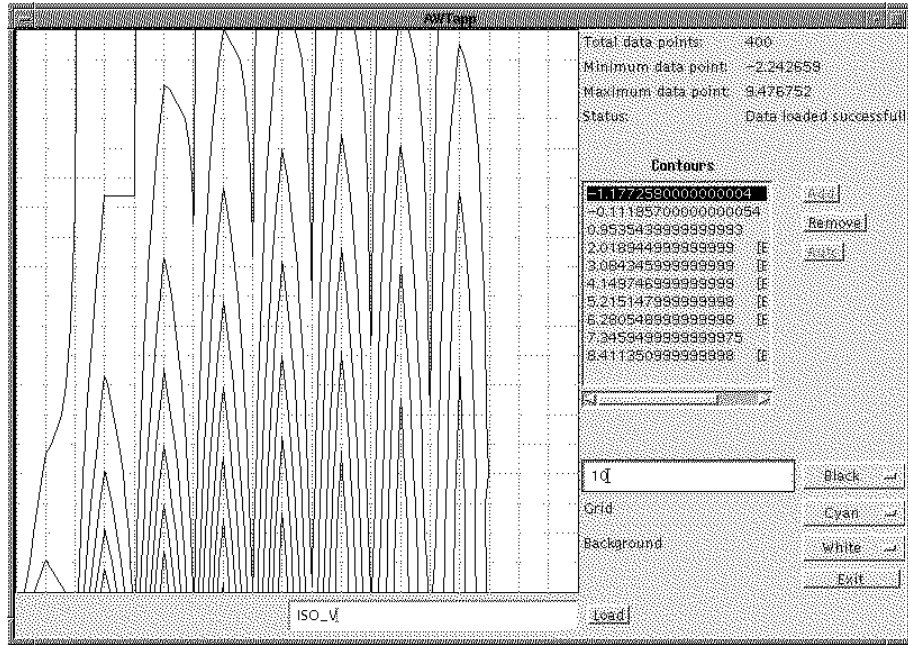


Figure 5: User interface: isolines

3.3.2 The output example

Running a simulation model through Web computing skeleton is an interactive process, where a user may affect and change both the simulation flow and the display of results.

The figure 5 illustrates a layout of the output of one of the FAST simulation programs. It shows isolines in the main window and a number of other information concerning both display (the number of isolines and the colour of isolines, grid and background) and contour characteristics. The window is controlled by the Java applet, allowing user to locally affect the display on it.

There are many advantages of showing the output results with a Java applet: (1) the host server machine is free to do other work; (2) the Internet load is reduced as the code is sent only once and a user may further interact with the applet changing the output shape and content; (3) while viewing the display, a user may start another simulation at a remote machine.

4 Discussion

This paper presents a step-wise approach to the development of complex Web collaborative systems. In the first phase a general-purpose Web computing skeleton is developed as a software environment in which more complex application can be embedded. The skeleton resolves the problems of Web-based collaboration and has a clear interface to embedded programs. In the second phase the actual Web-enabling of a concrete application has to be done. It is deduced only to the development of specific application

interface. Once developed, a new Web-enabled application can be embedded to the skeleton without a need to recompile the system.

The skeleton has been designed and developed with the state-of-the-art software techniques. The Web computing skeleton is:

Heterogeneous – it may run on any computer having Web browser as a universal front-end. Machine endependent code produced by Java interpreter, insures that the skeleton may be used at any computer with Java virtual machine.

Distributed – it runs on several computers connected via Internet using client/server programming paradigm. Contrary to Web-enabled programs that use CGI (Common Gateway Interface) scripts, the skeleton uses sockets and RMI (Remote Method Invocation) for active cooperation during the session.

Object oriented – it is developed in Java and uses advanced distributed object-oriented concepts. The whole skeleton consists of a number of classes and methods that can be remotely created and invoked.

Mobile – it comprizes portable code for clients and for user interface. The skeleton server contains two code types: resident - i.e. actual server code and (2) movable code - the code for remote clients.

Dynamic – it provides a dynamic environment that can be configured in the run-time. Before any Web user visits the server site, the skeleton is a clientless server. Once visited, the skeleton sends the code that makes a Web browser behave as an active client. In other words, it is a server whose first service is creating its own clients.

Extendible – it can be extended without re-programming. A skeleton administrator can add new pre-programmed functionality to the skeleton directory, thus extending its functionality in a run time.

At the current stage, the skeleton can be used as a general-purpose environment for Web-based collaboration. Several systems from different application domains are being prepared for Web-enabling, enriching the Web site of our laboratories.

The further work will focus on extending the distributed features of the skeleton. Though flexible and widely available, Java Remote Method Invocation (RMI) is not sufficient for building distributed platforms. The client-server programming model can be fully employed only by the use of inherently distributed platforms, like CORBA [11]. Recently, there have been some efforts to join Java and CORBA [12] in order to combine Java's heterogeneity and Web-enabled programming with CORBA's rich functionalities. Such a combined approach would significantly enrich the Web computing applications.

Acknowledgements

The authors appreciate the partial financial support provided by German South African cooperative programme in information technology. They would also like to thank Dr. Ekkehard Holzbecher for his constructive help concerning FAST simulation system.

References

- [1] R.T.Kouzes et al. Collaboratories: Doing Science on the Internet, *Computer*, Vol.29, No.8, 1996, pp.40–46.
- [2] Sun Microsystems. *The Java Virtual Machine Specification*, Technical Report, Sun Microsystems, Mountain View, Calif., 1995.
- [3] E. Yourdon. Java, the Web, and Software Development, *Computer*, Vol.29, No.8, 1996, pp.25–30.
- [4] Sun Microsystems. *Remote Method Invocation Specification*,
<http://www.javasoft.com/products/JDK/1.1/docs/guide/rmi/rmiTOC.doc.html>
- [5] Sun Microsystems. *Object Serialization Specification*,
<http://www.javasoft.com/products/JDK/1.1/docs/guide/serialization/serialTOC.doc.html>
- [6] E.Holzbecher. Modelling Software for Groundwater Flow, *Proc. 6th International Conference on Computing in Civil and Building Engineering*, Berlin, July 1995, pp.1225–1231.
- [7] B. Plewe. The GeoWeb Project. Technical Paper, SUNY at Buffalo, 1996;
<http://wings.buffalo.edu/plewe/paperwww.html>.
- [8] A. Koschel et al. A Federation Architecture for an Environmental Information System Incorporating GIS, the Web and CORBA.;
<http://www.fzi.de/dbs/publications/overview.html>
- [9] Esri. GIS Solution for Everyone. Home page.;
<http://www.esri.com/base/gis/index.html>
- [10] M.Niemeyer and C. Oczko, Proposal for a Specification of a Simulation Backplane. Version 1.0, Internal Report, 1996.
- [11] OMG. The Common Object Request Broker: Architecture and Specification. Object Management Group Inc., 2.0 ed., July 1995.
- [12] M.A. Hamilton. Java and the Shift to Net-Centric Computing. *Computer*, Vol.29, No.8, 1996, pp.31–39.