Dynamic Tool Integration in Heterogeneous Computer Networks

Wolfgang Mueller, Tim Schattkowsky
Paderborn University
Paderborn, Germany

Heinz-Josef Eikerling
Siemens Business Services
Paderborn, Germany

Jan Wegner
Zuken - EMC Technology Center
Paderborn, Germany

Abstract

Tool installation and automation of administrative tasks in heterogeneous computer networks becomes of increasing importance with the availability of complex heterogeneous computer networks. This article introduces a new approach for dynamic network tool management, i.e., TRMS. A variant of TRMS using SNMP - a well established standard for network administration – is outlined and illustrated by the application of the integration and management of design tools for Printed Circuit Boards (PCBs).

1. Introduction

Tool management and integration [5] is of utmost importance when establishing complex collaborative engineering environments [2,8]. With steadily increasing complexity of heterogeneous computer networks, the administration and integration of the huge numbers of various resources is currently one of the biggest challenges for system administrators. Resources in this context include *physical devices and hardware*, like computers or peripherals as well as *software*, like office applications, design environments or groupware, and *human resources*, like programmers or secretaries.

For the provision and management of these resources in complex and heterogeneous computer networks, radically new concepts and solutions have to be developed. Key issues and means for the development of such concepts can be summarized as follows.

Modularity/Extensibility/Scalability: to provide a high degree of reusability, the integration has to support modular and hierarchical tool organizations.

Distribution and Portability: tool administration has to support the integration of distributed, decentralized resources running on various platforms.

Internet Integration: most of today's, partly public available resources, are connected and only reachable via Internet. Existing widespread Internet exchange and transmission formats and protocols have to be supported.

Standardization: to provide affordable and simple data exchange between integrated systems, description languages and formats have to be driven by a wide industrial consensus.

This paper presents a new approach for dynamic and secure resource integration and administration, i.e., TRMS (Tool Registration and Management Services). TRMS allows the dynamic discovery of a tool using semantics descriptions of the desired tool behaviour. A tool is described by a set of significant properties based on which it can be discovered in the network. We outline a variant of that approach based on SNMP (Simple Network Management Protocol), which is a widely accepted standard for general network administration. The implementation gives the context for an illustrating example in the area of a realistic PCB design flow based on the Zuken Hot-Stage tool suite.

The remainder of this paper is structured as follows. The next section introduces related works. Section 3 presents our TRMS concepts for dynamic tool registration and discovery. Section 4 outlines an implementation of these concepts using SNMP. Section 5 explains the application to a PCB design flow. The paper finally closes with conclusions and an outlook.

2. Related Works

Main related works come from the domain of workflow automation and tool integration. We briefly outline CFI's Tool Encapsulation Specification (TES) and then discuss the application of web services and their related XML-based technologies in industrial tool integration scenarios with the indication of current limitations.

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 [1]. 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 basically 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) [4,8] which 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 [9] 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 the 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. A WSDL definition includes:

- *type*: hierarchical data type definitions
- *message*: type definitions of the exchanged data
- *operation*: description of a supported operation
- *port type*: set of the supported operations
- *binding*: concrete protocol and data format of individual port types
- *port*: an individual 'end point' with binding and network address
- *service*: set of combined ports

WSDL descriptions can be made available via a central UDDI registration (Universal Description, Discovery, and Integration) [10], e.g., in order to implement resource discovery. Currently, there are some serious reservations using such web services for industrial applications. The related standards like SOAP are currently evolving fast with the side effect of introducing compatibility issues and general uncertainty. Since these technologies are not stable yet, it is undesirable to apply them today in a true industrial context.

3. Dynamic Tool Management

The classical approach to tool administration is that a tool is installed on a central server and made available in the network via NFS or SAMBA. New versions and patches typically replace older ones, which results in a permanently modified working environment to which the user has to continuously get customized to. Due to significant upward or downward incompatibilities, system administrators as well as users are facing real challenges from time to time. Those challenges are even more significant in large Intranets where tools can be seen as a highly dynamic resource. Its availability has to be often managed over extremely long life cycles. In contrast to the installation and administration of hardware, the installation and administration of software is even more complex since software is made available in different versions under multiple operating systems and comes in various configurations, customisations, and service packs.

We present how JINI concepts (leasing, discovery, lookup) [7] can be meaningfully applied to the management of design tools. Through these concepts, tools can be discovered in the network by either their names and/or their properties. Here, not only one tool may match the discovery. When more matches are received, the optimal match with respect to static and runtime criteria is computed, e.g., a simulator on a machine with smallest CPU load.

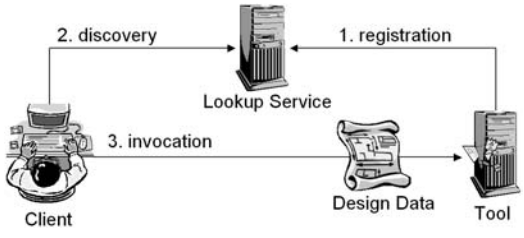

Figure 1: Dynamic Tool Management

Corresponding to JINI principles, we can identify three phases (see Figure 1) for dynamic tool management and administration: registration, discovery, and invocation.

A *tool* has to be registered with a lookup service to be available within the network. This lookup service is used by a *client* to discover a tool that is able to fulfil the client's needs. Once the client has discovered the tool, it is directly invoked to serve the client. Based on these concepts, we introduce a set of *Tool Registration and Management Services (TRMS)* that support dedicated core functions as given in Figure 2.

The *Tool Invocation Service (TIS)* is a service local to each client that is called to transparently invoke dynamically discovered remote tools. By wrapping TIS functionality, i.e., as a command line tool, transparent dynamic tool invocation can be easily integrated into stand-alone clients or workflow management tools.

The *Global Tool Lookup Service (GTLS)* is a single global tool registry that is responsible for assigning tools to incoming discovery requests from TIS instances.

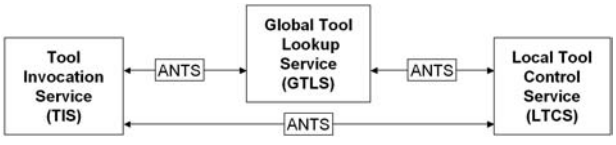

Figure 2: TRMS Core Services

Once a TIS has successfully discovered the required tool for a specific task it directly invokes the tool by calling a local resource. This is managed by the *Local Tool Control Service (LTCS)*. The LTCS has to be deployed locally on the computers hosting the tools and is responsible for activating the tool for remote clients and passing parameters and output data between the tool and the calling TIS. Furthermore, the LTCS periodically updates the GTLS based on specified conditions and availability of tools. It is also responsible for registering the local tools at the GTLS. Additionally, the LTCS supplies the GTLS with current runtime properties of the tool and the host, e.g., the current CPU load or memory allocation.

To enable dynamic tool management in heterogeneous networks, a high level transport mechanism that transparently transports service activations and the corresponding data, i.e., tool input and output data, is needed. The *Advanced Network Transport Services (ANTS)* are a set of services that provides secure end-to-end transport of data using existing network infrastructure, i.e., the Internet or a local area network (LAN). ANTS transparently choose the actual transport mechanism used. ANTS is used by TRMS to perform secure encrypted transmission wherever needed and to overcome networking problems, e.g., to tunnel through firewalls

The general TRMS approach does not depend on a particular implementation platform. Individual services can be implemented as a component framework as well as using SNMP as we outline in the next section.

4. SNMP-Based Management

This section outlines an SNMP-based variant of the previously introduced TRMS concepts. With SNMP we mainly address Intranet solutions. However, the presented concepts can also be easily applied for other environments, i.e., as a SOAP and UDDI based implementation.

4.1 Background

The Simple Network Management Protocol (SNMP) [6] was introduced as an IETF standard for the management of telecommunication networks in 1989. Today, SNMP is supported by vast majority of hardware and software manufacturers. SNMP provides the monitoring and control of arbitrary hardware and software components in a network by means of message exchange. For that, SNMP provides means for network integration through the data specification language ASN.1 and for message exchange.

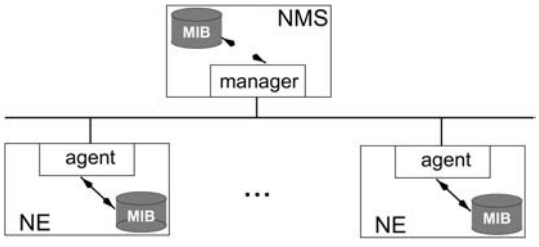

Figure 3: NMS and NE in a Network

Based on the UDP protocol, communication is established between at least one network management station (NMS) and one or more network resources, so-called network entities (NEs) as given in Figure 3. Entities can be simple devices, like a hub or a repeater, or complex hardware, like switches, gateways, or workstations. In SNMP, a manager of a management station communicates with the agent of an entity through the exchange of data from the local management information base (MIB)

SNMP messages are composed of two parts. The first part gives the SNMP version and a unique identifier, the so-called community or party. The second part contains the user data (PDU - Protocol Data Unit) composed of an operation and an object instance. SNMP defines five different base message types:

- GET: NMS orders an object instance from an NE
- GET-NEXT: NMS orders the next object instance from the list or table of NEs

- GET-RESPONSE: NE sends an object instance in response to a GET or GET-NEXT message
- SET: NMS sets a value of an NE attribute
- TRAP: is sent by an NE as a synchronous message to an NMS

The structure of an object instance is defined by means of the data specification language ASN.1 (Abstract Syntax Notation 1), which is typically provided by the manufacturer of the device. For interpretation of the object instance, the object definition has to be available in the MIB (Management Information Base) of the NMS and the MIB of the NE. For SNMP, only a subset of the ASN.1 is used, which only includes simple base types like Boolean, Integer, Real, and simple composed types such as sequence and set. An ASN.1 definition is composed of the definition of an object identifier, attributes (data types, annotations, and status information), and allowable operations. The object identifier uniquely identifies the object in the MIB and the entire network. The identifier is hierarchically defined along a tree structure. It is given as a combination of strings or integers. The object identifier "iso.org.dod.internet.mgmt", for instance, has "1.3.6.1.2" as integer representation.

The exchange of messages is defines along simple rules. An NMS usually periodically polls the NEs and ask them for values of previously defined types. NEs simply send these values without any further actions. Only in the case of an exception, the NE actively sends a TRAP message. The NMS decides itself if and how it should reply to that message. A typical communication pattern is, that a NMS (e.g., a PC) asks an NE (e.g., a printer) for the number of printed pages. For this, the NMS sends a GET message to the printer and asks for the value of the page counter. When the SNMP agent of the printer receives the request, it replies with a GET-RESPONSE message containing the counter value. If the NMS wants to change that value, it sends a new value using a SET message.

Based on SNMP, we now outline how TRMS concepts of registration, discovery, and invocation can be instantiated

4.2 Registration

In order to be discovered in the network, a tool first has to be registered at the Global Tool Lookup Service (GTLS). Here, the GTLS manages a global SNMP MIB database. This basically means that for SNMP integration, tools and their properties have to be specified in ASN.1.

For registration, main properties and functions have to be given. The publication of that data is taken over by the Local Tool Control Service (LTCS). Parameters which are available in the local MIB are send as SNMP variables via SNMP-SET to the GTLS. The GTLS registers the tool with the received data in its repository, which is organized as a MIB. Tool registration data include tool properties as well as host and operating system properties such as:

- name
- version number
- installation path
- list of accepted input and output formats with their version list of possible parameters
- tool classification: edit, display, convert,...
- interaction mode: batch, shell, X11, win

In addition to standard tool properties like name, version, installation path, and parameters, additional characterizations are required for remote access like the interaction mode. The tool classification is required for general tool queries when tools of specific categories are searched for. In our current environment, we have classification into: *display, editor, converter.* Nevertheless, the classification has to be generic and adapted to the individual needs in order to be extended on demand. The interaction mode defines the environment under which the tool can be executed. We distinguish basic batch tools with no display output (batch), tools where ASCII output is required (shell), output under X11 (x11), and output under windows (win). Based on this classification and on the available operating system, it can be decided if a remote invocation makes sense. For instance, in specific contexts (unavailability of Windows emulation on local host) it makes no sense to access a Windows interaction from a Sun workstation and it may only make sense to access X11 interaction from a PC when an X terminal emulation is locally available. For efficient resource management and increased performance, the following properties of the host IP address and the operating system and version have to be specified.

NAME	acroread	acroread	notepad
VERSION	3.0	3.0	4.0
TYPE	display	convert	edit
PATH	/usr/local/ Acrobat3/bin	/usr/local/ Acrobat3/bin	C:\winnt\ notepad.exe
INPUT	pdf ?1.1?1.2? 1.3?1.4	pdf ?1.1?1.2? 1.3?1.4	txt
OUTPUT	ps ?Level1 ?Level2	ps ?Level1 ?Level2	txt
FLAGS	"default" %s, "help"-h	"default" -toPostScript %s, help"-h	"default" %s
MODE	x11	Batch	win
HOST	131.234.80.6 1	131.234.80.61	131.234.80.66
OS_TYPE	sunos	sunos	winnt
OS_VERSION	5.6	5.6	4.0?3

Table 1: Selected Tool Properties

Table 1 gives examples for selected properties of 3 installed tools: acroread 3.0 in two flavours (display,

converter) and notepad 4.0. The table shows that acroread can be classified as a display for pdf and as a converter from pdf to PostScript. In both cases different input and output formats are accepted. They are enumerated with their version, which is separated by a '?', e.g., PostScript? Level1?Level2. In addition, tool parameters (flags) can be optionally specified where the default parameter (default) is mandatory and help parameter is recommended. Note here, that acroread has a different default parameter when being used as a PostScript converter. After registration, SNMP manages discovery and invocation of tools.

4.3 Discovery

TIS executes the discovery of a specific tool. In the following, we outline how discovery can take place based on SNMP message exchange. The general scenario starts with the specification of the *query* for the tool to be discovered. This query contains a description of the tool and is used by the GTLS to find an appropriate registered tool. In our environment, we can specify concrete tool names like "acroread" with optional version identifier like "3.0". Moreover, our concepts also support declarative queries by only specifying input and/or output format like "word6.0". Then, the GTLS checks for all available matches. It is possible to combine properties for queries like a tool of classification "converter" with input "PostScript" and output "pdf".

The query is sent by the client TIS to the GTLS by means of a GET message. The message includes the query request as well as additional static and runtime information of the client host like the operating system. The latter is required for later advanced tool selection. Based on the received data, the GTLS consults its MIB database. In general there can be several matches in the MIB, e.g., a simulator can be available on several servers. Then, an advanced selection process can be performed based on various optimisation criteria. For an optimised selection, TIS may consider actual runtime information as provided by the LTCS. Afterwards, the GTLS sends the tool reference to the client via a GET-RESPONSE, which forwards the result to the client.

4.4 Invocation

After discovery, the tool can be invoked. First, the input data, i.e., design data, are transferred to the tool by the means of ANTS. The ANTS implementation for the SNMP variant of TRMS is using the FTP [3] protocol for the actual data transfer. Once the input data has been transmitted to the tool machine, the client sends a GET message to the LTCS responsible for the selected tool. The LTCS invokes the tool using the input data from the ANTS transfer. After finishing the tool (successful or unsuccessful computation), the LTCS replies to the client by a GET-RESPONSE. Thereafter, the client starts the transmission of the output data via ANTS, which completes the discovery/invocation cycle.

5. Application

We present the application of the previously presented integration and management by the example of a distributed environment for PCB design. Due to steadily decreasing time budget allocated to the development cycle and due to the worldwide distribution of development teams, our environment demonstrates that we can achieve a high degree of efficiency when applying the presented concepts.

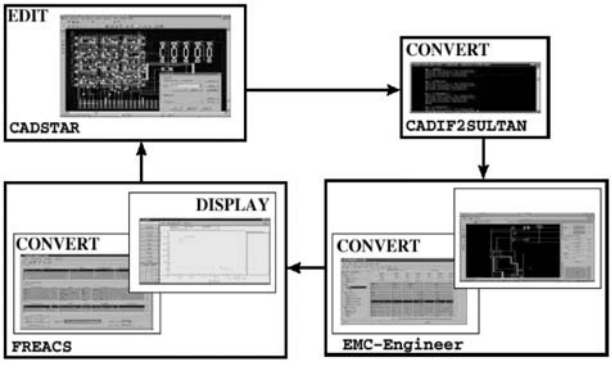

Figure 4: PCB Design Flow

Figure 4 sketches a typical workflow for a PCB design process utilising Zuken tools. The complexity here not lies 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 real challenge is the discovery of tools over the life cycle rather than the workflow definition and organisation.

The presented workflow depicts the processing steps from schematic entry over layout generation to final verification and simulation of analog properties. In our example, we apply tools from the Zuken Hot-Stage tool suite: CADSTAR-SI, EMC-Engineer, and the integrated simulators FREACS, Sigma, and ComoRan. Based on this tool suite, we focus on the CADSTAR integration and invocation where the application context is given by the design flow in Figure 4.

Our design flow starts with the CADSTAR design entry. In the Zuken tool suite it is denoted as a low level entry and available as a standalone version on PC. CADSTAR is a graphical capture for the complete design of PCBs. It exports its data in proprietary format as well in the widely known CADIF file format. The

CADIF2SULTAN converter transforms the output file into SULTAN format, which is required for EMC-Engineer input. The EMC-Engineer provides input data through manual interaction, e.g., assignment of component properties, for later simulation. The EMC-Engineer is executed on a UNIX-Workstation. After simulation, the simulation results are displayed by ANARES.

5.1 Tool Integration

For the integration of these tools, their properties have to be given as previously introduced in order to register them at the GTLS. We only outline the integration for CADSTAR. The other tools are specified correspondingly. As already given in Table 1, we have to specify the name, version, classification, installation path, input and output formats, host IP, interaction mode as well as the operating system with version (and service pack or patch level):

```
cadstar;edit;c:\Programms\CADSTAR\cadstar.exe;
4.5.1;.pcb,.scm,.paf;.pcb,.scm,.paf?3,.paf?4;
win;196.22.22.22;winnt;4(3),
```

Here, we use a shorthand format with no type information. The different fields are separated by semicolon. Versions are separated by a question mark. The example shows that CADSTAR exports files with ".pcb", ".scm" and ".paf" extension, where CADIF files (with extension ".paf") of version 3 and 4 are supported. We also see here that CADSTAR requires a Windows display (win).

5.2 Queries

As a short example, we outline a short sequence with four queries for tool discovery based on the previously integrated tools:

```
1. CLASS=edit;INPUT=.pcb;OUTPUT=.paf?3;
   OS_TYPE=winnt
2. CLASS=convert,INPUT=.paf;OUTPUT=.sultan
3. NAME=emc-engineer;INPUT=.sultan;
   OUTPUT=.dia;MODE=X11;OS_TYPE=SunOS
4. CLASS=display;INPUT=.dia
```

The queries are given as an ASCII specification where the preceding type identifiers correspond to those in Table 1. The first query specifies an editor under Windows NT, which imports .pcb files and exports CADIF version 3.0 files (files with extension .paf). The second query asks for a converter from CADIF to SULTAN. Thereafter, the EMC-Engineer under SunOS is searched for. The final query checks for a program displaying .dia-files. Note here, that it is sufficient only to specify input and output without specifying the program's name. When specifying such a sequence, our current system takes the output data from the previous call as input so that simple design flows can be easily implemented. Implementation of more complex design flows requires the integration of an advanced workflow management tool.

6. Conclusions

In this paper, we have presented an approach for dynamic tool management, namely TRMS, and applied its concepts to an SNMP based variant. The introduced variant was outlined by the example of tools of the Zuken Hot-Stage tool suite for PCB design.

We have applied SNMP to tool integration and mainly addressed tool administration in complex intranets. For site-spanning tool integration, one has to consider open and known security problems in order to highly protect the exchanged IPs. Due to those problems, SNMP-based solutions seem to be less applicable for open networks. Nevertheless, also XML-based alternatives with SOAP servers currently have significant unsolved security problems. We see that network-based solutions for non-secure environments still require significant investigations in authentication and encryption when exchanging control and highly sensitive (i.e., IP-protected) design data.

We have resolved the security issues by means of an adaptable transport layer, i.e., ANTS. Here, we have also developed a solution for bridging firewalls via additional proxy servers and encryption in order to provide site-spanning solutions with an emphasis on security issues.

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.

References

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

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

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

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

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

[6] Stallings, W. B.: SNMP, SNMPv2, SNMPv3 and RMON 1 and 2. Addison Wesley Longman Inc., Reading, Massachusetts, 1999.

[7] Sun Microsystems: Jini™ Specifications v1.2, http://wwws.sun.com/software/jini/specs/, 2002..

[8] Thronicke, W.; Fox, W.; 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.

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