An Advanced Infrastructure for Collaborative Engineering in Electronic Design Automation

T. Kostienko[1], W. Mueller[2], A. Pawlak[1], T. Schattkowsky[2]

[1]*ITE/SUT, Collaborative Engineering Lab, Poland*

[2]*C-LAB/Paderborn University, Germany*

ABSTRACT: Engineering collaboration gets its new global dimension with the omnipotent access to Internet. Engineers have severe requirements concerning: security of design data, quality of net connections, easiness of collaboration, etc. This article presents visions and middleware architecture to establish pan-European collaborative engineering infrastructure and its application in the field of Electronic Design Automation (EDA). We present a transparent infrastructure to engineers to enable their Internet-based collaboration during the design of complex electronic systems. In this context, we introduce an advanced collaborative infrastructure (ACI) for distance spanning, tool integration, and administration as well as open interfaces for XML-based data exchange. ACI constitutes a backbone for our collaborative research and engineering studies by fostering a combination of most recent plug-and-play technologies and secure, peer-to-peer data transfer with XML-based tool integration. ACI and its deployments have been developed with the EU project E-Colleg (IST-1999-11746).

Keywords: Collaborative Engineering, Distributed Workgroups, Tool Integration, Electronic Design

1 INTRODUCTION

The advancements in information and communication technologies currently reshape our work. Although new *electronic Work (eWork)* technologies penetrate different professional activities, it is in engineering, where they have a potential for the highest added value. We claim that engineering collaboration is especially sensitive to the new technologies though it is not sufficiently recognised by current practice yet. The shifting from the traditional manufacturing paradigm to a new, virtual and agile model is globally observed. Whereas the traditional model is characterised by the very limited information sharing, static organisational structure, and almost no cooperation, the virtual and agile model exhibits information sharing, collaboration, and dynamic organisation. Collaborative engineering is an innovative paradigm for product development, which integrates widely, distributed engineers for virtual collaboration [6].

Within general globalisation efforts, not only organizations and management but also engineering teams become distributed over large distances. The worldwide market of Integrated Collaborative Environments (ICE) is highly increasing [5]. The revenues for ICE markets are predicted to exceed 1 billion dollars very soon. Intranet, document management, workflow solutions, as well and group calendaring and scheduling are currently the main drivers of this market.

Collaborative engineering constitutes a new paradigm of engineering work that is central to the vision of the engineering working environment in the Information Society. Internet as a backbone of the collaborative infrastructure is per se trans-national in nature. From the technical point, existing collaborative engineering frameworks do presently not support very well the integration of very complex engineering environments when interconnecting multiple design groups and do not consider distance-spanning related issues like firewalls, security, remote administration, and distributed design flow automation. Distributed engineering development needs however new infrastructures, net-aware tools, and new design methodologies based on re-use in combination with advanced security and network and tool management [16]. If supported by appropriate advanced collaborative infrastructures and tools, it radically speeds up collaboration over wide distances by connecting Intranets via the Internet. New collaborative technologies allow for fast and flexible building of virtual distributed teams and efficiently makes use of distributed engineering resources.

This article presents the E-COLLEG [7] Advanced Collaborative Infrastructure (ACI) with integrated dynamic Tool Registration and Management Services (ACI), open XML-based data exchange, and collaborative tool extensions. ACI is tailored to distance spanning engineering collaboration and security enabled data exchange. The infrastructure is validated through two industrial Intranet-crossing case studies between Polish and French, as well as, between Polish and German industries applying real-world development scenarios [1]. In both application scenarios, common engineering practices have been identified and built into engineering workflows. Further, both truly pan-European collaborative industrial scenarios will provide credible evidence of the innovation of E-Colleg collaborative technology that will contribute to the new emerging paradigm of eWork with outsourcing involving SMEs.

The remainder of this article is structured as follows. Section 2 presents existing approaches, whereas Section 3 introduces the architecture and the functionality of the advanced collaborative infrastructure. Finally, Section 4 discusses shortly the undertaken applications.

2 EXISTING APPROACHES

In complex engineering projects, engineering skills are typically dispersed among various departments located in different cities and countries. Multiple CSCW (Computer Supported Collaborative Work) tools are frequently applied to manage existing parts of problems: Intranet solutions (Netscape, MS IE etc.), document management solutions (Open Text, Interleaf, Documentum etc.), workflow solutions (StaffWare, Keyflow Filenet, Banyan etc.) and group calendaring, scheduling, and shared workspaces (Lotus/IBM Notes, Microsoft Exchange/Outlook, BSCW etc.).

In Electronic Design Automation (EDA), current industrial practice is that, modelling and synthesis tools, as well as specialised libraries of components reside on different servers. Thus, engineers are enforced to move design data between tools using *email*, *ftp*, or *ssh*. In current practice, engineers and administrators are often enforced to download new tools and to configure them in their design environments. Due to limited licenses, remote access and remote management is almost impossible and extremely hard to manage over large distances. All these current and traditional approaches are error prone since installing new tools, and/or moving manually data among tools is usually not straightforward, and definitely not scalable for large, multi-site, collaborative engineering tasks. However, those tools mainly offer partial solution and do not combine very well.

Some support and standards come from the domain of workflow automation and tool integration: the Tool Encapsulation Specification (TES) and the Web Services Description Language (WSDL).

2.1 *Tool Encapsulation Specification (TES)*

CFI (CAD Framework Initiative) started work on tool integration in the context of the Inter-tool Communication (ITC) subcommittee in 1989. In ITC, the Tool Encapsulation Specification (TES) language was developed [3]. TES provides an integration language, which defines input/output behaviours of integrated tools.

A TES definition includes a tool identifier and a short textual tool description, a list of platforms able to run the tool, a list of names and types of tool parameters, the classification of tool parameters as optional and mandatory, and preprocessor instructions for formatting parameters, for instance, for the concatenation of parameters for invocation of a batch tool.

TES was defined as a LISP-based language in order to ease implementation, since LISP supports platform portability and dynamic binding. The focus of TES is on parameter processing and invocation. Integration of complete tool suites, e.g., combining MS Project with CAD tools with integrated UI is not explicitly supported. Hence, additional commodity tools like Windows Scripting Host have to be integrated. Additional services can be included similarly like databases or web servers. TES does not support semantic tool description and lacks compatibility with current web standards. Only very few systems are known which have implemented ITC concepts and TES. One example is the tool management system ASTAI(R) [4][18] that integrates TES and partly WSDL.

2.2 *Web Services Description Language (WSDL)*

We currently face a wide acceptance of approaches based on the eXtensible Markup Language (XML). The W3C defines a set of XML-based languages that are the foundation for the current notion of web services. The Web Service Description Language (WSDL) is used to describe interfaces of a web service. These interfaces can be accessed using the Simple Object Access Protocol (SOAP).

Consider the example of a web-based simulation service, which can be easily defined for an application server using WSDL for tool encapsulation. If required, the simulation service may be easily enhanced by a complex workflow, which integrates various additional resources, e.g., a web server. Similar to TES, the WSDL description can be used to define name, input, output, and data types via which a client communicates with that service. WSDL descriptions can be made available via a central UDDI registration (Universal Description, Discovery, and Integration), e.g., in order to implement resource discovery.

3 ADVANCED COLLABORATIVE INFRASTRUCTURE (ACI)

To provide an efficient and adequate infrastructure for collaborative engineering, we have defined ACI in the context of the E-COLLEG project [7]. Though it is currently applied for the distributed design of electronic systems, it is not limited to those. ACI implements basic engineering, tool management, and data transfer support. Thus, it could be applied in several other scenarios as well.

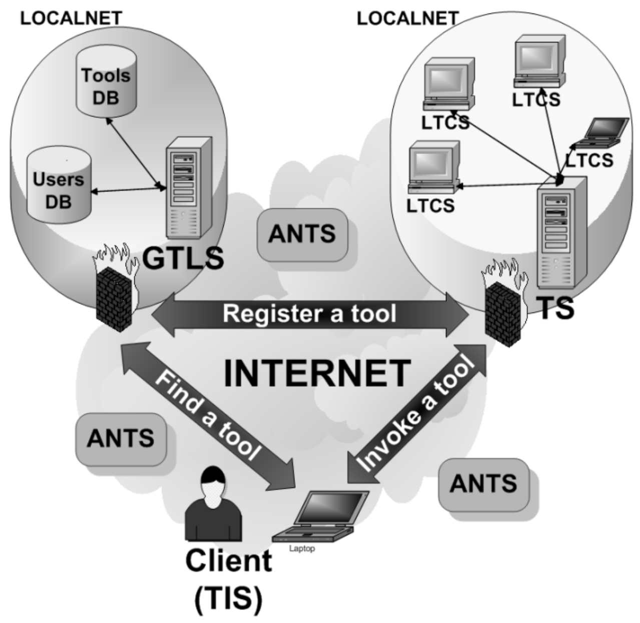

Figure 1: ACI System Architecture

ACI is based on three core services and a complimentary transport service, which enables secure firewall crossing communication between the components (see Figure 1). The Tool Invocation Service (TIS) is the interface used by clients to activate remote tools. TIS invokes the Global Tool Lookup Service (GTLS) to discover the appropriate tool for the requested task. The activation of the remote tool is done by the Local Tool Control Service (LTCS) via the Tool Server (TS). Together with the Advanced Network Transport Services (ANTS), the instances of these services form the ACI infrastructure.

The ACI core components GTLS, LTCS and TIS as well as ANTS are implemented as Web Services that are interconnected using the Simple Object Access Protocol (SOAP). The SOAP messages between the components are transported using ANTS.

In a general case, all ACI components are on separate machines connected to the Internet. Thus, they need to communicate through an insecure media and all data are encrypted (ciphered) and digitally signed by a sender.

In the reminder of this section, we first introduce the individual ACI components. Thereafter, we outline their interaction.

3.1 *Advanced Network Transport Services (ANTS)*

ANTS provides an abstract data transport mechanism through reliable Request/Response functionality that are mapped to actual transport mechanisms, i.e., SOAP/HTPP or just TCP. It introduces an abstract addressing schema that allows having unique addresses for each network node. The inbuilt routing mechanism supports routing messages though multiple physical networks including offline and relay networks as given in Figure 2. The possibility to map to almost any ISO/OSI layer 4+-transport mechanism overcomes several firewall problems. Using SOAP/ANTS transport gives full transparent location independent connectivity between all ACI components.

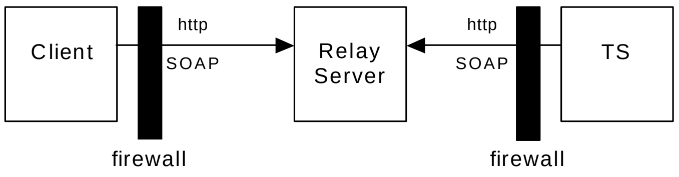

Figure 2: Firewall-crossing interaction of ACI components using ANTS

ANTS is based on a generic transport mechanism, which includes a simple Web Service that serves as a replacement for FTP denoted as the Simple Structured Storage Service (S3S). It is used within ACI to store the input and output data sets used during tool invocation. The main design goal for this service was indeed to provide a replacement fro FTP, which can be easier controlled and monitored.

S3S has been developed as an extremely easy to implement data storage service that offers several benefits over using FTP. These benefits include at first hand the accessibility over SOAP, which implies the possibility of crossing firewalls using ANTS or just using any other transport to access the service. Furthermore, ambiguities and limitations that occur when dealing with real world variants of the FTP protocol disappear. The S3S WSDL interface allows easy direct integration of S3S into object-oriented software and a uniform modelling of such systems.

The S3S data model is kept very simple (see Figure 3) to allow easy implementation. However, the support for attributes with multiple values allows introducing additional semantics that do not need to be explicitly implemented. Such semantics may be user defined, i.e., ACI stores input/output data sets along with their identifiers. Some semantics can be predefined to extend service capabilities. The possibility to define of multidimensional hierarchies and containment scenarios using attribute semantics is obvious. This also eliminates the need for explicit folder hierarchies as in FTP and thus simplifies the implementation of the service.

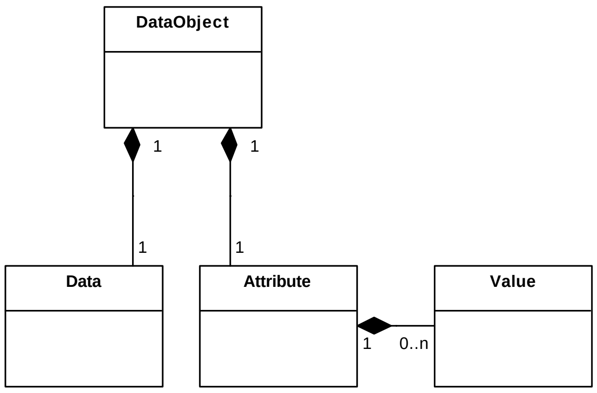

Figure 3: S3S Data Model

Data search and retrieval from the service takes place by queries that may contain conjunctions and disjunctions over data object attributes. The processing of such queries is very simple to implement and allows, as already outlined, the elimination of several FTP operations.

3.2 *GTLS (Global Tool Lookup Service)*

GTLS is a key component of the ACI environment. It manages two repositories with tools and users. Each tool has to be registered in GTLS that includes the information on machines that make it available. Remote clients can search for registered tools. GTLS is responsible for registration and modification of data on tools, users, and their privileges. GTLS is responsible for the security policy of the whole system by: registration of users and their activities; assignment of multilevel, complex access rights; registration of access to tools, maintaining statistics, identification of an intruder attack. Further, GTLS is responsible for generation of keys used for encryption of messages and generation of digital signatures and management of a repository of public keys of registered users that are used for authentication of messages (digital signature).

3.3 *Local Tool Control Service (LTCS)*

The LTCS constitutes a local call to the tool. The LTCS is a simple component which task is to receive SOAP request messages and their conversion into appropriate command lines that launches the tool. The conversion is executed with respect to the received tool request, design data, and to a locally available tool encapsulation specification which defined tool specific details like command line parameters, input/output formats, tool host etc. For tool encapsulation, we have defined an E-COLLEG specific XML schema.
The LTCS is placed in a local network and can only be invoked through the Tool Server (TS). This is mainly because LTCS is not equipped with a mechanism for verification of authorisation and there is neither user authorisation, nor encryption/decryption in LTCS.

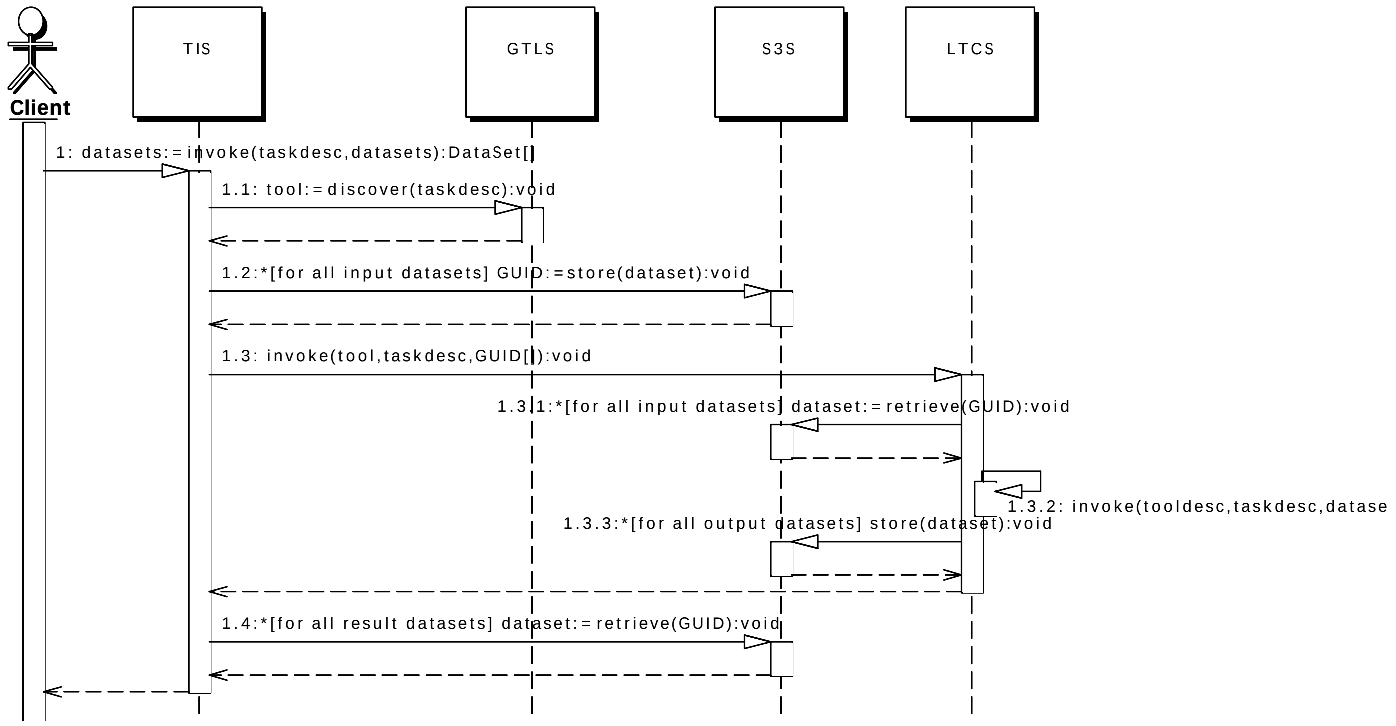
Client
TIS
GTLS
S3S
LTCS
1: datasets:=invoke(taskdesc,datasets):DataSet[]
1.1: tool:=discover(taskdesc):void
1.2:*[for all input datasets] GUID:=store(dataset):void
1.3: invoke(tool,taskdesc,GUID[]):void
1.3.1:*[for all input datasets] dataset:=retrieve(GUID):void
1.3.2: invoke(tooldesc,taskdesc,datase
1.3.3:*[for all output datasets] store(dataset):void
1.4:*[for all result datasets] dataset:=retrieve(GUID):void

Figure 4: TIS Behavior

Thus, TS decouples the client and the LTCS. The client encrypts a message before sending it. The TS decrypts the received message, checks the signature and the user privileges, and if correct, it invokes the appropriate LTCS. Its additional task constitutes brokerage in request/user authentication and queries the GTLS whether a request has sufficient privileges (*basic authorisation*).

The TS also provides mechanisms for optional, *additional authorisation* that is conducted using data in the local DB of the TS. This database is optional and may comprise additional restrictions on granting access to particularly sensitive tools. An organisation being the owner of those sensitive tools, and that manages the particular TS, may not "fully trust" GTLS. It may thus impose its own more restrictive verification of access rights.

3.4 *Tool Invocation Service (TIS)*

The TIS is used by clients to transparently invoke remote tools that will be discovered using GTLS and invoked by accessing the corresponding LTCS through the responsible TS that ensures the security policy (see Figure 4).

Furthermore, the Client may have access to every component of the ACI environment through a GUI that gives access to all possible ACI operations (search for tools, tools registration, users registration). A particular operation may be realised from a GUI level depending on the user privileges that are stored in the central DB in GTLS.

3.5 *ACI Component Interaction*

We finally outline an ACI session, which gives an example how the different components may interact. Figure 5 depicts operation steps (**1-4**) that a user has to follow in order to invoke a selected tool.

1. A user needs to initiate his/her own ACI session. He/she proceeds with a login at GTLS. The user gives his parameters, i.e., login and a password that are sent by the user client to the GTLS in a form of an XML (SOAP) message. Parameters being sent are encrypted by a combination of a public key and a temporary symmetric key. The message is signed by the user's private key in order to assure an additional authentication.

GTLS upon receipt of the message deciphers it with a use of its own, secret private key. The deciphered message contains login parameters. If these data are correct, the user's public key is fetched from the user database (DB) and the digital signature is verified. If correct, a session is created for the user.

Complete user data (public key, privileges) are placed in *cache* of GTLS, which also holds session data (login time, operations done, etc). With the use of this key the digital signature of a SOAP message is verified in order to check authenticity of the user.

2. As the user is logged in, he/she may search for a required tool. He may provide 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 ciphered using the public GTLS key and the generated symmetric key, signed (digital signature is generated based on a secret private key of the user), and finally sent to GTLS.

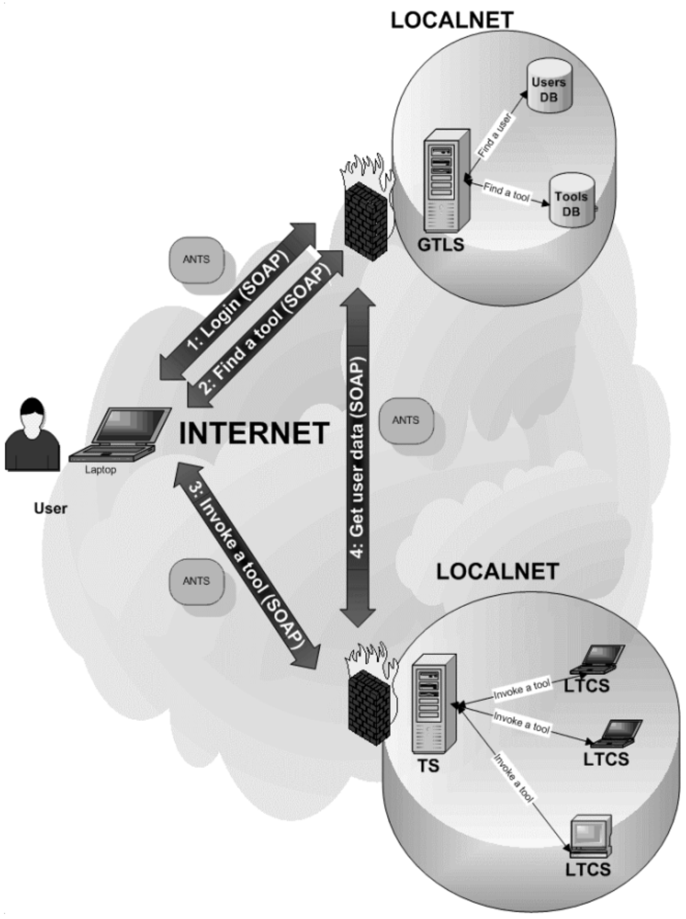

Figure 5: ACI operation.

GTLS upon receipt of the message deciphers it with the use of its own private key. Further, a session is fetched from cache that contains user data (public key, privileges, etc). A digital signature is verified using the user public key. If the user has appropriate privileges, the tool searching service is invoked with the searching parameters that were sent in the message. Information on tools being found is sent back to the user who has initiated this search.

The return message is ciphered (using the public key of the client), as well as signed (with the use of the GTLS private key) and sent back from the GTLS to the Client. Once the tool is discovered, the client with the use of data received from GTLS prepares parameters for locating the tool. These parameters are placed in the XML file that is ciphered with the use of the public key of the appropriate TS and with the use of generated symmetric key, as well as using the private key of the client.

3. Tool data also comprise a reference to the TS that gives access to the tool. Using these references the Client makes connection to the given TS and invokes the tool. The TS upon receipt of the XML message deciphers it with the use of its own secret private key. Further on, based on the user session number, the TS searches its own local *cache* with sessions being stored. If a session with indicated ID is not discovered, it connects to GTLS to continue the search. Messages are of course ciphered and a digital signature is sent.

4. GTLS searches its *cache*. Once the session is found it sends the complete data on the user (public key, privileges,..) and session back to the TS. Upon receipt of the data on the session and on the user, the TS places them in its local *cache*. Due to that, TS must not continuously connect to the GTLS, but instead it fetches session/user data from its own local *cache*.

As soon as, the session and user data are found, the digital signature of the user is verified. Using a definition of the privileges that are received from GTLS, it is verified whether the user has rights to invoke a particular tool. After this verification, an additional check of privileges is effectuated on the TS, which verifies, if additional, local restrictions are defined for this particular tool.

This is done using the local DB that is located on the machine of the TS. This is an optional DB, where one can define additional restrictions independently from the privileges DB, e.g., on the rights to invoke specific well-protected tools, which are limited to a small number of users. If additional restrictions are identified, it is checked, whether the user has sufficient rights. If the user passes both verifications he/she receives access the tool. In such a case, TS verifies, if the invoked tool is placed on the same local host as the TS. If this is the case, the tool is invoked and launched. If not, TS manages the reference to the LTCS, where the tool is located. The result of the tool operation is converted into an encrypted XML file, signed and sent back to the client.

4 APPLICATIONS

The ACI services were implemented, and evaluated in several considerably complex industrial application scenarios in the currently running ECOLLEG project [7] covering:

1. Printed Circuit Board Design
2. IP-Based Circuit Design

4.1 *Printed Circuit Board Design*

In a first scenario, we investigated a distributed environment for Printed Circuit Boards (PCB) design in cooperation with Zuken GmbH, Germany. The experiment was restricted to Intranet network communication and applied SNMP (Simple Network Management Protocol) for communication and control [12].

In that context, we have implemented a typical workflow for a PCB design process utilizing tools from the Zuken Hot-Stage tool suite: CADSTAR-SI, EMC-Engineer, and the integrated simulators FREACS, Sigma, and ComoRan. The presented workflow depicts the processing steps from schematic entry over layout generation to final verification and simulation of analog properties

The complexity in that application did not lie in the number of different tools and their distribution over different servers rather than in the large number of versions and versions of their supported input and output formats considering the complete tool life cycle. Thus, the focus of the work was on the discovery of tools over the life cycle rather than the workflow definition and organization.

4.2 *IP-Based Circuit Design*

The second category of application scenarios considered two applications of distributed IP-Based modeling and verification of VHDL models. Both applications are designed and verified by a distributed team of engineers in the framework of a multisite infrastructure coupling the design flows of a design team from major industries in conjunction with an SME. The first scenario is mainly focused on tool aspects, i.e., integration of heterogeneous tools. The second one mainly addresses design data exchange aspects. Both are evaluated in the infrastructure of two different major industries so that sufficient conclusions can be drawn for wider applications.

In the first application, the Institute of Electron Technology (ITE) used an IP-based methodology for electronic system design. ITE, Warsaw, played a role of an IP provider to Thales Optronique, Paris. Each IP component was modelled at VHDL Register-Transfer Level. The model was developed as a reusable IP - parameterised and thus configurable. The component model was earlier verified by simulation. Several VHDL-based tools could be applied at that design stage. The verified model was synthesised in a selected FPGA technology. As the result of logic synthesis, a netlist of hardware components was obtained. During the next step the implementation of this netlist into a selected type of FPGA device had taken place. Appropriate workflows have been defined and verified [10].

The second application scenario integrates Infineon Technologies, Munich, and Silesian Univ. of Technology, Gliwice. It constitutes an experiment in collaborative testbench development. The experiment that was undertaken [2] aimed at development of a testbench for functional verification of a Serial ATA hard disk controller VHDL model using the distributed workflow functionality of ASTAI® [4].

The environment that has been developed, integrates a set of distributed workflows that cross borders of involved organisations. Infineon Technologies has been motivated towards this work by the extremely high effort for making functional tests. Independently, Infineon was working on standardized IP-based design flows and integrating SMEs into Infineon's internal design flow. The application of the collaborative technology based on the E-Colleg collaborative infrastructure opens new possibilities for Infineon’s testbench development tasks, because, among others, the existing environment can be used on each partner site, and be coupled using collaborative workflows.

5 CONCLUSIONS

We have presented the architecture of an advanced collaborative infrastructure (ACI) for Electronic Design Automation application. ACI has been developed for distance spanning, tool integration, and administration as well as open interfaces for XML-based data exchange. ACI provides a combination of most recent plug-and-play technologies with peer-to-peer communication and XML-based tool integration. We have applied ACI to tool integration and mainly addressed tool administration in complex Intranet and Intranet crossing communication and data exchange. For such site-spanning tool integration, one has to consider security as an issue of highest priority in order to protect the exchanged design data. Advanced mechanisms for security assurance, namely: encryption with the use of both symmetric and asymmetric keys, digital signature, user authentication and authorisation, have been built into the ACI architecture. As one of the industrial requirements, ACI provides bridging firewalls via ANTS’ peer-to-peer based technologies.

Current case studies and trials in well-defined testbeds together with EDA tool vendors, system houses, and chip manufacturers show promising results and demonstrate the applicability of the presented ACI concepts.

ACKNOWLEDGEMENTS

The work described herein is funded by the IST project E-Colleg (IST-1999-11746). We gratefully acknowledge the fruitful discussions and valuable remarks of our E-Colleg partners. Further on, we thank Manuel Carballeda (Thales Optronique) and Mathias Bauer (Infineon Technologies) for their cooperation and the GTLS development team members from SUT (Silesian Univ. of Technology), namely: Pawel Fras, Jaroslaw Magiera, and Marek Szlezak for their contributions.

REFERENCES

[1] M. Bauer, H.J. Eikerling, W. Mueller, A. Pawlak, K. Siekierska, D. Soderberg, X. Warzee: Advanced Infrastructure for Pan-European Collaborative Engineering. E-work and E-commerce, B. Stanford-Smith & E. Chiozza (Eds.), IOS Press, 2001.

[2] M. Bauer, P. Penkala, A. Pawlak, D. Stachañczyk, P. Fras,: Collaborative Environment for Testbench Development. EUROMICRO DSD2002 (Digital System Design) Symp., Work in Progress Session, Dortmund, Sept. 2002.

[3] CFI: Tool Encapsulation Specification; Version 1.0.0. CAD Framework Initiative Inc., Austin, USA, 1992.

[4] C-LAB: ASTAI® Manual v. 2.2, Paderborn, 2000, www.c-lab.de

[5] Collaboration : Technology and Market Dynamics, IDC report for DARPA, June 1997.

[6] Cutkosky, M., Tenenbaum J., Gliksman J.: Madefast: An Exercise in Collaborative Engineering over the Internet, ACM Communications, Sept. 1996, vol. 39, no. 9.

[7] E-COLLEG (2000-2003), EU project IST-1999-11746, www.ecolleg.org.

[8] EXTERNAL (2000-2002), EU project IST-1999-10091. www.external-ist.org.

[9] A. Kokoszka, K. Siekierska, A. Trung, P. Fras, A. Pawlak: Are workflow management systems useful for collaborative engineering? ECPPM2002, Portorož, Slovenia, Sept. 9-11, 2002, A.Balkema Publishers, pp. 245-252.

[10] A. Kokoszka., Q.T Nguyen., K. Siekierska, A. Pawlak, D. Obrebski, N. Lugowski: Distributed Design of Semiconductor IP Based on the Workflow Concept, Proc. of the IEEE Design and Diagnostics of Electronic Circuits and System Workshop, Gyor, 18-20 April, 2001.

[11] H. Lavana et al.:. OpenDesign: An Open User-Configurable Project Environment for Collaborative Design and Execution on the Internet. ICCD 2000.

[12] W. Mueller, T. Schattkowsky, H.J. Eikerling J. Wegner, Dynamic Tool Integration in Heterogeneous Computer Networks. Proceedings of DATE 03, Munich, Germany, 2003.

[13] Q.T. Nguyen, A. Kokoszka, K. Siekierska, A. Pawlak, D. Obrêbski, N. Lugowski: Organization of a Microprocessor Design Process Using Internet-based Interoperable Workflows. 3rd Int. IEEE Symp. on Quality Electronic Design, 18-21 March 2002, San Jose, CA.

[14] A. Pawlak A., W. Cellary, A. Smirnov, X. Warzee, J. Willis: Collaborative Engineering based on Web - how far to go?, Advances in Information Technologies: The Business Challenge. J.-Y. Roger et al. (Eds.) IOS Press, 1997.

[15] J. Postel, J. Reynolds: File Transfer Protocol (FTP). RFC 959. 19985.

[16] F.J. Rammig: Web-based System Design with Components Off The Shelf (COTS). Forum on Design Languages, Tübingen, Sept. 2000.

[17] D. Schefstroem, G. van den Broek.: Tool Integration. Wiley Series in Software Based Systems. John Wiley & Sons, 1993.

[18] W. Thronicke, W. Fox, et al.: From Tool Integration to Workflow Management - A Lean Integration Solution. In Proc. 2nd World Conference on Integrated Design and Process Technology, Austin, TX, Dec. 1996.

[19] W3C, http://www.w3.org, 2002.

[20] M. Witczynski, A. Pawlak: Virtual organisations enabling net-based engineering. eBusiness and eWork. E-2002 Conference Proceedings, October 2002, Prague.