The South African Institute of Computer Science
and
Information Technology

The 1997 National
Research and
Development
Conference

Riverside Sun
Vanderbijlpark
13 & 14 November

Hosted by

Potchefstroomse Universiteit
vir Christelike Hoër Onderwys

The Department of Computer Science and Information Systems
Potchefstroom University for Christian Higher Education
Vaal Triangle Campus

PROCEEDINGS

Edited by L.M. Venter & R.R. Lombard
The South African Institute of Computer Science
and
Information Technology

Proceedings
of the
The 1997 National
Research and
Development
Conference
Towards 2000

Riverside Sun
Vanderbijlpark
13 & 14 November

Edited by
L.M. Venter
R.R. Lombard
Foreword

This book contains a collection of papers presented at a Research and Development conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT). The conference was held on 13 & 14 November 1997 at the Riverside Sun, Vanderbijlpark. Most of the organization for the conference was done by the Department of Computer Science and Information Technology of the Vaal Triangle Campus, Potchefstroom University for Christian Higher Education.

The programming committee accepted a wide selection of papers for the conference. The papers range from detailed technical research work to reports of work in progress. The papers originate mainly from Academia, but also describe work done in and for Industry. It is hoped that the papers give a true reflection of the current research scene in Computer Science and Information Technology in South Africa. Since one of the aims of the conference is Research development, the papers were not subjected to a refereeing process.

A number of people spent numerous hours helping with the organization of this conference. In this regard, we wish to thank the members of the Organizing committee, and the Programming committee who had very little time to screen the abstracts and compile the program. A special thanks goes to the secretary of the department, Mrs Helei Jooste, whose very able work was interrupted by the birth of her first child.
Organizing Committee

Conference General Chairs
Prof. J.M. Hattingh (PU for CHE)

Organizing Chair
Prof. Lucas Venter (PU for CHE)

Organizing Committee
Mrs. S. Gilliland
Mr. J.P. Jooste
Mr. R.R. Lombard
Mrs. M. Huisman

Secretariat
Mrs. H. Jooste

Program Chair
Prof. A de Waal (PU for CHE)

Program Committee
Prof. D. Kourie (UP)
Prof. C. Bornman (UNISA)
Prof. L.M. Venter (PU for CHE)
# Table of Contents

Foreword

Organizing Committee

List of Contributors

**Software Objects Change: Problems and Solution**  
S.A. Ajila

26

**Liming-like Curve Constructions**  
M.L. Baart and R. McLeod

27

**A Model for Evaluating Information Security**  
L. Barnard and R. von Solms

**Integrating Spatial Data Management and Object Store Technology**  
S. Berman, S. Buffler and E. Voges

31

**Metamodelling in Automated Software Engineering**  
S. Berman and R. Figueira

32

**Using Multimedia Technology for Social Upliftment in Deprived Communities of Southern Africa**  
L. Bester and E. de Preez

33

**Extending the Client-Server Model for Web-based Execution of Applications**  
L. Botha, J.M. Bishop and N.B. Serbedzija

36

**Access Control Needs in an Electronic Workflow Environment**  
R.A. Botha

45

**The Use of the Internet in an Academic Environment to Commercially Supply and Support Software Products**  
B. Braude and A.J. Walker

51

**Explanation Facilities in Expert Systems Using Hypertext Technology**  
T. Breetzke and T. Thomas

63

**Theoretical Computer Science: What is it all about, and is it of any relevance to us?**  
C. Brink

75

**Representing Quadrics on a Computer**  
M.A. Coetzee and M.L. Baart

76
The Generation of Pre-Interpretations for Detecting Unsolvable Planning Problems
D.A. de Waal, M. Denecker, M. Bruynooghe and M. Thielscher

The Emerging Role of the Chief Information Officer in South Africa
B. Dekenhah

A Java-Implemented Remote Respiratory Disease Diagnosis System on a High Bandwidth Network
A. Foster

Early Results of a Comparative Evaluation of ISO 9001 and ISO/IEC 15504 Assessment Methods Applied to a Software Project
C. Gee and A.J. Walker

A Neural Network Model of a Fluidised Bed
M. Hajek

The Effects of Virtual Banking on the South African Banking Industry
M.L. Hart and M. Dunley-Owen

Linear Response Surface Analysis and Some Applications
J.M. Hattingh

Model Checking Software with Symbolic Trajectory Evaluation
A. Hazelhurst

A Risk Model to Allocate Resources to Different Computerized Systems
H.A. Kruger and J.M. Hattingh

Returns on the Stock Exchange
J.W. Kruger

Cardinality Constrained 0-1 Knapsack Problems
M.F. Kruger, J.M. Hattingh and T. Steyn

An Investigation in Software Process Improvement in the Software Development of a large Electricity Utility
M. Lang and A.J. Walker

Design and Implementation of a C++ Package for Two-Dimensional Numerical Integration
D.P. Laurie, L Pluym and Ronald Cools

Algebraic Factorization of Integers Using BDE's
H. Messerschmidt and J. Robertson
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Optimization of Routes after the Process of Recovery</td>
<td>176</td>
</tr>
<tr>
<td>M. Mphahlele and J. Roos</td>
<td></td>
</tr>
<tr>
<td>Using a Lattice to Enhance Adaptation Guided Retrieval in Example</td>
<td>177</td>
</tr>
<tr>
<td>Based Machine Translation</td>
<td></td>
</tr>
<tr>
<td>G.D. Oosthuizen and S.L. Serutla</td>
<td></td>
</tr>
<tr>
<td>Information Systems Development and Multi Criteria Decision</td>
<td>192</td>
</tr>
<tr>
<td>Making / Systems Thinking</td>
<td></td>
</tr>
<tr>
<td>D. Petkov, O. Petkova</td>
<td></td>
</tr>
<tr>
<td>The Development of a Tutoring System to Assist Students to Develop</td>
<td>193</td>
</tr>
<tr>
<td>Answering Techniques</td>
<td></td>
</tr>
<tr>
<td>N Pillay</td>
<td></td>
</tr>
<tr>
<td>Combining Rule-Based Artificial Intelligence with Geographic</td>
<td>194</td>
</tr>
<tr>
<td>Information Systems to Plan the Physical Layer of Wireless Networks</td>
<td></td>
</tr>
<tr>
<td>in Greenfield Areas</td>
<td></td>
</tr>
<tr>
<td>K. Prag, P. Premjeeth and K. Sandrasegaran</td>
<td></td>
</tr>
<tr>
<td>A Distributed Approach to the Scheduling Problem</td>
<td>202</td>
</tr>
<tr>
<td>V. Ram and P. Warren</td>
<td></td>
</tr>
<tr>
<td>More readings than I thought : Quantifier Interaction in Analysing the</td>
<td>203</td>
</tr>
<tr>
<td>Temporal Structure of Repeated Eventualities</td>
<td></td>
</tr>
<tr>
<td>S. Rock</td>
<td></td>
</tr>
<tr>
<td>Ray Guarding Configuration of Adjacent Rectangles</td>
<td>221</td>
</tr>
<tr>
<td>I. Sanders, D. Lubinsky and M. Sears</td>
<td></td>
</tr>
<tr>
<td>Developing Soft Skills in Computer Students</td>
<td>239</td>
</tr>
<tr>
<td>C Schröder, T. Thomas</td>
<td></td>
</tr>
<tr>
<td>Information Security Awareness, a Must for Every Organization</td>
<td>250</td>
</tr>
<tr>
<td>M. Thomson and R. von Solms</td>
<td></td>
</tr>
<tr>
<td>Pla Va: A Lightweight Persistent Java Virtual Machine</td>
<td>253</td>
</tr>
<tr>
<td>S Tjasink and S. Berman</td>
<td></td>
</tr>
<tr>
<td>Beliefs on Resource-Bounded Agent</td>
<td>267</td>
</tr>
<tr>
<td>E. Viljoen</td>
<td></td>
</tr>
<tr>
<td>Object-Orientated Business Modelling and Re-engineering</td>
<td>268</td>
</tr>
<tr>
<td>M. Watzenboeck</td>
<td></td>
</tr>
</tbody>
</table>
On Indexing in Case Based Reasoning Applied to Pre-Transportation Decision Making for Hazardous Waste Handling
K.L. Wortmann, D. Petkov and E. Senior

Author Index
List of Contributors

S. Ajila
Department of Mathematics and Computer Science
National University of Lesotho
Roma, 180
Lesotho

L. Baart
Department of Mathematics
Vaal Triangle Campus of the PU for CHE
PO Box 1174
Vanderbijlpark, 1900

L. Barnard
Faculty of Computer Studies
Port Elizabeth Technikon
Private Bag X6011
Port Elizabeth, 6000

S. Berman
University of Cape Town
Rondebosch, 7701

J.M. Bishop
Computer Science Department
University of Pretoria
Pretoria, 0002

R.A. Botha
Faculty of Computer Studies
Port Elizabeth Technikon
Private Bag X6011
Port Elizabeth, 6000

B. Braude
Software Engineering Applications Laboratory,
Electrical Engineering
University of the Witwatersrand
Private Bag 3
Wits, 2050

T. Breetzke
Faculty of Computer Studies
Port Elizabeth Technikon
Private Bag X6011
Port Elizabeth, 6000

C. Brink
University of Cape Town
Rondebosch, 7700

M. Bruynooghe
Departement Computerwetenschappen
Katholieke Universiteit Leuven
Celestijnenlaan 200A
B-3001 Heverlee
Belgium

S. Buffler
University of Capetown
Rondebosch, 7701

M.A. Coetzee
Department of Mathematics
PU for CHE
Private Bag X6001
Potchefstroom, 2520

R. Cools
Katholieke Universiteit Leuven
Celestijnenlaan 200A
B-3001 Heverlee
Belgium

E. de Preez
Faculty of Computer Studies
Port Elizabeth Technikon
Private Bag X6011
Port Elizabeth, 6000

D.A. De Waal
Department of Computer Science and Information Systems
PU for CHE
Private Bag X6001
Potchefstroom, 2531

B. Dekenhah
The Board of Executors

M. Denecker
Departement Computerwetenschappen
Katholieke Universiteit Leuven
Celestijnenlaan 200A
B-3001 Heverlee
Belgium

M. Dunley-Owen
Department of Information Systems
University of Cape Town
Rondebosch, 7700

R. Fiqueira
University of Cape Town
Rondebosch, 7701

A. Foster
Department of Computer Science
University of Cape Town
Rondebosch, 7701

C. Gee
Software Engineering Applications Laboratory,
Electrical Engineering
University of the Witwatersrand
Private Bag 3
Wits 2050
Extending the Client-Server Model for Web-based Execution of Applications

L Botha, J M Bishop  
Computer Science Department  
University of Pretoria  
Pretoria 0002, South Africa  
lbotha@cs.up.ac.za  
jbishop@cs.up.ac.za

N B Šerbedžija  
GMD FIRST  
Rudower Chausee 5  
D-12489 Berlin  
nikola@first.gmd.de

Abstract

In this paper, options for executing applications in a distributed manner on the World Wide Web are examined. An approach to providing a Java-based framework for such applications is presented. The Web Computing Framework has the ability to open and maintain Web connections and provide collaboration for software components over the Internet. The work is part of a wider on-going project whose aim is to make environmental simulation models and geographical information systems (GIS) publicly available to Internet users.

Keywords: Web Collaboration, Web Computing.

1 Introduction

In a previous study[1], the idea of distributed computing over the Worldwide Web was explored. The software produced for this study has been expanded and used in further applications. In this paper we look more deeply into the various options for Web computing and give more details as to how the software operates from the Java side.

In the recent years much effort has been put into the development of World Wide Web technology, primarily providing global access to hypertext. As the use of the Internet spread, applications that are more complex became feasible [2]. With the appearance of the platform independent Java programming language and the virtually universal presence of the Java run-time system [3], the dream of global heterogeneous distributed computing became a reality. This paper investigates the emergence of Web-enabled applications and proposes 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 the collaboration of several remotely located applications, using the World Wide Web as the communications infrastructure.

Contrary to the "classical" distributed components that are static and rely upon a fixed hardware configuration within a local area network, this new approach is based on the use of dynamic functionality that can be easily migrated and executed on any machine connected to the Web.
In this paper, an approach to provide a Web-based framework for the distributed execution of various programs is presented. Section 2 gives an overview of the problem that is being addressed. Section 3 presents a Web Computing Framework that facilitates the type of distributed execution envisaged. Section 4 discusses the results achieved and indicates the directions of further work.

2 Application Structure

Many applications today have essentially the same high-level structure, namely a user interface, some kind of server that the user interface communicates with and possibly some more processes that perform some work on the server side. Most typical two- and three tier client/server designs follow this model (figure 1). An example is any database application, where a user frontend communicates with a database server, or an application server talks to the database.

GIS applications also fall into this model[9]. The GIS database, or geobase, is usually stored remotely because of its immense size. The user interface communicates with an application server that queries the geobase and presents the user with the results.

Another example may be a Web search engine, where the user interface communicates with a database server that performs searches on a database of web pages, while some intelligent robot collects Web page information and continually updates the database.

![Figure 1: The Client/Server Model with respect to the Web](image)

The type of application described here lends itself very well to distribution on the Web, because of the clean separation of the user interface to the rest of the system.

There are essentially two ways of approaching the problem of providing Web access to these applications. One way is to make use of the standard CGI scripting mechanism that Web servers already provide. The second is using a Java-based mechanism with remote objects.
2.1 CGI Scripts

Most Web servers provide the functionality to execute scripts at the server side. Instead of sending a static HTML Web page to a Web browser, scripts that execute on the server side can be used to dynamically create HTML pages, thereby creating Web pages on the fly. The Hypertext Transfer Protocol (HTTP) specification states that a Web page must be transferred from the server to the client in one discrete block.[10]

Functionality for the transfer of data from the client to the server is provided in the form of user-defined tuples of strings. These tuples take the form of <name, value> pairs. This mechanism makes it possible to send simple text-based data back to the server for processing. No provision is made for the transfer of a continuous stream of data in either direction, though.[10]

The server side scripts may be in any executable form — Perl scripts, shell scripts, Tcl scripts or even small C programs can be used. To provide some measure of security, the script usually must be installed on the Web server by the administrator of that server.

2.2 Java Programs

Java applets may be downloaded from a server and execute on the client side. As it is a full programming language, Java provides the capability to manipulate complex data structures and because of its extensive networking support, a client is able to establish a network connection to a server and transfer these complex data structures. The result is more fine grained control over the system.

A Java applet may communicate with a server via:

- standard network sockets;
- remote objects using the Java Remote Method Invocation (RMI) mechanism[4]; or

The penalty one pays in using the Java applet approach is a large increase in effort and complexity, since much programming needs to be done. The client interface needs to be rewritten completely. If the server is written with a known interface, such as CORBA, it can be used without any changes[6], however often a wrapper has to be constructed to establish a convenient protocol between the Java client and the server.

2.3 Choosing between CGI and Java

The most appropriate method of implementation depends very much on the nature of the data that flows between the server and the client interface and the measure of control that the client needs to exert over the behavior of the server.

If the data that flows between the client and the server is simple and relatively small in size, the better option would be CGI scripts. There is no way of ensuring accurate or secure
data transfer, though. Examples of this kind of application are an email client [7] and a simple SQL query interface of a SQL database [8]. If, however, one would like to transfer complex data structures from server to client and vice versa, a Java client and server will be necessary. An encrypted email reader and a GIS query interface [9] are two examples of where such an approach would be very useful.

The second issue is that of the client's control over the server. Some applications are by nature stateless and therefore do not require the client to have well-defined control over the server. Often the client wants to process a transaction, after which it does not want anything else from the server. The CGI script approach works very well for this kind of application. Database queries or sending email messages are two examples. This is often not sufficient, however. As soon as an application requires more than a couple of interactions between the server and the client, one runs into the problem of controlling the state of the transaction. If, for example, a user wants to control the simulation of a neural network on a parallel computer, he needs to be able to constantly receive feedback of what is happening, as well as be able to control the process in a fine grained manner that is virtually impossible to provide with the CGI framework. In this case a Java-based solution works very well.

The CGI-based solution has been explained extensively over the past couple of years and expanding on it is beyond the scope of this paper. Instead, the focus will be kept on the Java-based solution. In the next section, a Web Computing Framework is described that enables one to build such applications.

3 The Web Computing Framework

The Web computing framework is a set of Java classes that provides certain functions to support Web computing, using the Remote Method Invocation (RMI) mechanism for inter-object communications. The purpose of the framework is to provide the basic software structures for Web computing, i.e., it should establish all the necessary connections and enable collaboration between the different remotely located programs. The actual functionality is obtained by embedding application programs in the framework and providing an appropriate interface to these programs.

3.1 Design Principles

The Web Computing Framework is a distributed and extensible software template for enabling existing software to run in the World Wide Web environment. It is designed to allow robust, distributed collaboration of remote objects.

Figure 2 illustrates both the physical layout and logical organization of the framework components. Physically, the framework is distributed over at least three machines, each running different components: a Web browser, a Web server and an application server. By visiting the server sites, a remote Web user automatically obtains the interface code needed for further collaboration. The software downloaded from the servers is indicated...
by the dashed rectangles. The number of application servers within the framework is not restricted.

![Diagram of Web Computing Framework](image)

**Figure 2: Web Computing Framework**

The logical structure of the framework is given in figure 2(b). It contains the following four major components:

- **mobile code**: a set of objects that can be downloaded to an anonymous Web browser and act as clients to the services provided by the servers.

- **server code**: a daemon that waits to be called by its mobile clients. Once called, it performs the task requested: it initiates application programs and interacts with the clients. The server also maintains a dynamic list of servers with descriptions of the available functions that the servers provide. The descriptions may be used by the mobile clients to automatically (or by user intervention, manually) choose the correct server for any request.

- **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 use of Internet users.

The services offered by the framework are actually embedded application programs. Most of these programs are executed at the server site, but it is possible to move a considerable amount of processing onto the client machines. The capability to distribute processing to the remote client, away from the server, is arguably one of the most useful benefits to be gained when using this approach.

3.2 Framework Internals

The internal structure of the framework can be divided into three logical parts, namely the applet running in a remote user’s Web browser, the directory server and one or more application servers. Each of these parts may execute on physically separate machines.

The applets running in the remote user’s Web browser present an interface to the processing that most of the time is taking place elsewhere. The framework has one predefined applet, which the browser downloads from a Web server. The 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 execute 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. 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 class files that can be run in the user’s Web browser as a user interface to the application.

Figure 3 shows the typical flow of control within the framework. 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 client chooses (5) an application to execute and is authorized by the access control system, the initial applet contacts the relevant application server (6) and downloads the new application interface classes (7), using the custom class loader built for that purpose. The new application interface is then started up and control passes to it. The application interface contacts the application server and maintains remote collaboration (8).

The framework does not specify the mechanism used for the remote communication between the application interface and the application server, but rather leaves that up to the person implementing the communication. Any communication method can be used, such as Remote Method Invocation (RMI), CORBA or sockets. The best protocol depends very much on the kind of application that is running and the nature of the data that flows between the client and the server. Using RMI has the advantage that the directory server’s services can be accessed from an application because the directory server uses RMI.
tion of objects that are in address spaces that are physically remote from the calling objects. The fact that RMI is designed specifically to operate in the Java environment means that it is less complex than the alternatives such as CORBA, which must cater for multiple programming languages. This single-language approach allows RMI to take advantage of the Java object model and integrate seamlessly into the Java environment.

Invocation of a remote method with 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 URLs in the form "rmii://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 Virtual Machine, local objects must be passed by copy rather than reference in remote method invocations.

4 Examples

Currently the framework can be used as a general-purpose environment for Web-based collaboration. The domains investigated were environmental simulation models, geographic information systems and frontends to complex parallel computing and artificial intelligence systems. Two systems from different application domains have been prepared for execution on the Web. These are:

FAST, a groundwater simulation system  The FAST system[1] is a collection of several simulation programs used for modelling in an irrigation field, prediction of pollutant concentrations, modelling saltwater intrusion etc. The FAST programs are equipped with a rich graphical user interface. Nevertheless, programs are single-user and PC-based and only available in the laboratory where the whole system is produced.

The FAST simulator is a perfect candidate for execution on the Web. Embedding the FAST system into the framework would provide wide availability of the simulator with low development costs.

To embed the FAST system into the framework two modifications had to be done:

1. Development of a Java-based simulation platform

2. Redesign of the user interface

The first extension belongs to the server part of the framework (ie. the resident code) and the second to the user interface (ie. the mobile code).

A GIS emergency response system  The emergency response system [9] consists of a GIS geobase server, various monitor servers and a GUI client which controls all the
3.3 Java-based Implementation Details

The Web Computing Framework is programmed in Java and consists of two types of objects. The first is a set of mobile Java applets that are prepared to run on a remote machine. The second is non-movable code that resides on the server site waits to be invoked from a remote location. The applets provide a remote interface to the non-movable server code. They may be graphical user interfaces or purely software interfaces that gives other objects access to the functions provided by the servers.

Communication among the framework components is implemented using the Java Remote Method Invocation (RMI) mechanism [4]. The Remote Method Invocation system provides a mechanism for distributed computing by providing referencing and manipula-
processing. All geographic data is taken from one or more physically distributed databases. The servers involved are a traffic monitoring and accident reporting program, an ambulance scheduling system and an ambulance tracker. The output of these programs is collated and displayed on a road map.

The application is inherently distributed and it is therefore quite straightforward to prepare it for Web execution. Preparing the emergency response system for the framework consisted of

1. Development of a Java-based user interface
2. Development of a Java-based software interface to the existing Windows-based program. This consisted merely of constructing a Java wrapper for a Windows DLL that controlled access to the GIS database.

Extensive tests have been done on the performance of this system. Two maps with 1018 features and 7471 coordinates and 1330 features and 18373 coordinates respectively were used as tests. When transferring data as byte streams with integer optimisation (the most efficient) the system took 14 seconds to load the first map. This was with the full RMI and Web computing framework approach. Without RMI, ie. running native code on the client, the time drops to 4 seconds (on a Pentium Pro). Nevertheless, the advantages of having distributed objects with Java and RMI outweigh the performance costs.

5 Further Work

This paper presents an approach of how to use a Web Computing Framework to facilitate the development of complex distributed Web-based systems. The framework is developed as a software environment in which applications that are more complex can be embedded. It resolves the problems of Web-based collaboration and has a clear interface to the embedded applications.

Further work will focus on extending the distributed features of the framework. Though flexible and widely available, RMI is somewhat restrictive in its features and is therefore not sufficient for building highly distributed platforms. The client-server programming model can be fully employed only by the use of inherently distributed platforms, such as CORBA [5]. Recently, there has been some effort to unify RMI and CORBA in order to combine Java’s heterogeneity and Web-enabled programming with CORBA’s rich functionality. Such a combined approach would significantly enrich the Web computing applications.

References


