7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 27 – 29, 2004

INTERNET BASED DISTRIBUTED METHODS IN SUPPORT OF REMOTE AND COLLABORATIVE DESIGN

Marius PISLARU[1], Alexandru TRANDABAT[2], Cornelia VARGA[3], Stefan HANGANU[4]
"Gh. Asachi" Technical University of Iasi
Bd.Dimitrie Mangeron nr.53, RO-70050 Iasi
[1)] *mpislaru@ee.tuiasi.ro,* [2)] *ftranda@ee.tuiasi.ro,* [3)] *cornelia@ee.tuiasi.ro,* [4)] *stefan.hanganu@conexgrup.ro*

Abstract. *The main objective of the article consists in setting up the frame for an efficient trans-institutional internet based environment for promoting cooperative research, development and training activities among partner institutions, especially in the design of dependable tools and systems in support of quality assurance, control and management in engineering practice, the most dynamically developing fields nowadays. The main results of the project will be the establishing of a new type of active research and development network of cooperative actions in design, research and teaching activities, realized by joining the resources of all partners and the environment of an internet-based system.*
Keywords*: remote sensing, distributed measurements, CAD dedicated tools, virtual instrumentation, expert systems, neuro-fuzzy models, digital signal processing,*

Introduction

Hence, a synergistic link can be created between partners, which are performing an impressing activity in different but related research and teaching activities, such as: microelectronics design, remote sensing, distributed measurements, CAD dedicated tools, virtual instrumentation, expert systems, neuro-fuzzy models, digital signal processing, all of them in support of reaching new dimensions in the quality and dependability of tomorrow's control and management systems. This link can provide also new opportunities for increasing the design quality in terms of higher dependability of the created systems and reduced time-to-market implementation. This overview propose to insurance the basis for a new quality in cooperative research by facilitating an immediate exchange of information, sharing of software tools and resources (computing power), enabling joint work on research projects and practical design, providing access to libraries, benchmarks and serving as a source of information, i.e. in the field of innovation or standardization, not only for the involved partners, but also for any other interested person and institution, governmental or community body, NGO or academic, with peculiar focus on national industries, with special emphasis on SMEs The final purpose will be the direct improvement of human potential in the field of product quality and safety, including protection of the environment by virtual laboratories of expertise and, as result, a perspective contribution to the welfare of society and to the conformity assessment needed to facilitate international trade in the fields of electricity, electronics and associated technologies

Research methods and implementation

The Internet-based tools environment for advanced systems is a great opportunity for partner sides to realize the infrastructure for future applications development. This overview is a research initiative aiming to make possible a user-transparent distribution of resources over computer networks. It can be divided in three parts:

• a Framework of reusable software, composed by modules and design data representation primitives;
• a web based design environment, implemented over the framework foundations, together with a Service Space, which provides the necessary control on the distribution of resources and the data sharing among partners;

• a Communication Channel, which allows synchronous and asynchronous interaction among the designers. The modules can be distributed over nodes of a Internet Protocol based network. The designers interact with all of the modules using a Java-enabled client software, e.g. a web browser. The Framework Server is responsible for provide the designers with a list of all the automation tools and services available in the network. The designer uses the so called 'Tool Launcher' to execute any of the tools by downloading its building blocks from the Framework and initiating the Java Virtual Machine (JVM) embedded in the local client software. This model ensures the platform independence of the environment, because the tools can be executed without any kind of modification in a wide range of hardware/software platforms.

The tools can be executed completely on the client side, as a separate thread of execution within the JVM, or they may be executed in the server side, using the client software only as interface. The second option is particularly interesting when no platform-independent tool is available, or when the tool requires intensive computation so that the executable code translation from platform-neutral to platform-native becomes too costly, Figure 1.

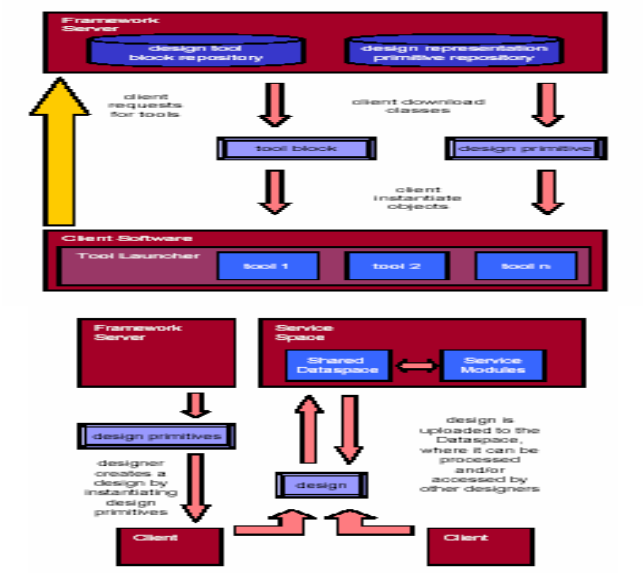

Figure 1. Framework server concept

The Service Space provides facilities for the integration and accessibility of executable service modules distributed in different servers. It uses the concept of shared data-spaces: instances of the framework design representation primitives are kept persistent, so they can be accessed by several users and processed by different service modules.

The Communication Channel support for collaborative work was included in the abstract classes within the design representation framework, and the behavior of the collaborative sessions is controlled by a collaboration service in the Service Space. Currently, the collaboration support should include the following features:

• separation of concepts: the design semantic and its graphical/textual representation are modelled by different objects, in order to allow several visualizations - by different designers - from a single design block;

• update/notify mechanism: a 2-way update/notify mechanism will be implemented, to grant consistency between the design semantic and its representations.

The research includes the actual approaches in the area of World Wide Web based design environments, by the use of a widely known user interface – a web browser – and the possibility of remote access, mainly remote execution of applications. In this case, the partner acts as a data provider and analyzer, with a low degree of interactivity. Further, with the arise of platform independent programming solutions such as Java programming language, some research efforts will be done to distribute the data processing to both sides of the network: remote and local machines. The final project must permit maximum access, so its architecture should be based on the distribution of the resources between client and server sides of the network. In order to define the automation tools distribution over the network, the tools are divided in two groups, regarding the level of interaction of the designer with each tool. Highly interactive tools are loaded from network and executed on the client side, while poorly or non-interactive tools can be executed remotely on the server side, Figure 2.As shown on Figure 2, the environment supports several servers and clients. On the server side, the performed functionality includes the storage of the tools,

design data and the hypermedia structure of the environment; here the non-interactive tools are executed, as mentioned before. On the client side, the interactive tools are loaded from the server and executed. Some design data can be temporarily stored. The non-interactive tool invocation and result analysis is also done from the client side. The distribution of the processing load among the servers is an interesting feature, since it is transparent for the user. So, the design environment may be projected to execute the heavier tasks on the computationally better machines. On the other hand, the client side can be a simple machine, running only a web browser, which is the requirement for a WWW based environment.

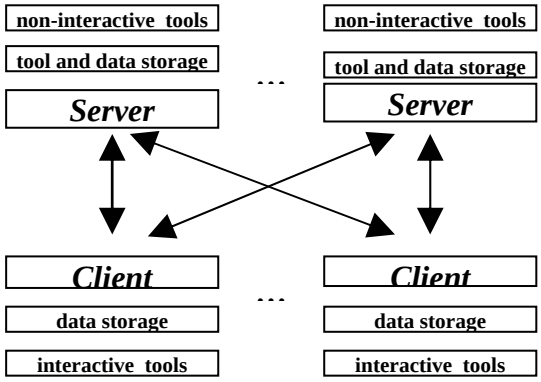

Figure 2. Projected client-server structure

For a really useful environment, it is mandatory to provide the methodology and project management support. These needs are maximized when dealing with distributed teams, which is one of the motivations of the WWW based design environments.

A design environment must provide efficient support for design methodologies. It means that the environment must allow the designer to specify the design flow in a higher level of abstraction, without having to deal with data representation formats and tool invocation parameters. Following this concept, the design environment should be able to translate the input from the designer – which is a sequence of real tasks, such as "edit the counter X of the logic block Y and simulate it" – into a sequence of the design tools. After the definition by the designer of the design tasks, the design environment should provide the tools invocation at the right order and also deal with the data availability for each tool. In some cases, the data conversion may also be needed, so the environment should deal with that task, too. Sometimes, the design tasks definition may be difficult at an early stage. So, the design environment should allow the definition of abstract tasks, which will be specified later. It is desirable that the design environment could even suggest to the designer one or more design flows, starting from a given task. The proposal for alternative paths, for the user defined design flows, is also an interesting feature to be implemented on a design environment.

In order to allow the design flow modeling, it is necessary to define a tool-tool integration architecture. The architecture must provide facilities to model from simple tool connections to complex design methodologies. To take advantage on the hyperdocument-centric approach of the project, the design flow can be modeled as a chain of hyperdocuments. Each one of the hyperdocuments embeds the user interface of each of the design tools. So, the hyperdocument links connect the tools in the right order, allowing the user to navigate across the design tasks. This chain should be done previously, before the design process start, based on the task sequence entered by the designer. So, a tool is necessary to collect the task sequence information from the designer and to generate and store the hyperdocument chain. The storage is done on the hypermedia server, so the hyperdocuments would be downloaded as needed by the designer. The chain may be also dynamically edited, when the designer wants to transfer design information from one tool to another, during the design process, in order to follow alternative flows. The implementation of this architecture requires some resources to be added to the design environment. The first, as mentioned, is a tool for design flow definition, through a graphical user interface, by the user. This tool is also responsible for the hyperdocument chain generation and storage. The second resource is a requisite for the functionality of the first one: a tool information database. To allow the translation from task list into tool sequence, the design flow editor must

have complete knowledge about the design environment tools, its input and output data formats and its relationships with the design tasks.

The third resource to be added is the only one who may cause changes on the model. It is a module to be added to every design tool, allowing the hyperdocuments of the generated chain to configure the design tools on invocation time.

This resource is necessary, because the design flow information is stored on the hyperdocuments, but the tool during its invocation must access it in order to setup its functionality to fit the design flow requirements. For instance, the download of the design data is done by the tool and the information about the data storage network location is stored on the generated hyperdocument.

The main point on this architecture is to keep the focus on the document: the designer is saved from the tasks of keeping track on the design data or even choose a tool to visualize that data on any of the design steps. It is only needed to follow the previously created hyperdocument chain, so the design data will be automatically transferred from step to step and the right tools will be downloaded in order to visualize and edit that data, Figure 3.

Some cautions are needed, in order to conceive an interactive and evolutionary environment, which permits quick modifications, if necessary, due to some possible causes:

- if some modules are to be executed both in the client and server sides of the network;
- when the hyperdocument-based tool-tool integration, in spite of its flexibility and possibility of dynamic tool chain creation and simple maintenance, may allow insufficient tool-tool binding. The integration may be too loose and inefficient in some cases, mainly when repeated iterations are necessary or when complex multi-view design blocks are involved;
- if an object-oriented model should substitute the file-based data storage and transmission. The object-oriented model would allow better integration with the design tools and support efficient design data storage, using a unified data model. The normal file-based approaches require an intense use of format converters, due to the lack of a unified model.

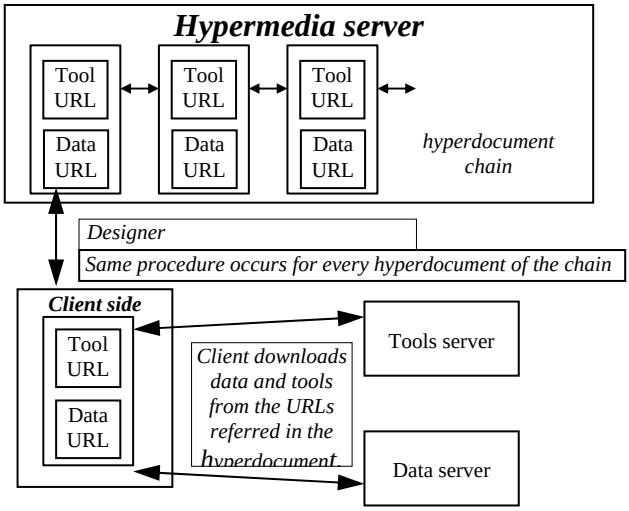

Figure 3. Tool-tool integration architecture

In figure 4, the general structure for the design environment is briefly presented. On the server side of the network, two repositories of Java classes for pre-defined design blocks are playing a similar roll as the design libraries on current design environments.

On the client side, the user invokes a design tool through the web browser. The invocation is done by downloading a hyperdocument, which assembles the tool by instantiating the building blocks from the framework repository. When the design task is finished, the user can update the design representation class repositories and/or store the design as persistent objects on Java Spaces – also on the server side of the network. Typically, the first procedure would be done to add new reusable design blocks to the repository, while the second one would be taken to ongoing design storage. Both the class repositories and the design are available to other users over the network. It is important to notice that any tool development can be done in a similar way. Tool designers can add new tool blocks on the framework repository by adding the classes which will initiate the tools.

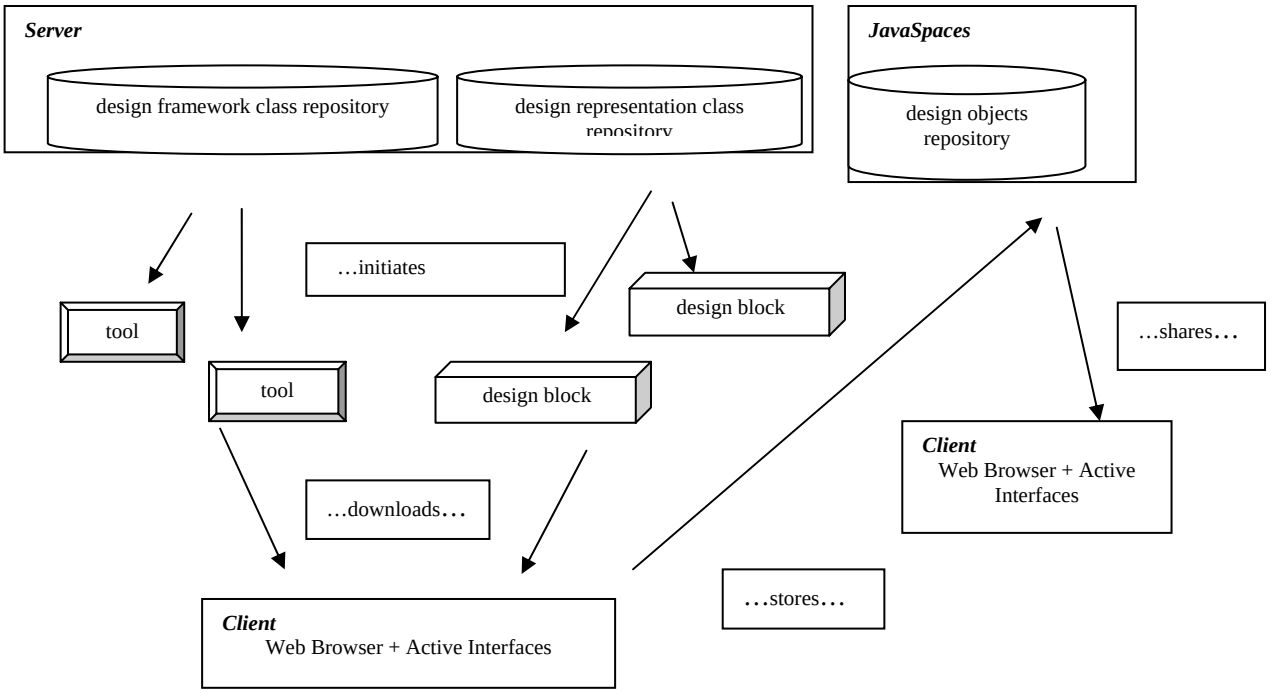

Figure 4. Design environment

Conclusion

The feasibility of such a solution has already been tested, see literature, and this approach has been followed in a collaborative manner by two different research groups, with the aim of creating an international knowledge base, accessible from the students of both countries teams. This is the first step through the building of a common educational background by remotely sharing information and instrumentation among the students and the researchers of the involved Universities. Nowadays, the project involves experimentally the Technical University Gh. Asachi, Iasi, Romania, and the T.U. Darmstadt, Germany, leading to a common teaching method based itself on the reciprocal validation of the student knowledge and on a continuous know-how exchange

References

[1] U. Mayer, J. Becker, T. Hollstein, M. Glesner, L. S. Indrusiak, R. Reis (2000) *An Internet-Capable CAD suite for the Multi-Level Design of Complex Microelectronic Systems*; Design Automation and Test Conference in Europe 2000 (DATE 2000), User Forum, Paris, France

[2] J. Becker, U. Mayer, M. Glesner, L. S. Indrusiak, R. Reis (2000) *Providing Flexible Internet-Infrastructure for FPGA-Based CAD Courses*; 3rd Europ. Workshop on Microelectronics Education (EWME´2000), Aix en Provence, France

[3] T. Kuhn, W. Rosenstiel, U. Kebschull (1999) *Description and Simulation of Hardware/Software Systems with Java*. In: 36th DESIGN AUTOMATION CONFERENCE, New Orleans, USA. p.790-793

[4] R. Helaihel, K. Olukotun (1997) *Java as a Specification Language for Hardware-Software Systems*. In: Proceedings of the International Conference on Computer-Aided Design (ICCAD), p. 690-697.

[5] L. Francis Chan, M.D. Spiller, A.R. Newton (1998) *WELD – An Environment for Web-Based Electronic Design*; Proc. of 35th Design Automation Conference (DAC´98), pp. 146-152, June 15-19, San Francisco, USA.