

CAD Design Flows Development in a Cross-Platform Computing Environment

Shesha Krishnapura, Computing Technology/Design Technology, Intel Corp.

Ty Tang, Computing Technology/Design Technology, Intel Corp.

Vipul Lal, Computing Technology/Design Technology, Intel Corp.

Index words: NT*, UNIX*, mixed-flow, cross-platform

* All other brand names are the property of their respective owners.

Abstract

With the advent of low-price, high-performance Intel® architecture workstations together with Microsoft® Windows NT* operating systems (referred to as IA-NT from here on) that support Microsoft productivity tools, the IA-NT workstation has become the preferred desktop for CAD design engineers. However, due to the complexity of migrating UNIX*-centric legacy CAD tools and scripts to an NT environment, a mixed operating system platform for CAD design has become a computing reality. This paper describes the innovative technical solutions for a production-capable NT-UNIX cross-platform CAD design flow environment for development, maintenance, and deployment activities. Although the target systems chosen are the ones used in Design Technology at Intel, our solutions are applicable to other cross-operating systems.

The NT-UNIX platform poses various technical challenges when developing the CAD design flows consisting of tools from both platforms. These tools have to work together on a shared design database while effectively utilizing common infrastructure scripts, despite the fact that each computing platform supports a different scripting environment.

To meet some of these challenges, we developed two technologies that allow seamless integration of software, designed for either the UNIX or NT platform, into a platform-independent production usage environment. These two technologies have been used to port more than 45 tools made up of more than 3,000,000 lines of code from UNIX to NT, and to execute more than 1,000 test flows, as well as to develop a few mixed NT-UNIX applications.

Introduction

Traditionally, Intel has been using high-end UNIX*-based RISC workstations for microprocessor design activity. However there are emerging compelling reasons why this traditional design environment should change to incorporate the IA-NT workstation. The main reasons for this change are as follows:

1. The advent of low-priced, high-performance Intel architecture workstations coupled with Windows NT* operating systems (IA-NT), which make IA-NT a formidable alternative to UNIX-RISC workstations.
2. The maturity of NT towards a stable, scalable operating system that supports high-end CAD applications.
3. Next-generation Intel® CAD tools are moving from the legacy single CAD design environment, which is driven by scripts through command line interface, to a new kind of CAD environment. This new environment incorporates CAD design tools and office productivity tools into a tightly integrated visual cockpit that uses modern distributed computing components and Internet-driven technology that support multiple simultaneous CAD design environments. The IA-NT workstation provides an excellent development and design environment for these new-generation CAD applications.

In reality, we cannot convert the existing UNIX-RISC-based design flow to an IA-NT base in a single step due to the following reasons:

1. Many of the design automation tools are based on UNIX-centric scripts that are not easily ported to an NT environment.

* All other trademarks are the property of their respective owners.

2. Some of the internal CAD tools are tightly integrated with external CAD tools that are not available on NT.
3. The current design team skill set is UNIX-centric and would need to be updated for NT-centric design work.
4. Microprocessor design teams in the midst of projects cannot handle a change in environment due to the nature and complexity of such a change. This means that IA-NT can only be used on new projects.

The solution to converting to IA-NT therefore is to have a transition phase to support a production-capable mixed NT-UNIX design flow environment. To achieve this transition phase, the following needs to be done:

1. Build a robust NT-UNIX mixed computing environment with a shared file system. This will support the IA-NT Desktop with backend IA-NT and UNIX compute servers (see Figure 1).
2. Migrate high compute usage CAD design flows, comprised of tools and scripts from UNIX, to native NT and keep UNIX-centric legacy tools, which use low computing power, on UNIX.
3. Develop cross-platform utilities for production use for a mixed NT-UNIX design environment where CAD tools on NT and UNIX are used in a seamless fashion.
4. Develop next-generation CAD tools native to IA-NT.

In this paper, we limit our discussion to the NT-UNIX cross-platform environment. We outline the various challenges faced and the techniques employed for the production use of a mixed NT-UNIX environment for CAD tool development in Design Technology.

Overview of Existing Design Environment

The existing design environment at Intel is UNIX-centric. It consists of tightly integrated CAD tools, scripts, and design data that are in the order of tens of millions of lines of code. A significant portion of the tools and scripts are legacy codes that have been shared among generations of engineers and are hard to replace.

From a high-level point of view, the design tools, scripts, and data can be grouped into following four main categories, in which the first two categories are part of the CAD design tools development environment and the last two categories are part of the microprocessor design project environment:

1. *Design Tools*: A set of internally developed and

external vendor CAD tools that are tightly coupled by UNIX-centric “glue” scripts into a tool suite.

2. *Gluing Utilities*: These are scripts and small programs that integrate the various tool components in a tool suite into a functional design environment for microprocessor designs at Intel. Some of the tasks include internal to external tools data format translation, data extraction from netlists, simulation, wave form analysis, and design database management.
3. *Microprocessor Design Project Utilities*: Sets of scripts and programs developed by design automation engineers in design projects to validate design logic, process design data, generate design models, analyze performance, etc.
4. *Designer Private Utilities*: Scripts and programs developed by individual design engineers to aid them in their work such as analyzing and filtering design data, generating test stimuli, running tools in a particular sequence, etc.

This complex environment is represented in Figure 2.

In the next sections we describe the technical challenges we faced while developing CAD design flows in a mixed NT-UNIX environment, and we outline some of the innovative technical solutions we adopted to overcome these challenges.

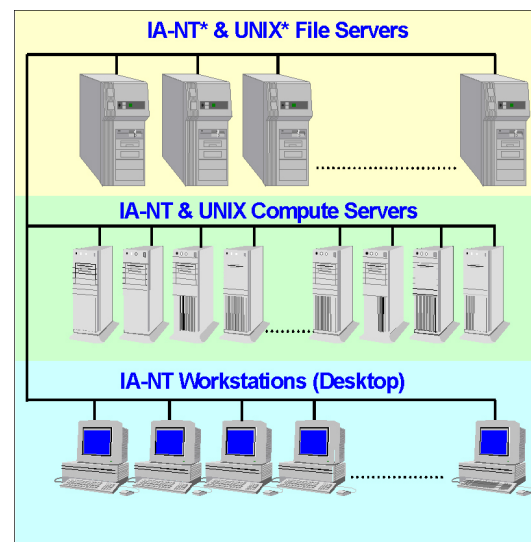


Figure 1: Simplified view of NT-UNIX mixed environment

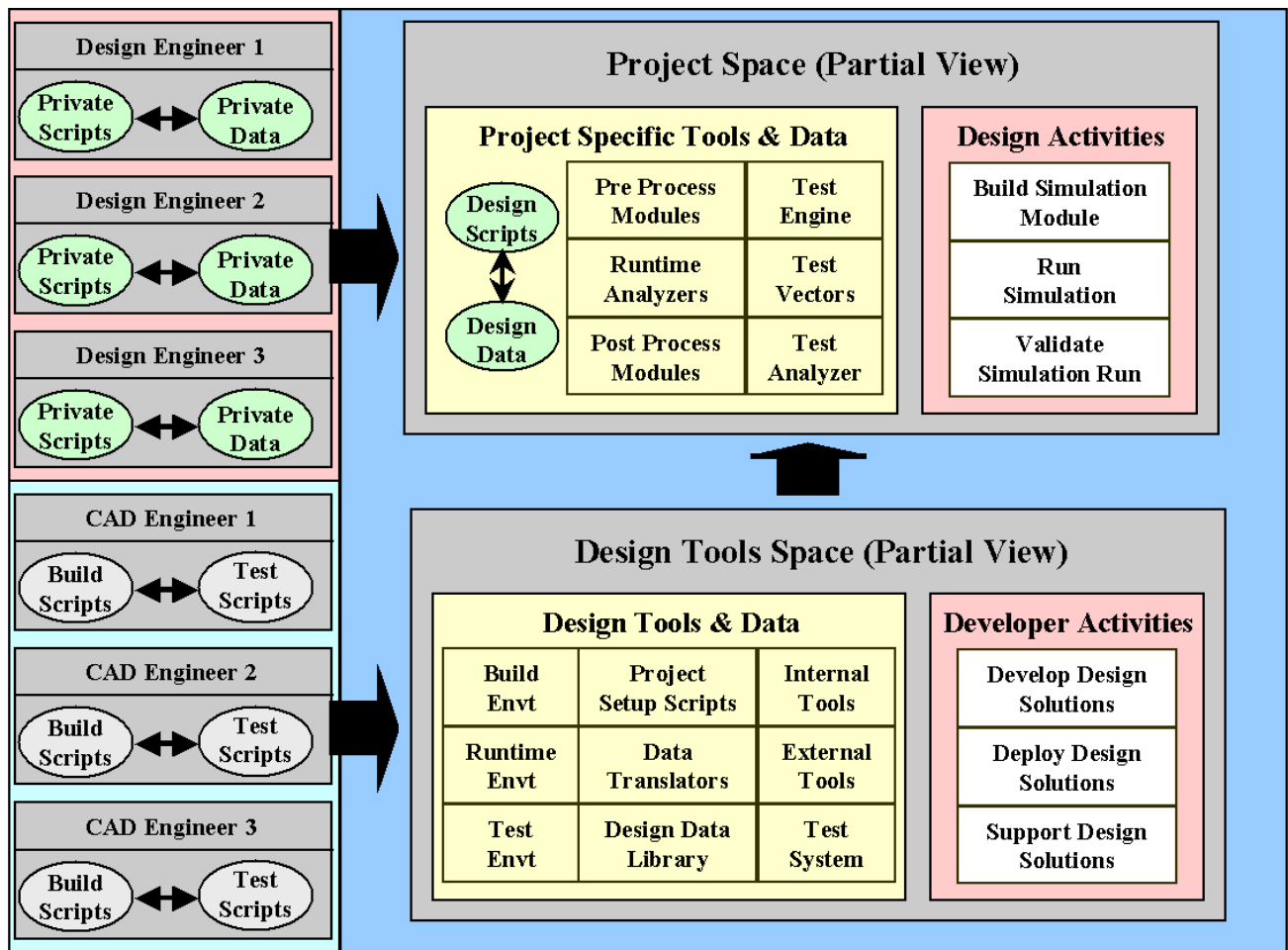


Figure 2: Simplified view of UNIX-based design environment

UNIX* to NT* Migration and NT-UNIX Cross-Platform Challenges

The complexity of the existing UNIX-based design environment at Intel makes it impractical to convert to a homogeneous NT environment in one step. The viable solution is to develop a heterogeneous UNIX-NT integrated design environment and convert more and more UNIX-centric components to IA-NT over time.

As with the migration of any operating system, the migration of an existing UNIX-based CAD design environment to a mixed NT-UNIX design environment presents us with a number of technical challenges of which the major ones are as follows:

1. New applications developed for or ported to NT depend on reusable components available only on UNIX. These reusable components include

- external vendor tools and libraries
 - internal libraries that have not been ported or are not portable to NT
 - legacy design data in a database that can only be accessed on a specific UNIX platform
1. Design flows that execute a set of tools, all of which may not be available on a single platform.
 2. The demand for common infrastructure scripts to drive the tools on both UNIX and NT is difficult to meet since the scripting environment on NT is not fully mature and is not 100% compatible with the UNIX scripting environment.
 3. How to maintain a single test system and test vectors for cross-platform validation.

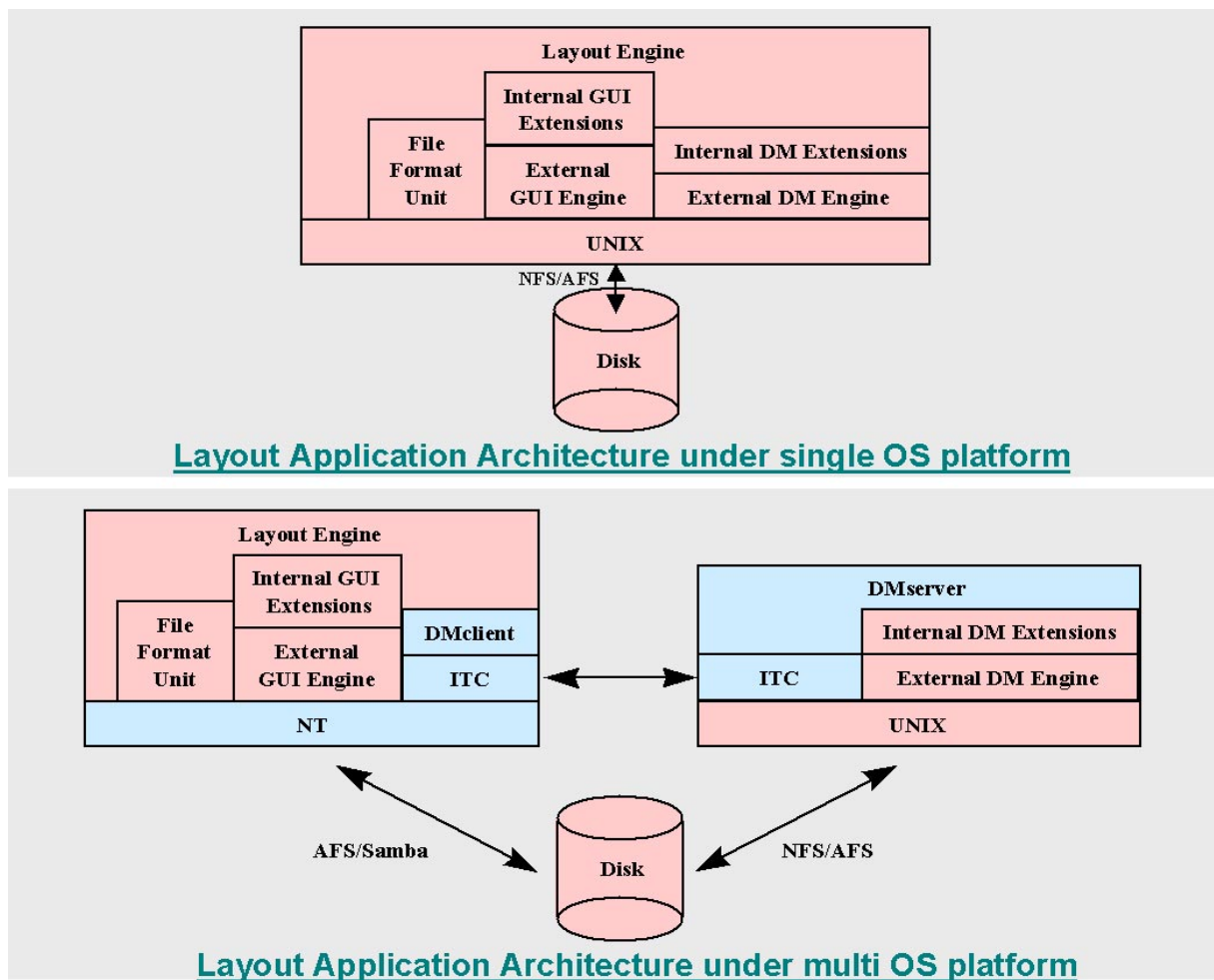


Figure 3: CAD layout application in single and multi OS platforms

Innovative Solutions for an NT-UNIX Cross-Platform Development Environment

The compile-time client-server model and the run-time client-server model were developed and integrated into the mixed NT-UNIX design environment to overcome the challenges detailed above. These models allow for the seamless integration of software developed for either UNIX or NT platforms into a platform-independent NT-UNIX CAD development and design environment.

In the following sections, we describe these technologies and outline the problems they solve, the architectural details, and our pilot results.

Compile-Time Client-Server Model

The compile-time client-server model allows UNIX*-based layered CAD applications to be migrated to IA-NT architecture in a situation where a complete

migration to NT* is not practically feasible due to either the non-availability of vendor-provided components or non-portable internal legacy tools/libraries.

The UNIX-based layered application for a CAD layout capability is described in Figure 3. In this application, the Data Management (DM) engine is a vendor library integrated with an internal application not native to an NT platform. The figure shows the client-server application designed for an external DM engine using the innovative solution Inter Tool Communication Library (ITC) and integrated with a layout application. The architecture and functional components of the ITC library are described below.

The Inter Tool Communication Library for Compile-Time Client-Server Architecture

An important component of the compile-time client-server model is the ITC library, which provides inter-tool communication functionality for client-server applications.

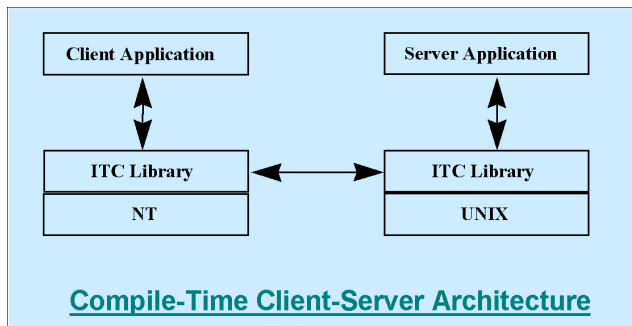


Figure 4: Compile-time client-server architecture

The aim of the ITC library is to provide the following:

- a reliable, recoverable, trackable, and efficient NT-UNIX communication channel with simple APIs for point-to-point inter-tool communication between a client running on an IA-NT system and a server running on a UNIX system
- transparent handling of Byte ordering between RISC and IA architecture
- transparent handling of NT-UNIX path conversion

- interface code to engineer compile-time client-server model for rapid implementation with built-in message build/extract capability and remote function execution mechanism

In this paper we do not focus on the implementation details of the ITC library APIs. Nonetheless, the next two figures clearly illustrate the interaction between an NT-client application and a UNIX server to remotely execute the extended features provided by libraries available only on UNIX. Figure 5 shows the interaction between the various ITC APIs on the NT client and the UNIX server describing the execution flow. Figure 6 shows a sample DMclient-DMserver application of which the UNIX-DMserver provides the NT-DMclient with complementary database access and management capabilities supplied by a UNIX-centric DM library not available on the IA-NT. The master server is capable of supporting multiple clients as shown in the figure.

The summary of the interaction between the client and server is as follows:

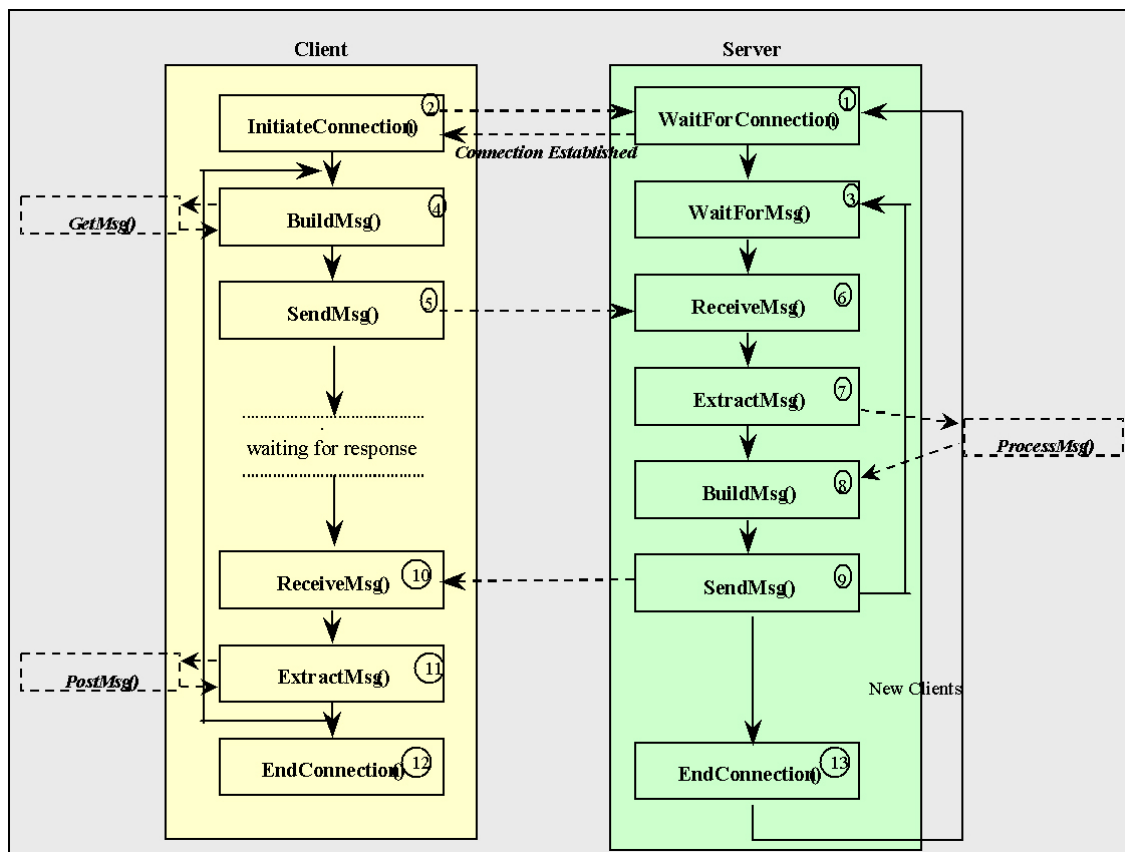


Figure 5: Interaction between NT client and UNIX server execution flow

- The client connects to the DMserver.
- The server forks off a slave server to process the client.
- The master server is free to handle incoming clients.
- The slave server, which is dedicated for each client, exits after servicing the client.

Over the past two years, the CAD organization at Intel has successfully migrated many tools to IA-NT and developed new IA-NT resident tools using this client-server technology.

The compile-time client-server technique is valid for the migration of any one platform to another platform. Until the complete set of native CAD tools and libraries are available to IA-NT, including the external vendor tools, this technology remains a good intermediate solution for the unavailable software components during UNIX to IA-NT migrations.

Run-Time Client-Server Model

Run-time client-server platform independent execution capability is a transition technology to enable the integration of a mixed NT*-UNIX* design environment. Its main aim is to address the very problems that prevent the design organization from moving di-

rectly to a pure IA-NT compute environment: not having the complete set of design CAD tools on IA-NT and having legacy UNIX-centric design data, infrastructure, and gluing scripts that cannot be ported to IA-NT.

The idea behind run-time client-server platform independent execution technology is to extend the remote procedure call concept to a remote execution environment. The result is a tool environment that allows transparent execution of CAD tools in a mixed NT-UNIX environment. In this cross-platform environment, the user will be working on an NT desktop and executing design commands from a remote UNIX Xterm using the same familiar design flow written in scripts. The user view is illustrated in Figure 7.

Run-Time Client-Server Architecture

The main components of the run-time client-server platform independent tool execution environment as shown in Figure 8 are a UNIX server daemon, an NT server service, and CAD tools and scripts that are set up to run in this environment.

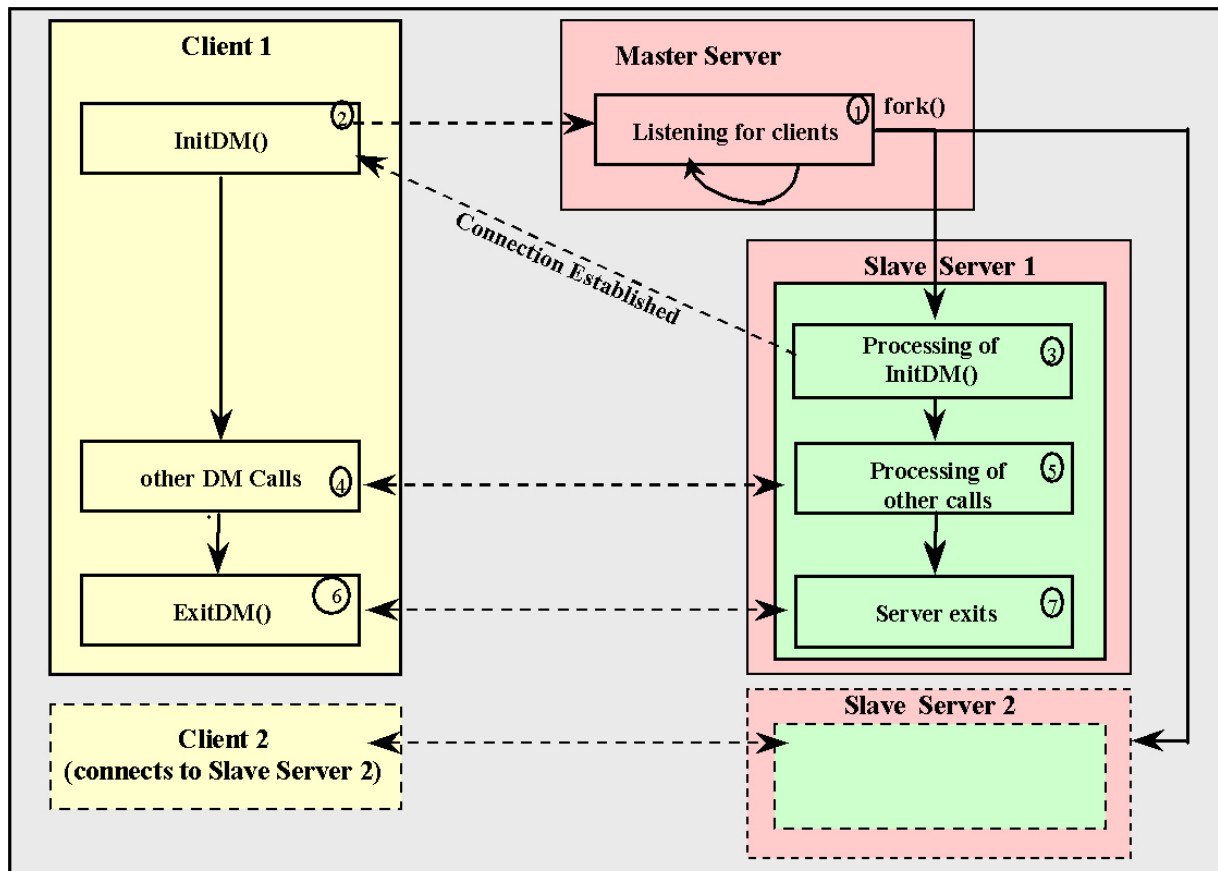


Figure 6: Execution flows between IA-NT DMclient application and UNIX DMserver

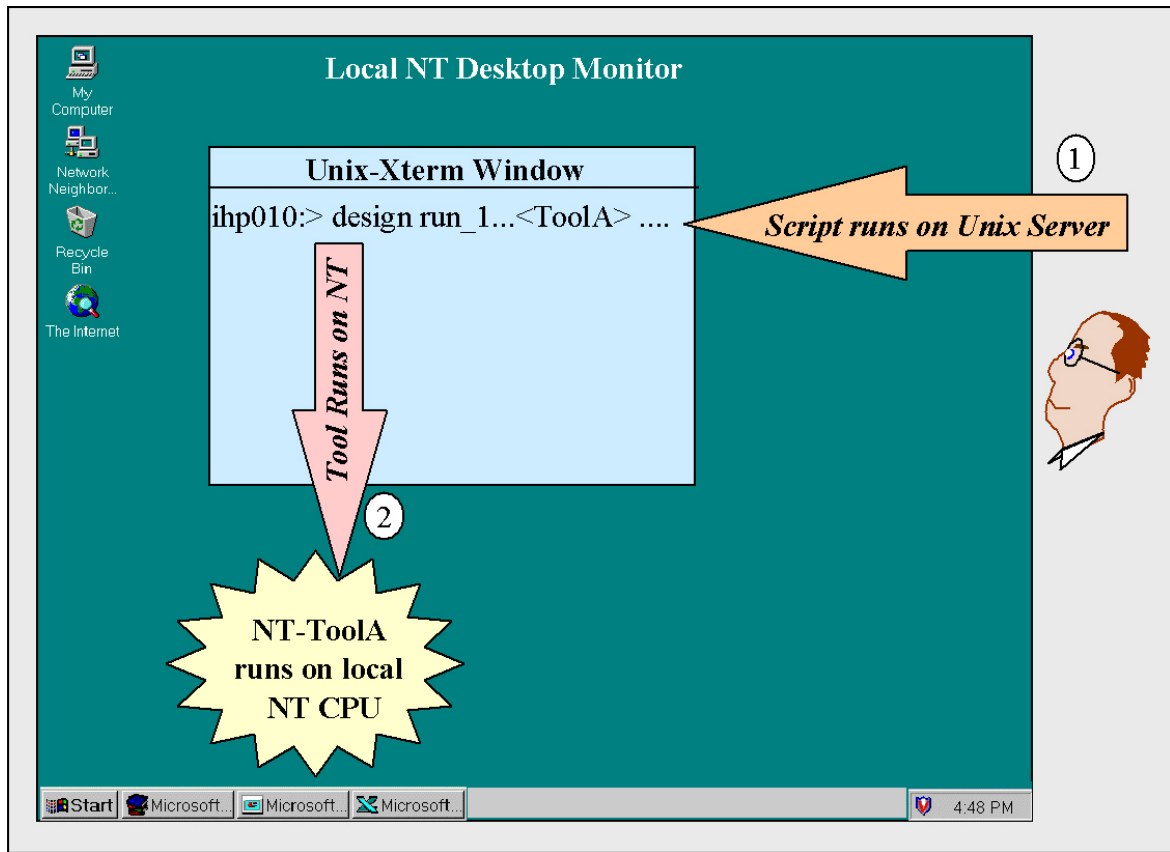


Figure 7: User view of run-time client-server platform independent execution environment

Every tool executable in this environment has two complementary components, namely the tool executable itself and the tool stub, each of which is running on different platforms. The tool stub is a wrapper that launches the tool executable on the remote system.

the user, duplicates the run-time environment (with the necessary system path conversion and platform specific adjustment), and invokes the actual tool executable. The server also handles the routing of STDIO streams to the tool stub and passes the exit code of the tool executable to the tool stub. The tool stub exits with the same exit code.

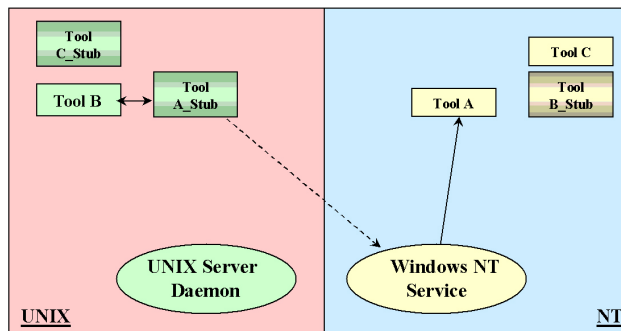


Figure 8: Overview of the run-time client-server platform independent execution environment

When a tool stub is executed on the local system, it takes a snapshot of the run-time environment on the local system and communicates this information to the server daemon (or service). The server impersonates

A sample CAD design flow consisting of three tools working in a cross-platform execution environment using the above stated run-time client-server architecture is shown in Figure 9.

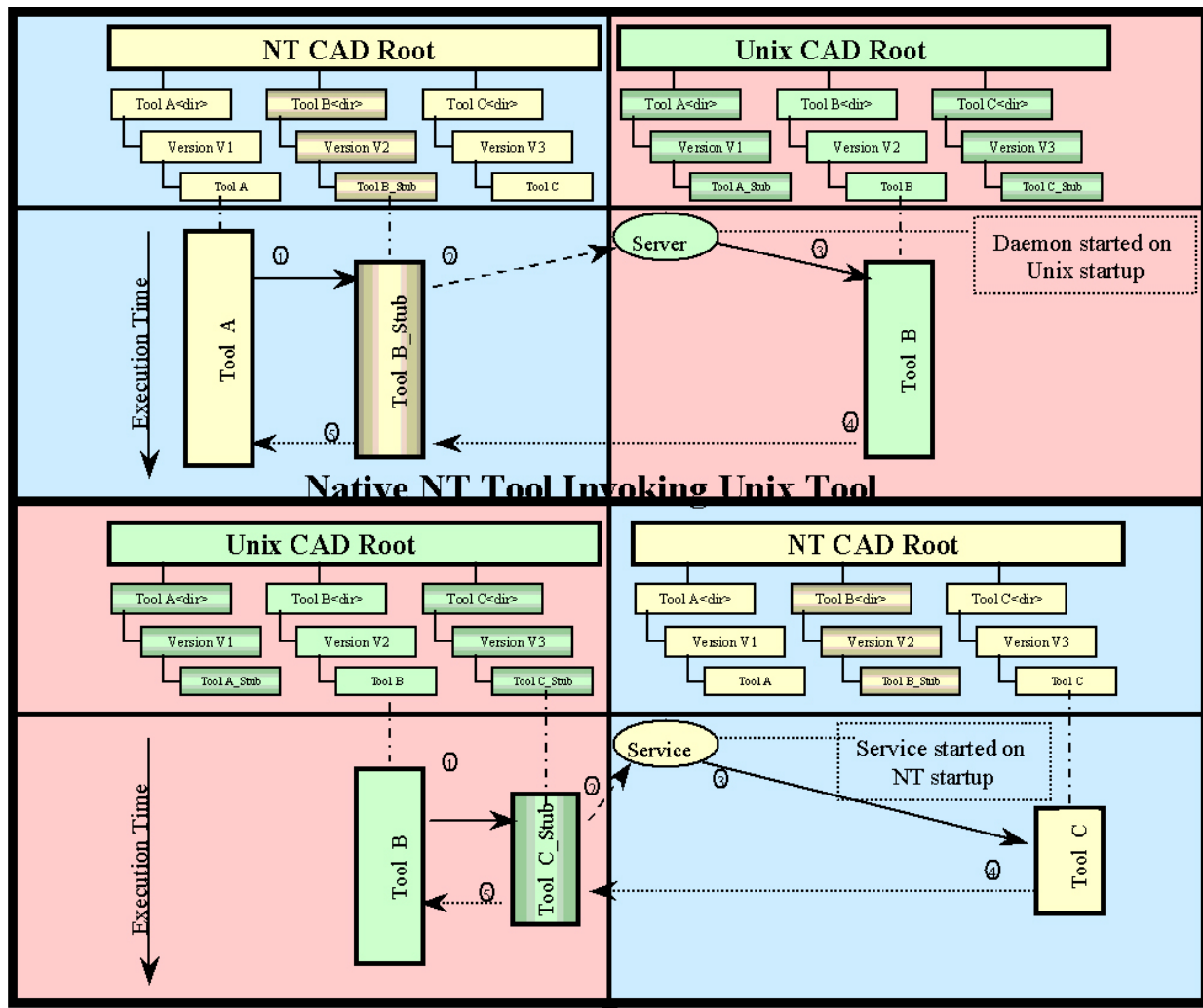


Figure 9: Flow execution using platform independent execution capability

The run-time client-server platform independent execution environment provides the following features to enable transparent remote tool execution:

1. Conversion and customization of the run-time tool environment. The run-time environment on the local system is duplicated on the remote system for tool execution.
2. NT and UNIX shared file-system support so that it can handle the tool stubs, executables, data, and output files residing on shared file systems.
3. Handles the redirection of standard input, standard output, and standard error streams between the tool stub on the local system and the tool executable on the remote system.
4. Provides the capability to execute binary files on the remote system (UNIX or NT).
5. Provides the capability to launch the scripts on the

remote system (UNIX or NT).

6. Duplicates the shared file system credentials for transparent user access from local to remote system.
7. Impersonates the local system user on the remote system and executes the tool with the same user credentials.

Figure 10 details the architecture of the run-time client-server capability.

Over the past year, the CAD organization at Intel has successfully used this technology to validate more than 40 IA-NT ported CAD tools using the same test system from UNIX by running more than 1,000+ test flows in a seamless mixed NT-UNIX platform. These test flows also used some of the UNIX-centric tools that are not yet available on NT.

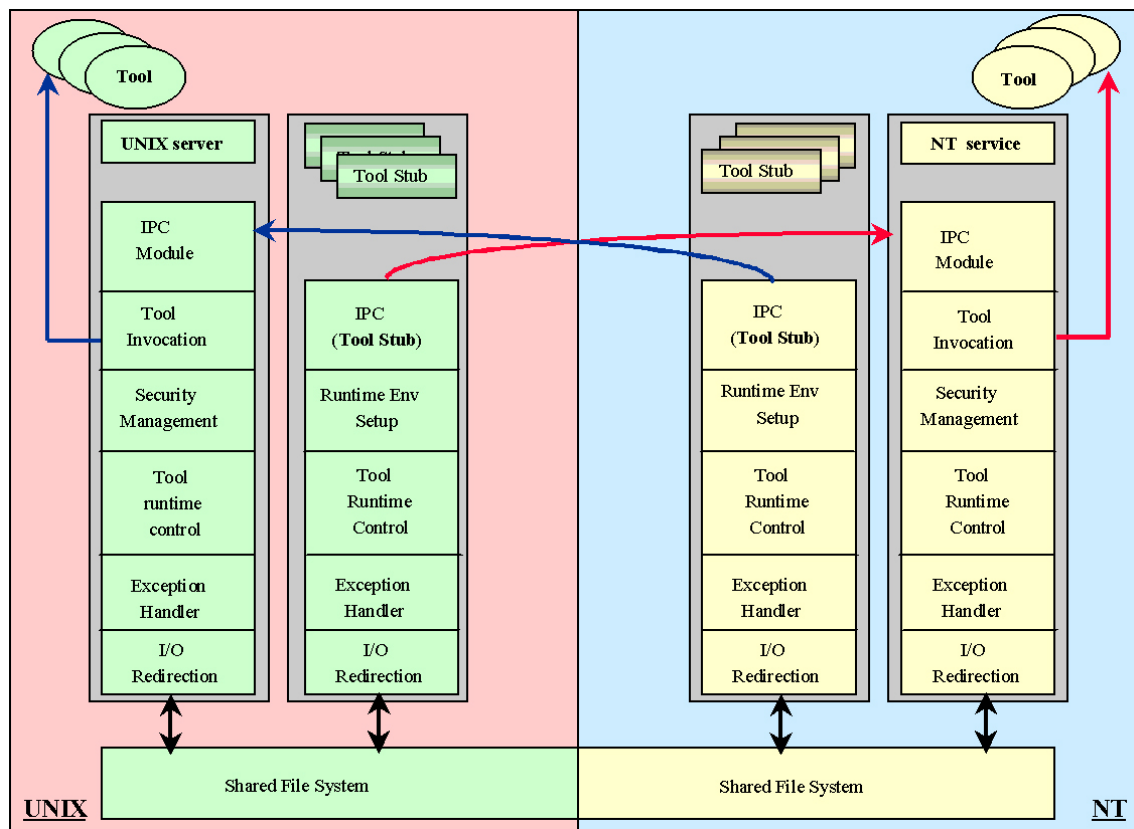


Figure 10: Architecture overview of run-time client-server platform independent execution technology

Conclusion

This paper describes the innovative technical solutions employed by Intel's CAD organization to develop a production capable NT*-UNIX* cross-platform CAD design flow environment for development, maintenance, and deployment activities.

This work is important for Intel to continue to maintain a powerful, effective, and competitive design environment for the future generation of microprocessor design projects. The simple but powerful technique described in this paper is easily proliferable to future Intel® microprocessor design project environments as Intel moves from a UNIX to an IA-NT design flow environment.

Acknowledgments

We thank DT and Computing Technology management for providing the opportunity to work on this exciting project. We are grateful to our department manager Tae Paik for his vision, inspiration, patience, and continuous encouragement during this project. We

wish to acknowledge Tae Paik for the brainstorming session that helped us to come up with a run-time client-server model and Daaman Hejmadi who worked with Shesha Krishnapura (author) on the initial concept of a compile-time client-server model. Thanks to Sunil Bhasin and Gil Kleinfeld for applying the ITC Library to new applications. Thanks to Athena-NT Technical Working Group members for ratifying the technical concepts and documents. Special thanks to Tzvi Melamed, Greg Hannon, Chenwei Chiu, and Ming Lin for their technical feedback and encouragement during the development work. Thanks to Srinu Rama for implementing parts of the run-time client-server utilities and conducting performance tests. Thanks to Athena and Nike development engineers who are using the compile-time and run-time technologies in their products.

References

- [1] Alexander Wolfe, "Intel taps Windows NT in design-software shift." *EETIMES*, issue 948, April 7, 1997, pages 1, 148.

[2] Richard Goering, "Can NT win in IC design?"
EETIMES, issue 992, page 70.

Authors' Biographies

Shesha Krishnapura is a staff engineer and manager for the Software Platform Engineering Group in CT/DT. He received an M.S. degree in computer science from Oregon State University in 1991 and a B.E. in electronics engineering from the University Vishveshvaraya College of Engineering, India in 1986. He has worked on digital switching system software and CAD platform tools. His interests are in client-server modeling and platform migration. He joined Intel in 1991. His e-mail is shesha.krishnapura@intel.com.

Ty Tang is a senior software engineer in the Software Platform Engineering Group in CT/DT. She received her B.S. in computer science and computer engineering from the University of California at Los Angeles in 1990. Her expertise is in various levels of system programming under UNIX* and NT* platforms, software engineering and migration of CAD tools and system software to various platforms. She joined Intel in 1994. Her e-mail is ty.tang@intel.com.

Vipul Lal is a senior software engineer in the Software Platform Engineering Group in CT/DT. He received a B.S. in computer engineering from the University of Pune, India in 1993. He has developed system and application software for various flavors of UNIX* and NT* operating systems. He joined Intel in 1996 and is now working on cross-platform system and application software development. His e-mail is vipul.lal@intel.com