Proceedings of the IASTED
International Conference

PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS

November 6-9, 2000
Las Vegas, Nevada, USA

Editors: M. Guizani and X. Shen

A Publication of The International Association of Science and Technology for Development - IASTED

Volume II

IASTED/ACTA Press
Anaheim * Calgary * Zurich

ISSN: 1027-2658
ISBN: 0-88986-304-0

Proceedings of the IASTED International Conference
PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS
November 6-9, 2000, Las Vegas, Nevada, USA

A CORBA AND WEB TECHNOLOGY BASED FRAMEWORK FOR THE ANALYSIS AND OPTIMAL DESIGN OF COMPLEX SYSTEMS

CARLOS ARÉVALO, JUAN COLMENARES, NELSON ARAPÉ, NESTOR QUEIPO

Applied Computing Institute, Faculty of Engineering, University of Zulia,
Maracaibo, Zulia, Venezuela

ABSTRACT

This paper presents a framework for the problem of analysis and optimal design of complex systems through a distributed computing environment, and discusses its design and implementation. Examples of the complex systems under consideration include problems in dissimilar areas such as hydraulic fracture optimal design and reservoir characterization in petroleum engineering, optimal placement of electronic components in thermoscience research, and chemical vessel pressure design in mechanical engineering.

The framework allows the coupled execution of heterogeneous software (e.g. simulation, optimization, data acquisition and data analysis programs) and provides a WEB-based user interface. Framework applications use OMG CORBA technology to communicate with each other. The framework provides a CORBA-WEB gateway that receives HTTP requests and converts them to CORBA requests.

The framework was used to find the optimal thermal design of an electronic device. From the results obtained, the framework holds promise of becoming an effective and efficient environment for the analysis and optimal design of complex systems.

Keywords: CORBA, Distributed Systems, WEB-based Software Engineering, Modeling and Optimization.

1. INTRODUCTION

The analysis and design of complex systems usually requires the use of software tools for such tasks as design of experiment, modeling and optimization. Software written for these tasks must be integrated in the sense that the data produced by a program becomes input data for another program. This complicates further if there is an iterative cycle (i.e. an optimization algorithm calls repetitively another program as part of the evaluation of the objective function). In these cases the two programs engage in an interactive loop where one program sends data and waits for results from the other program, perhaps thousands of times.

In a typical scenario, an engineer or researcher has several packages for optimization and modeling and perhaps some custom code, which have been written in different languages and run in different hardware/software platforms. The coupled execution of this software usually requires porting code between platforms and, when this is not possible, moving data from one computer to another through files or the network. These tasks are time consuming and draw away attention to other problems that are not related to his or her interests. Besides, the derived solutions are seldom reusable.

During the last decade technologies such as CORBA, DCOM and, recently, SOAP have emerged and evolved that allow the integration of software across dissimilar hardware/software platforms. However the use of these technologies directly requires expertise in software development, object-oriented programming and distributed computing, and even then is a time consuming task. A solution to this problem is the creation of a workgroup framework, which facilitates the integration of software tools for the analysis and design of complex systems across a network.

Several groups have worked on the development of frameworks for the integration of software across dissimilar hardware/software platforms. Sergey Melnik et al. [5] developed a framework for the integration of heterogeneous software systems in which individual communication protocols, data manipulation languages and data are represented in a generic manner preserving their ontological variety. Michelena et al. [7] developed a CORBA based framework for the design of large, complicated systems, which they have used in the design of various mechanical systems, including a pressure vessel, an automotive hybrid power train and a tracked vehicle. In the area of integrating CORBA and WEB technologies Merle, Gransart and Geib [6] developed CorbaWeb, a gateway between the World Wide Web and CORBA, based on the CorbaScript scripting language.

In response to the above mentioned problems, we have designed a framework based on CORBA and WEB technology for the integration of the software tools typically used for the analysis and design of complex systems. The proposed framework has been designed employing object-oriented techniques and their basic

services have been implemented in Java. Currently, several applications written in C++ and Java have been incorporated. This paper describes the framework's design, functionality and architecture.

The framework was used to find the optimal thermal design of an electronic device, using a neural network application to build a model of the device and a genetic algorithm module to carry out the optimization cycle.

2. FRAMEWORK OVERVIEW

The framework's main goal is to allow users to execute applications deployed over a network using a WEB browser. The user may run a single application or several applications that interact with each other. This is done in the following way:

- The user enters the framework URL. He is presented with a welcome page that has fields to login to the framework. Once the user has been validated, an HTML page is displayed where he can select an application to work with.
- When the user selects an application, a request is sent to the latter to return its user interface (an HTML page to be displayed in the browser).
- The user fills in the information required by the application and clicks a button to execute it. A request is sent to the application to execute. This request includes whatever information the user entered in the browser.
- The Application performs the requested task and returns an HTML page that displays the results on the browser.

This is a simplified case. The application could display several pages and interact with the user in a more complex way if it so desires. It may also enable the user (through its user interface) to select another application to interact with and to enter any information required for that interaction to occur. Once the user has specified what application modules will execute and interact, the applications communicate directly using CORBA. There is no middle process between applications once they start executing.

The requests that come from the WEB client are HTTP requests. On the other hand, the application modules are CORBA objects that use the IIOP protocol. Therefore, the requests must be converted from one protocol to the other. This is achieved by a CORBA-WEB Gateway, which will be described in the next section.

3. FRAMEWORK ARCHITECTURE

The framework has the following components (fig. 1):

- A **CORBA-WEB Gateway module** that serves as a bridge between the WEB and CORBA. This module receives HTTP requests from a WEB browser, translates them to CORBA messages, and sends them to the target application module. If necessary, it translates the response into HTML format and sends it back to the client for display.
- A **user manager module** that handles user access and permissions.
- A **project manager module** that creates and manages user projects.
- **Application modules** that perform specific tasks such as data analysis, modeling and optimization.
- A **naming service** that manages a directory of all framework resources and applications.

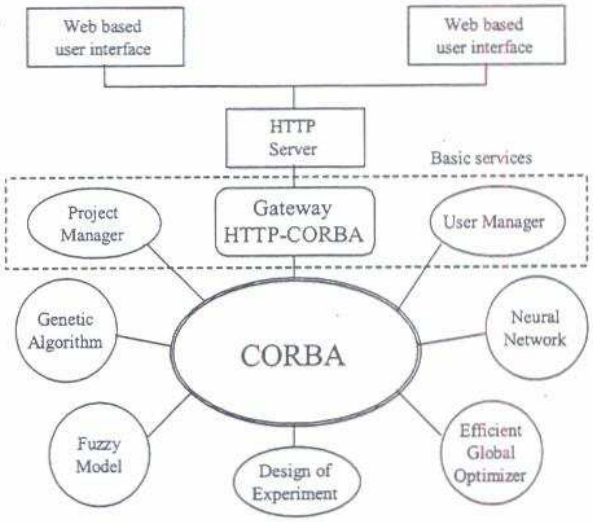

Figure 1. Framework architecture

3.1 The CORBA-WEB Gateway

The gateway consists of a core gateway object and several Java servlets that handle different types of data such as html, text or gif (fig. 2). When the HTTP server receives a request it invokes one of the servlets, which gathers the information needed to execute the request and invokes a method in the gateway core object, passing the name of the target application and the operation to be executed. The gateway then performs the following tasks:

- First, it asks the naming service to retrieve a CORBA reference for the target module.
- Then it calls the target module to obtain the list of parameters required to make the request.
- Next, it creates a dynamic request by means of the CORBA dynamic invocation interface.
- Finally, it performs the request, waits for the result, and once the target application finishes, it returns the output data to the calling servlet, which in turn returns the data to the client for display.

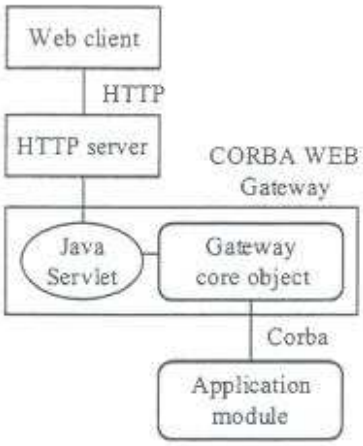

Figure 2. CORBA WEB gateway architecture

For each request made by the WEB client, a new thread is created to service it. Since the CORBA request is always made synchronously, the thread will block until the application returns. However the gateway will continue to process requests due to its multithreaded nature.

The gateway module includes the following objects (fig. 3):

- The dispatcher servlets that receive the requests made by the WEB client.
- A core gateway object. This object receives the request from the servlets and coordinates the construction and execution of the dynamic CORBA requests.
- A request builder. This calls the target object to get its interface definition and builds the request.
- A request invoker that executes the request.

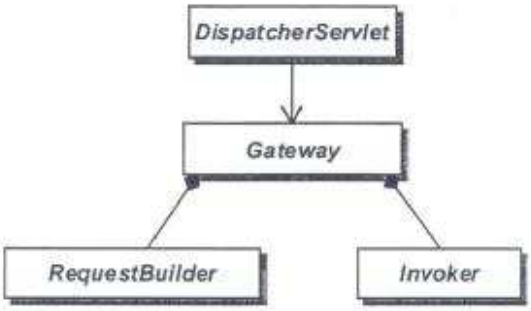

Figure 3. UML class diagram for the CORBA-WEB gateway.

3.2 User manager module

The User Manager Module is a CORBA based user administration tool. It allows an administrator to create and delete framework users, and change their properties through a WEB interface. It also offers an Unix-like user authentication service through the authenticator object.

This tool is implemented in Java as a framework application. This application is started from the administrator user WEB page provided by Project Manager when the administrator logs in.

The User Manager Module class diagram is illustrated in the figure 4.

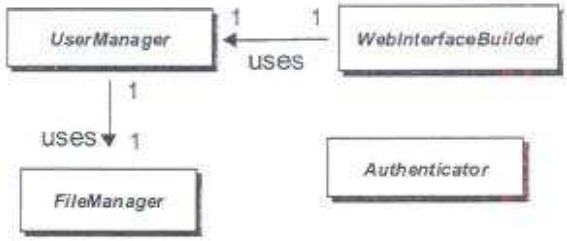

Figure 4. UML class diagram for the User Manager Module.

3.3 Project manager module

The framework has an experiment oriented approach. An experiment is a persistent entity that stores data related with configuration parameters and results of applications. Experiments are grouped in projects. These are stored in a repository that is handled by the Project Manager Object.

The Project Manager is a basic framework service that allows the users to create, rename and delete projects and experiments through a WEB interface. Applications may ask it to store results associated with experiments and retrieve configuration data for their execution.

The project manager WEB interface is the entry point to the framework. It includes fields for user name and password. The user authentication is made invoking a method of the authenticator object that belongs to the User Manager Module. Then, a WEB page is displayed containing links to the user projects and to other dynamically built HTML documents to create, delete and rename projects and experiments.

The Project Manager Repository is mapped on a file system. Projects and experiments are directories where application files are stored. The Project Manager Module is implemented in Java using concurrent programming techniques.

The Project Manager Module class diagram is illustrated in the figure 5.

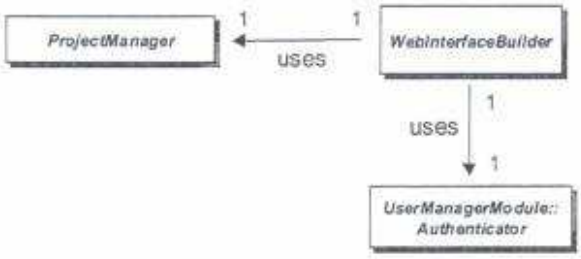

Figure 5. UML class diagram for the Project Manager Module.

3.4 Naming Service

The naming service manages a directory of all the framework resources, organized as a hierarchical tree which follows a structure similar to the application interface hierarchy described in the next section. This service was implemented directly with the CORBA naming service.

3.5 Application modules

The Application modules are the most important part of the framework, as they are the resources the end user is looking after. These modules perform the particular tasks for which they are intended and some additional operations that relate to the framework itself, such as returning the HTML pages that make up the module's user interface, and initializing a user session. The modules are also responsible for registering themselves with the framework name service.

An application module is comprised of one or more CORBA objects. We have developed a set of CORBA interfaces that these objects should implement. The purpose of these interfaces is to standardize the call protocol that clients must use to invoke operations on the objects. The interfaces are organized in a hierarchical tree illustrated in figure 6.

Below we show a fragment of the framework interface definitions.

```
typedef sequence<double> DoubleVector;

typedef sequence<DoubleVector>
      DoubleMatrix;

interface Application {
  void InitWorkSession(
            in string userName,
            in string projectName,
            in string experimentName);
  string getUserInterface();
};
```

```
interface Function
{
      DoubleVector evaluate(
            in DoubleVector inputData);
      DoubleMatrix multiEvaluate (
            in DoubleMatrix inputData);
};
```

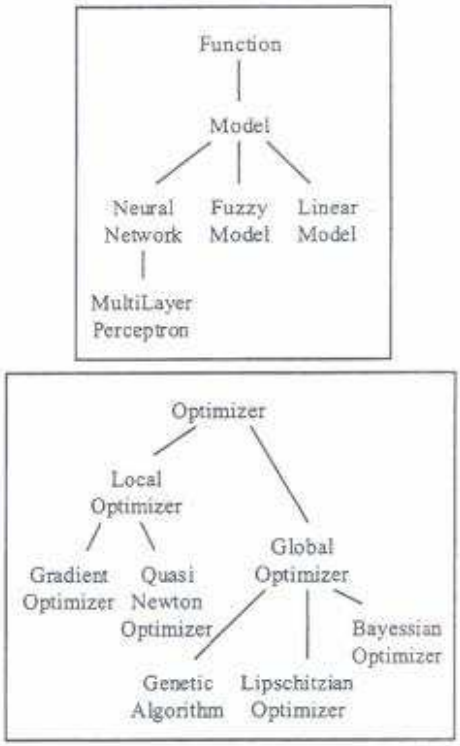

Figure 6. Application Module Interface Hierarchies.

An application module may have objects that implement different interfaces. If the module is to have a WEB based user interface it should have an object that implements the `Application` interface.

Usually, one of the first messages an application module receives is a request to retrieve its user interface. This is an HTML page that will be displayed by the WEB client. This page should have at least one action button that invokes one of the framework servlets. The page also contains the name of the target application and the operation to be invoked. When the user selects the execute button the operation is invoked on the application by the mechanism outlined previously. There may be several execution buttons. An example could be a neural network application that has a button to train the network and a button to execute it, or a genetic algorithm program that has a button to create a GA with some features and a button to run the optimization.

To ease the process of integrating an application into the framework, we have designed an object that will make

the task of creating dynamic HTML pages simpler. The object is named `UserInterfaceManager` and is implemented in Java. An application module can create a `UserInterfaceManager` (using a factory object), configure it, and use it to create dynamic HTML pages.

At the moment the framework has the following application modules:

- An optimization module based on genetic algorithm. This module has a CORBA object that implements the `GeneticAlgorithm` interface. The implementation is based on the Banda genetic algorithm developed by Brian W. Bush [1].
- An optimization module based on the Efficient Global Optimization algorithm developed by Donald R. Jones et al. [4]. This application has a CORBA object that implements the `BayessianOptimizer` interface (derived from `GlobalOptimizer`). The code for this model has been completely developed internally.
- A neural network module. This module implements two CORBA objects, `NeuralNetworkTrainer` and `NeuralNetworkModel`. It allows the creation of multi layer perceptron neural networks, which can be trained using various training algorithms. This module is a wrapper to the Stuttgart Neural Network Simulator [11].
- A fuzzy model module. This module implements two CORBA objects, `FuzzyModelBuilder` and `FuzzyModel`. It allows the creation of Sugeno type fuzzy models using a fuzzy discretization method and an orthogonal parameter estimation [13]. The code for this module has been written internally.
- A linear model module. This module implements two CORBA objects, `LinearModelBuilder` and `LinearModel`. It can be used to create linear models using classical techniques such as linear regression and stepwise elimination. The code has been written internally.
- A design of experiment module. This module creates designs of experiments based on simple random and latin hypercube sampling. It implements the `DesignOfExperiment` interface and is based on internally written code.

4. CASE STUDY

The framework was used in the problem of finding the optimal thermal design of an electronic device. The device is composed of an electronic component embedded in a polyurethane substrate. The heat generated by the electronic component is conducted through the substrate to the device surface, where it dissipates into the air by convection. We wish to find the set of design variables that minimize the temperature in the electronic component.

This optimization problem was part of a larger project previously conducted [10] and was carried out using stand alone applications that run under Microsoft Windows and Unix. We used input-output data obtained from a code that modeled the thermal behavior of the device. The procedure to get the optimal device parameters was as follows: first, a neural network simulator application was used to train a neural network. Then the weights that constituted the resulting network were extracted from the files created by the application. Finally, a subroutine was written to evaluate the neural network as part of the objective function for the genetic algorithm.

The same task was carried out using the framework. First, we created a neural network based surrogate model for the device using the neural network module. Then we configured the genetic algorithm module to use the neural network model as the objective function and used it to get the optimal design.

The results were similar to those obtained in the original work. However, doing this task using the framework was considerably easier. The whole experiment was setup interactively, using the WEB based user interface, and all applications were executed in the computer where they were originally installed.

5. FUTURE DEVELOPMENTS

In the near future we intend to include an additional data base service that will allow access to relational databases and two application modules, a Quasi-Newton local optimization module and a Kohonen neural network module.

6. CONCLUSION

This paper described the implementation of a framework for the analysis and optimal design of complex systems based on CORBA and WEB technology. The framework was used in the problem of finding the optimal design of an electronic device. The applications performed adequately and the framework proved to be flexible enough to setup interactions between several applications using the WEB based user interface. With the inclusion of additional modules for local optimization, and other modeling techniques and simulations, this framework holds promise of becoming an effective tool for the optimal design of complex systems.

BIBLIOGRAPHY

[1] B. W. Bush. "Banda Java Packages. Version 8.1". http://www.sladen.com/bwbush.

[2] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[3] M. Henning and S. Vinoski. *Advanced CORBA Programming with C++*. Addison Wesley. 1999.

[4] D. Jones, M. Schonlau and W. Welch. "Efficient Global Optimization of Expensive Black-Box Functions". *Journal of Global Optimization*, 13, 1998, 455-492.

[5] S. Melnik. *Generic Interoperability framework*. Working Paper. Stanford Digital Library. HTTP://www-diglib.stanford.edu/diglib/ginf/wd/ginf-overview/

[6] P. Merle, C. Gransart and J.M. Geib. "CorbaWeb: A WWW and Corba Worlds Integration". Presented at the 2th COOTS, Workshop on Distributed Object Computing on the Internet, Toronto, Canada, June 1996.

[7] N. Michelena, C. Scheffer, R. Fellini and P. Papalambros. "A CORBA-based object-oriented framework for distributed system design". *Mechanics of Structures and Machines*, 1999.

[8] Object Management Group. *CORBA Scripting Language. Request For Proposal*. OMG Document: orbos/96-06-13.

[9] P. Papalambros, N. Michelena and N. Kikuchi. "Distributed Cooperative Systems Designs", *Proceedings of the 11th International Conference on Engineering Design*, 2, 1997, 265-270.

[10] N. Queipo, C. Arevalo and S. Pintos. "The Integration of Design of Experiments, Surrogate Modeling and Optimization for Thermoscience Research". *Proceedings of the ASME Winter Annual Meeting, Session HTD-58 (Multidisciplinary Inverse Problems and Optimization in Heat Transfer)*. 1998.

[11] University of Stuttgart. Institute For Parallel And Distributed High Performance Systems (IPVR). *Stuttgart Neural Network Simulator Users Manual*. http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/UserManual/UserManual.html.

[12] Vinoski S. "Integrating Diverse Applications Within Distributed Heterogeneous Environments". *IEEE Communications Magazine*, 35(2), 1997.

[13] L. Wang and R. Langary. "Building Sugeno Type Models Using Fuzzy Discretization and Orthogonal Parameter Estimation Techniques". *IEEE Transactions on Fuzzy Systems*, 3(4), 1995, 454-458.