34 COLLABORATIVE INFRASTRUCTURE FOR DISTANCE- SPANNING CONCURRENT ENGINEERING

Paweł Fraś, Tomasz Kostienko, **Jarosław Magiera,** Adam Pawlak, Piotr Penkala, **Dariusz Stachańczyk, Marek Szlęzak, Maciej Witczyński**
Silesian University of Technology, Gliwice, POLAND
pawlak@ciel.pl

Design of complex Systems-on-a-Chip requires often integration of large teams of engineers who work in remote locations. Seamless and secure integration of distant tools across different organization over the Internet is still one of major challenges that obstacles efficient engineering collaboration. The paper explains the main element of the advanced collaborative infrastructure that supports integration of distributed engineering tools. It constitutes TRMS (Tool Registration and Management Services) which are scalable, easily accessible, secure, and available on different platforms due to their implementation in Java. Requirements on TRMS, its architecture and functionality, as well as conclusions stemming from first deployments are presented.

1. INTRODUCTION

Concurrent engineering for distributed product development is a new paradigm of engineering work that has become feasible due to dynamic progress in information and communication technologies. Engineering workstations due to broadband access to the Internet turn into global workbenches. This new engineering paradigm can be enabled with new innovative infrastructures, net-aware tools, and new design methodologies based on re-use in combination with advanced security and network and tool management. Many research efforts were directed towards collaborative infrastructures during recent years. However, a few research groups have targeted research on applications in electronic system design automation (EDA) only, like WELD (Chan, 1998), REUBEN (Lavana, 1998), ASTAI(R) (ASTAIR, 2003) and MOSCITO (Schneider, 2003).

The EDA community has been inspired by the WELD (Web-based Electronic Design) project that aimed at providing reliable and scalable connection and communication mechanisms for distributed users, tools and services. The developed WELD system comprises principally remote servers, network services, and client applications. *Remote servers* enable access to remote command-line tools

encapsulated by server wrappers or tools with built-in support for a socket connection and WELD communication protocols. *Network services* such as: distributed data manager, proxies and registry services, allow incorporation of infrastructure components on demand. *Client applications* use the WELD infrastructure to access network resources. Clients are either Java browser clients, or generic clients developed in socket-enabled languages using WELD protocols.

REUBEN (Lavana, 1998) (Reconfigurable and reUsable Benchmarking Environment) is the asynchronous and synchronous collaborative OmniFlow environment, where OmniFlow is a universal client, which creates GUI based on cdtML (extended XML) description of a task flow. Communication among remote tools is transparent to a user and it is based on the TCP protocol or a socket-based solution. CVS (Concurrent Versioning System) is used for management of distributed data.

Another approach presents the ASTAI(R) environment which was developed by the C-LAB laboratory from Paderborn, Germany. ASTAI(R) consists of two types of applications. Firstly, Tool Server (TS), which enables ASTAI(R) users to use shared tools. The access to these tools is through the Client application. Communication between elements of the system is based on the CORBA standard. This solution is a problem for networks protected by systems of firewalls or proxies. The ASTAI(R) environment comprises sophisticated Workflow Management System (WfMS) which enables automation of tasks and data flows. ASTAI(R) GUI presents a design process as a number of related operations called workflow activities, whose dependencies are controlled by the WMS. This environment allows distributed engineering team work as sub-workflows may be easily defined. Security aspects are referred to the operating system. ASTAI(R) does not encrypt data transmission. Also, control of user activities and access to shared tools are not directly supported.

An interesting solution is demonstrated in the MOSCITO system. It is an Internet-based multi-agent system which can be controlled and observed by the user's front end program – the MOSCITO desktop or web client. MOSCITO consists of three software layers: kernel layer, interface layer, and user extensions. The kernel provides functionality for basic object and data management, file handling, XML processing and communication. The Interface layer provides programming interfaces for integrates new tools, new workflows, and appropriate viewers. Each interface is represented by a Java class which contains basic functionality. Communication between tools (agents) is organized as a socket-based solution. MOSCITO does not support data transport encryption and proxy servers.

Existing distributed collaborative engineering infrastructures do not support adequately the integration of dispersed design groups and their tools, and do not consider distance-spanning related issues, like: firewalls, security (including user authorization, data and transfer encrypting), distributed inter-organization workflows, and remote administration of users and tools.

The paper presents the Tool Registration and Management Services (TRMS) developed within the EU E-Colleg project (IST-1999-11746) (Bauer, 2001). The TRMS system offers to dispersed engineers some additional features that enhance existing distributed collaborative engineering infrastructures (Mueller, 2003).

The paper is organized as follows. Firstly, requirements on TRMS are enumerated; secondly, the overall architecture and operation of TRMS are

described. Finally, the TRMS deployment is addressed and potential benefits of TRMS use are envisioned.

2. REQUIREMENTS ON TRMS

Remote tool integration is one of central problems of concurrent engineering. Although this is an old problem with a history of many international efforts (Indrusiak, 2002) like CAD Framework Initiative, interoperability of tools dispersed in the Internet still is a challenge. Below, the main requirements on TRMS are enumerated:

- Dynamic tool and user account management
- Information on registered tools should be accessible Internet-wide.
- Tools should be registered in one central repository
- Encryption/decryption of messages and data
- Authentication through a digital signature
- User authorization at the central server
- Flexible though secure mechanism for handling user privileges
- Secure access to tools behind a firewall.

The above list demonstrates that the requirements with respect to security were defined with a particular care. It was assumed that unquestionable confirmation of identity of actors operating in TRMS is a must, and it ought to be realized in the *authentication* process. An extent of granted privileges to TRMS resources will be verified in the *authorization* process. The *access control* process need to be directly responsible for the execution of granted privileges and a definitive decision about facilities of resources. TRMS deals with sensitive data. For this reason exchanged data should be protected in a special manner during transfer through the network. Additionally, *confidentiality* and *integrity* of data should be guaranteed. Mechanisms which will force a definite behavior of users and control over correct functioning of accessible services are an essential complement to security requirements.

3. ARCHITECTURE OF THE TRMS ENVIRONMENT

Figure 1 presents the general architecture of the TRMS environment. In the oversimplified perspective, it comprises three components, namely: the GTLS server, the Tool Server(s) and the Client(s).

3.1. GTLS – Global Tool Lookup Server

The main component of TRMS is the GTLS server. It is responsible for registration and modification of data on users and their privileges, elements of the system, as well as, information on accessible tools and machines that make them available. GTLS is also responsible for the security policy of the whole system, registration of user activities and of access to tools, maintaining statistics, identification of an intruder attack. Furthermore, GTLS is responsible for the generation of keys used

for the encryption of a transmission and the generation of digital signatures (Magiera, 2003).

GTLS uses a relational database to store information about users and native XML database for storing information about registered tools, tasks and workflows.

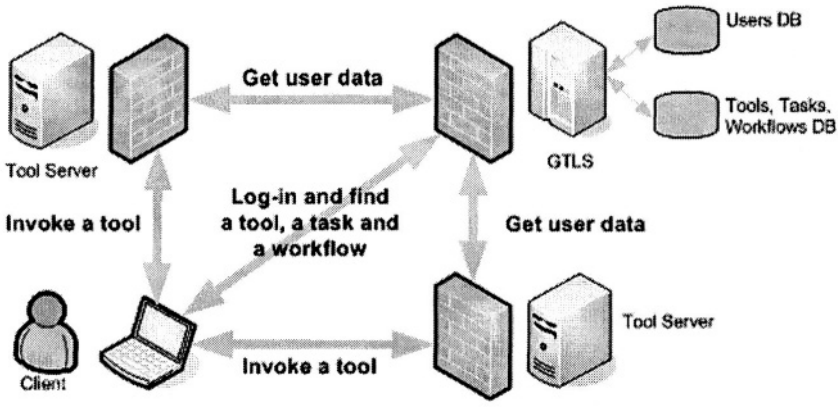

Figure 1 General architecture of the TRMS environment.

3.2. Tool Server

The Tool Server (TS) is responsible for controlling users' access to tools. A client invoking a tool does it through the Tool Server. Tool Server allows for sharing of the tool through the Internet. Its additional task is brokerage in user authentication. The Tool Server queries GTLS whether a user who invokes the tool has sufficient privileges. With the positive verification of the signature and the respective tool is launched.

3.3. Client

The Client application completes the TRMS system. It has a simple GUI that allows for login to the system, its administration and usage of available tools.

In the general case, all TRMS components are on separate machines connected to the Internet. Thus, there is a need for a communication between them. The security policy of the E-Colleg partner companies motivated the use of the HTTP/SOAP communication protocol. It was natural to use XML as a data transfer format. Of course, all sensitive data are encrypted and digitally signed by a sender.

TRMS uses ANTS (Advanced Network Transport Services) for transporting all data between clients and remote tools. The developed by C-Lab ANTS component is a SOAP Web service that enables routed data transport through gateways and firewalls. ANTS can also be used to distribute SOAP messages across network boundaries. This enables TRMS to be deployed independently from the network infrastructure if ANTS connectivity is supported (Mueller, 2003).

4. TRMS OPERATION

TRMS allows a user to invoke a tool with allowed parameters based on a description of the tool in the central repository on GTLS. All operations in the TRMS environment are invoked in a safe way by use of asymmetric encryption, verification of authenticity, digital signature and checking permissions.

4.1. User login to GTLS

Firstly, a user needs to initiate his/her own session at TRMS. So, the user needs to login at the central GTLS server. The user login and a password are sent by the Client to GTLS as an XML message. Authorization data being sent are encrypted by a combination of a public key, and a temporary symmetric key. The message is signed by a private key of the user in order to guarantee an additional authentication.

GTLS, upon receipt of the message, decrypts it using its own, secret private key. The deciphered message contains the user's login and password. These parameters for user authorization are compared with data in the users database (DB) of GTLS. Firstly, the user is identified by the login name, secondly the password is verified. If either user login or password are not correct the login operation is stopped, and the user gets appropriate information. If these data are correct, a public key from the user's DB is fetched, and the signature is verified.

If login, password and the digital signature are correct then a session is established for the user. A unique number is assigned to this session. The user is allowed to launch one session only. Data on the session together with complete data on the user (public key, privileges) are placed in the cache (e.g. memory cache) of GTLS.

Thus, if the user makes subsequent connections to GTLS (tool search, tool registration, etc) or to TS (tool invocation) the user only needs to provide a unique session number. This number allows identifies the user's session in the GLTS cache. Here we have session data (login time, operations done, etc), as well as user data, including the user's public key. With the use of this key a digital signature of a SOAP message is verified in order to check the authenticity of the user.

As soon as the login action is positively verified, a session is initialised and its number is assigned to the user. This unique session number is added to each message being sent. This number allows for the identification of the user's public key. The last one is used by GTLS and TS to verify the correctness of the digital signature.

4.2. Tool search

If the user is already logged in, he may search for a required tool, providing search criteria using a convenient GUI. These search criteria are integrated into the XML message that incorporates the user session number in the SOAP message. The message is encrypted using the public GTLS key and the generated symmetric key and signed (digital signature is done based on a secret private key of the user), and finally sent to GTLS.

GTLS upon receipt of the message decrypts it with the use of its own private key. Next user data are found in the cache. Using the user's public key that has been

found in the cache a digital signature is verified. In case of a positive verification of the digital signature, user privileges for tools search are checked. If the user has appropriate privileges, the tool searching service is invoked with the user's searching criteria. Information on tools being found is sent back to the user.

4.3. Tool invocation

Once the tool is found, the client receives the parameters list from GTLS with which the tool can be invoked. Further on, the user defines the values of selected parameters. All data for tool invocation are comprised in the XML file that is encrypted with the use of the public key of the appropriate TS and signed by the private key of the user.

4.4. Querying GTLS for user data

The Tool Server queries GTLS for data on the user. GTLS searches its own cache, and once the session is found it sends back to the TS complete data on the user (public key, privileges, ...) and data on the user session. TS upon receipt of these data places them in its local cache. Due to this, with the subsequent client invocation to the TS during the same session, TS does not need to connect repeatedly to GTLS, but instead, it fetches session/user data from its own local cache.

As soon as, session and user data are found, the digital signature of the user is verified. Using the definition of privileges that is received from GTLS, it is verified whether the user has rights to invoke a particular tool. If yes, the tool is invoked and launched. A result of tool operation is converted into an XML file (parser), and sent back to the client.

5. SECURITY ISSUES

Specificity of work in a distributed collaborative environment and requirements formulated by the E-Colleg industrial partners were the basis of the proposed solution in the security domain. Security of communication, controlled access to common resources, as well as capability of security management are essential implemented elements in the developed TRMS (Magiera, 2003). Utilization of encryption methods allows assurance of data integrity and confidentiality at the time of transport through the network. Encryption is done with the use of a public key of a recipient that is the addressee of the message, and using the symmetric key that is generated before each message is sent. This combined encryption is due to a lengthy ciphering process with the use of the public key. This process may introduce delays. Encryption with the use of a symmetric key doesn't have these restrictions. Additionally, each message contains a digital signature, which is done on a basis of the private key of the sender of the message. This signature is verified on the addressee side and allows the confirmation of identity of the message sender. The process of identity confirmation is called authentication. Additional specific processes are needed in order to assure a right level of security. These are authorization and access control that are processes, which check and confirm rights of an actor that takes part in an action. Access control to TRMS resources proceeds

through verification of rights. The TRMS comprises a flexible system of privileges that allows adjustment of access rights to requirements appropriately. Security features implemented in the TRMS system are complemented by a mechanism which forces a definite behavior of users and appropriate control of access services. It allows a user to set the security level according to current necessities.

6. APPLICATION EXAMPLE

This short section illustrates potential applications of TRMS. The objective of a simple experiment is to convert a source image file and its transfer using e-mail. In the TRMS environment this task can be realized using the sequential workflow.

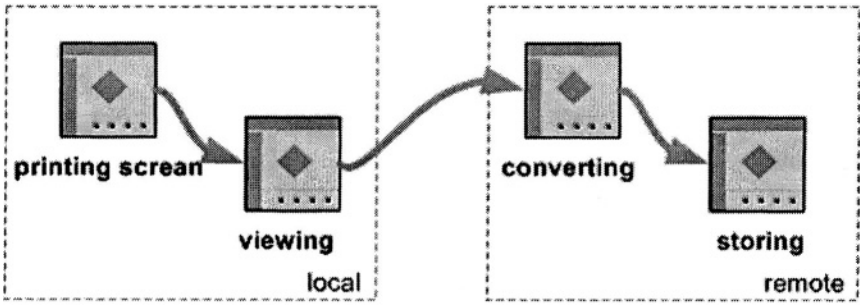

Figure 2 Simple sequential workflow.

The procedure requires that all tools used in the workflow are registered in the tools DB of GTLS. During this operation a user needs to define required parameters of each of involved tools, e.g. a path to the tool, required tool parameters, Tool Server on which the tool is installed. The following tools have been installed for this experiment: Gadwin PrintScreen(1), IrfanView(2), bmp2jpg converter(3) and the archiving script (4). First two tools were installed on the client local machine, whereas other on a remote server. Upon registration of tools an appropriate workflow was created that might be invoked by any TRMS user with adequate privileges.

Upon workflow initiation, a user dumps an appropriate part of the screen to the BMP file using (1). He can view the file with the use of the browser (2). In the following, the file is transferred to the remote TS, where with the use of (3) it is converted to the appropriate format (JPG). After conversion the graphic file is transferred using (4) to the archive.

This scenario illustrates easiness of invocation of remote tools in TRMS. Sequential tasks may be thus automated and arbitrarily repeated using this WfM tool.

7. CONCLUSIONS

The presented TRMS architecture and its functionality fulfill requirements for a secure collaborative infrastructure. The TRMS infrastructure supports tool integration and management. In our opinion, TRMS is a solution for distributed cooperating engineers and gives them the possibility for a simple invocation of distant tools. Distributed engineering work will become much easier if engineers are able to cooperate in larger networks with many registered tools. Their work will be accelerated and thus work costs will be reduced. A registered tool will provide a service to distributed design engineers. It is expected that the overload due to tool installation and configuration processes will be reduced. Additionally, encrypted and digitally signed messages assure a high level of security in sensitive design information exchange. TRMS constitutes infrastructure well tailored to distance-spanning engineering collaboration in complex distributed product design.

Further R&D related to TRMS includes enhanced WfMS and improved support for engineering teamwork with both synchronous and asynchronous collaboration.

7.1 Acknowledgments

E-Colleg TRMS was a collaborative R&D effort of the E-Colleg consortium that included: W. Mueller, T. Schattkowsky, H.-J. Eikerling, and S. Bublitz from C-Lab (Univ. Paderborn and Siemens Business Services). The team from C-Lab elaborated the first TRMS definition. M. Bauer (Infineon Technologies), M. Carballeda (Thales Optronique), as well as K. Siekierska and A. Kokoszka (Institute of Electron Technology, Warsaw) are kindly acknowledged for TRMS deployments and feedback on experiments.

8. REFERENCES

1. Chan F., Spiller M., Newton R.: WELD- An Environment for Web-based Electronic Design, Proc. DAC 1998, San Francisco.
2. Lavana H., Brglez F., Reese Konduri G.: OmniDesk/OmniFlow: An Agent-Based Architecture and a Toolkit for Building Configurable and Collaborative Workflows of Heterogenous Applications and Data for the Web. CBL Report 1998-TR@CBL-10-Lavana.
3. Lavana H.:A Universally Configurable Architecture for Taskflow-Oriented Design of a Distributed Collaborative Computing Environment, Thesis PhD, 2000
4. ASTAI(R) home page, http://www.c-lab.de/astair
5. Schneider A., Ivask E.: Internet-based collaborative system design using Moscito, Workshop on Challenges in Collaborative Engineering (CCE'03), 15th-16th, April 2003, **Poznań,** Poland.
6. Bauer M., Eikerling H.J., Mueller W., Pawlak A., Siekierska K., Sodeberg X., Warzee X.: Advanced Infrastructure for Pan-European Collaborative Engineering. In: Stanford-Smith B; Chiozza E: E-work and E-commerce, Novel solutions and practices for the global networked economy. IOS Press / Ohmsha, Berlin, 2001
7. Magiera J., Kostienko T., **Fraś P., Szlęzak M.**: Security issues in Tool Registration and Management System, Proc. Workshop on Challenges in Collaborative Engineering (CCE'03), 15th-16th, April 2003, **Poznań,** Poland.
8. Mueller W., Schattkowsky T., Eikerling H.J., Wegner J.: Dynamic Tool Integration in Heterogeneous Computer Networks, Proc. Design Automation and Test in Europe DATE 2003, Munich, Germany, 3-7 March 2003
9. Indrusiak L.: A Review on the Framework Technology Supporting Collaborative Design of Integrated Systems, Exame de Qualificação, Porto Alegre, 2002