Using SoftBench to Integrate Heterogeneous Software Development Environments

Migrating from mainframe-based computing to client/server-based computing can result in a heterogeneous collection of machines that do not interoperate, forcing software developers to deal with unfamiliar system commands and systems that cannot share data. A SoftBench control daemon is described that enables developers to integrate heterogeneous computing systems into efficient, tightly coupled software development environments with consistent, easy-to-use graphical user interfaces across all machines.

by Stephen A. Williams

Many companies today are migrating from mainframe-based computing environments to client/server-based technologies using various workstations and PCs. They are attracted to the client/server architecture because of industry claims of benefits like increased efficiency, lower operating costs, and less reliance on a particular vendor.

Often, however, the result is a heterogeneous collection of machines that do not interoperate well. Because the operating systems on the disparate machines all come with their own sets of tools, software developers must learn a new set of commands for each system that they use. In addition, developers must deal with the inconsistencies that arise when applications available on one system are not available on another and when data cannot be shared between machines because the different toolsets cannot communicate.

To solve these problems, the advanced system development and integration division of Science Applications International Corporation (SAIC) uses Hewlett-Packard's SoftBench product to integrate its customers' diverse systems into efficient, tightly coupled software development environments with consistent, easy-to-use graphical user interfaces. This article discusses why and how SAIC uses SoftBench to solve its customers' multiplatform software development problems. The article details some of the common pitfalls encountered when developing software in an open systems environment, explains how SAIC deploys SoftBench to integrate such systems, and concludes by discussing the benefits of such an integration.

Open Systems

Companies are adopting client/server-based open systems for a wide variety of reasons. Some companies hope to increase computing efficiency by distributing the data and processing load, thereby providing faster response times and quicker access to system resources. Other companies want to lower their development costs by using lower-cost, yet faster workstations and PCs. Yet others must move to open systems to remain compatible with their customers and keep a competitive edge in the marketplace.

While migrations to open systems can provide great dividends, they can also become more unwieldy than the systems they replace. Many client/server topologies contain a wide variety of machines, such as high-end servers running the UNIX® operating system, PCs running Microsoft® Windows, and legacy systems running proprietary operating systems. In addition, even similar machines will often run different operating systems (e.g., variations of the UNIX operating system) or even different versions of the same operating system. The resulting heterogeneous collection of machines makes it difficult to create an efficient and cooperative software development environment. Fig. 1 depicts an example of such an environment. Note that most applications cannot communicate with each other.

Although many open system standards exist to help such diverse collections of machines communicate, most of them are low-level network standards that simply provide a way for bits to be transferred between machines. The open systems community still lacks accepted high-level application standards that would allow disparate programs to interact with each other. Thus, applications from different vendors often cannot interoperate, which greatly restricts the benefits of implementing many client/server solutions. This lack of communication also means that data must be replicated across machines, wasting resources and increasing the risk of data inconsistency.

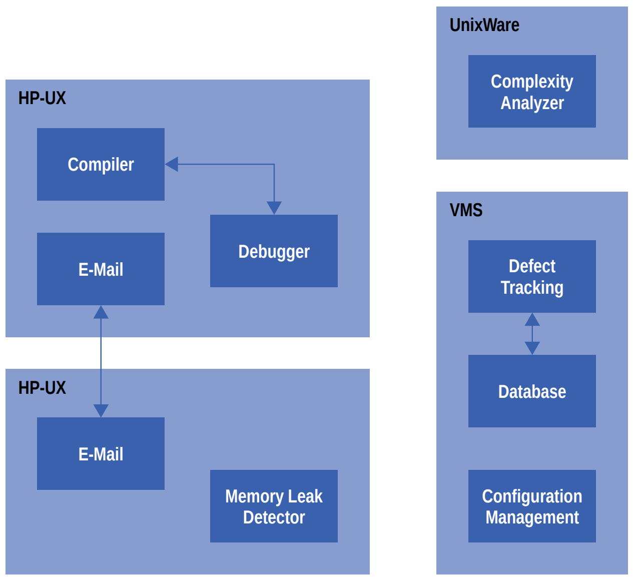

Fig. 1. *An example of a heterogeneous collection of machines in which the applications on different systems cannot cooperate or communicate with each other.*

Another problem with developing in a multiplatform, multivendor environment is the lack of consistency in the user interface to the systems. Because developers must deal with multiple operating systems, they have to learn how to operate each system's interface individually. This can be an especially formidable task considering the arcane commands used by some operating systems and the differences between file systems across platforms (i.e., hierarchical versus fixed depth, / versus \, etc.). Developers must also remember which system contains each application that they need, where it is located, and how to start it.

Furthermore, developers in a heterogeneous environment must learn how to operate the different user interfaces for each of the applications that they use. While some applications now have elegant graphical user interfaces, the look and feel of each system are often different. Also, many applications do not have graphical user interfaces at all, which requires that developers memorize command-line options to these programs. These inconsistencies not only lengthen a developer's learning curve, but also make developers less efficient when switching between applications.

Integration

Previous issues of the HP Journal have described how to use SoftBench to integrate disparate applications running under the HP-UX* and Solaris operating systems.[1,2,3] In this article we will concentrate on how to use SoftBench to integrate applications running on other platforms. The key to accomplishing this integration is to port SoftBench's subprocess control daemon (SPCD) to each operating system that is to be integrated. The SPCD provides a standard, robust, and secure method of executing subprocesses on remote systems. This is accomplished by providing an API through which encapsulations can interact with the SPCD over a network socket connection. Through the API, encapsulations can instruct the SPCD to start or stop a subprocess on the remote machine, send input to a subprocess, and receive output from a subprocess.

Thus, once the SPCD is ported to a given system, encapsulations† can be written for applications running on that system as easily as if the applications were running on an HP-UX or Solaris system running SoftBench.

Why SoftBench? There are several reasons why SAIC chose to use SoftBench to integrate heterogeneous software development environments. First, the standard SoftBench environment comes with a rich set of state-of-the-art software development tools, all of which use a consistent, easy-to-use graphical user interface. In addition, SoftBench provides a graphical user interface to the operating system (via the SoftBench development manager) which hides many of the intricacies of the operating system and its file system.

Another advantage to using SoftBench is the framework for interapplication communication it provides through SoftBench's broadcast message server. This framework allows applications with no direct knowledge of each other to communicate and therefore interoperate. This functionality allows one application to be substituted for another with no adverse effects on other applications. It also allows new applications to be integrated into the environment without making any changes to existing applications.

Probably the most important reason to use SoftBench is its extensibility. Through the use of the encapsulator library, which provides functions to communicate with the SPCDs, the SoftBench environment can be extended to include non-SoftBench applications. In addition, the encapsulated applications can run on any operating system to which the SPCD has been ported.

† A SoftBench encapsulation means integrating a tool into the HP tool integration architecture.

Using SoftBench for Integration. Given the above reasons for using SoftBench to integrate a heterogeneous software development environment, how does one go about implementing such an integration? The first step is to install SoftBench on at least one HP or Sun workstation. Note that it is not necessary to place such a workstation on each developer's desk because SoftBench can be run remotely using the X Window System and developers can use any machine running an X server. This includes DOS, Windows, MacOS, and most versions of the UNIX operating system. Thus, a company implementing a SoftBench environment can probably leverage much of its existing hardware inventory to keep costs down.

Next, the SPCD needs to be ported to each operating system in the environment that contains applications that need to be integrated. Of course, there's no need to port to HP-UX or Solaris since SoftBench (and thus the SPCD) already runs on those systems. As discussed earlier, the SPCD provides a standard method that SoftBench applications can use to execute subprocesses on remote systems. Although other methods of remote subprocess control could be used in such an integration, the SPCD is probably the best choice because it is specifically designed to work with SoftBench. Also, note that there is no need to port all of SoftBench since only the SPCD is needed for remote subprocess control.

Because the source code for the SPCD is not freely available, the SPCD can only be ported by Hewlett-Packard or its authorized agents. SAIC has been granted such authority in the past to complete SoftBench integrations for a number of its customers. The operating systems to which SAIC has already completed the SPCD ports include:

- UNIXWare
- MP-RAS
- VMS
- Pyramid DC/OSx
- Stratus FTX
- Windows NT
- Tandem Guardian
- Tandem OSS.

A port to MVS was started but not completed.

As can be seen from the diversity of the operating systems to which the SPCD has already been ported, the SPCD code is quite portable. However, there are a number of requirements that the SPCD makes of a target operating system. The list below details the basic requirements that SAIC uses to determine the level of effort in an SPCD port:

- An ANSI C compiler
- A C++ compiler
- A Berkeley-type TCP/IP sockets capability
- The capability for a process to start up and communicate with several subprocesses like the UNIX fork() and pipe() system calls
- The capability for a process to detect input from several sources at the same time like the UNIX select() system call
- An interface that allows system calls to be made from C
- A way to set the environment of a controlled process like the UNIX getenv() system call
- Functionality similar to the UNIX inetd server
- Network File System (NFS) capability.

Note that the SPCD has been ported to environments that do not have fork(), select(), the inetd server, or NFS. While these items do make the port much simpler, it is still possible to port to environments that do not include all of the items listed above.

Once the SPCD has been ported to the appropriate operating systems, custom encapsulations must be written for each of the applications to be integrated into the SoftBench environment. Each encapsulation's job is to act as an intermediary between a non-SoftBench application and the SoftBench environment, making it look like the application is a fully integrated SoftBench tool (see Fig. 2). Performing this job entails a number of responsibilities, such as starting the application to be integrated, establishing a connection to the SoftBench environment, and sending the appropriate notification messages to SoftBench whenever the encapsulated application performs an action about which another tool might want to know. Furthermore, the encapsulation must listen for messages requesting a service of the encapsulated application and then instruct the application to perform the requested task.

To simplify the process of writing an encapsulation, SoftBench comes with an encapsulator library that provides an easy-to-use API to the SoftBench environment. The encapsulator library provides functions to:

- Send and receive SoftBench messages through the BMS
- Control remote subprocesses using the SPCD
- Create graphical user interfaces that are consistent with other SoftBench tools.

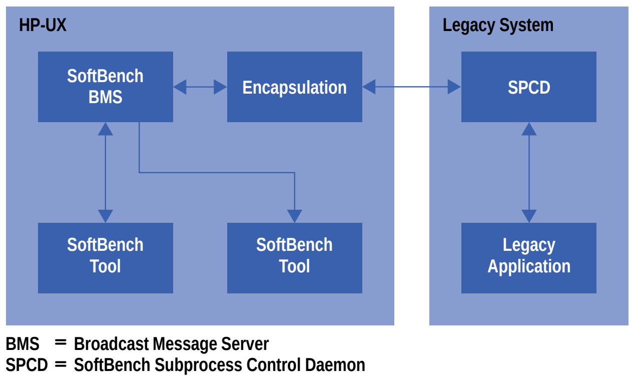

Fig. 2. *The organization of software components after SoftBench is set up. The SPCD has been ported to a legacy system, and an encapsulation has been written for each legacy application to be integrated in the SoftBench environment.*

Because the encapsulator library has only been ported to the HP-UX and Solaris operating systems, encapsulations that link with encapsulator routines must run on a machine using HP-UX or Solaris. While the encapsulator library could be ported to other operating systems, this is usually unnecessary since an encapsulation can use the SPCD to execute a subprocess on a remote host as easily as on a local host. This is one of the major advantages gained by porting the SPCD to all operating systems in the environment.

A few other limiting factors must be taken into account when writing encapsulations. First, it is difficult, if not impossible, to integrate applications that have no command-line interface. For example, if the only way to interact with an application is through a graphical user interface, then an encapsulation of that application must emulate mouse movements and button clicks to communicate with it. This is generally not a feasible option.

Another factor to consider when writing encapsulations is the granularity of the information provided by the application to be encapsulated. If the application does not give some sort of notification for each action that it takes, then the encapsulation will be limited in its interpretation of what the application is doing. For more information about the limitations of the encapsulator library see **reference 1**.

Once the necessary encapsulations have been written, the next step in integrating an application into a heterogeneous computing environment is to extend the SoftBench environment so that all of the desired applications are seamlessly integrated into it. This is accomplished by modifying the SoftBench configuration file softinit to include references to each of the new encapsulations (see Fig. 3). This action informs SoftBench about the new functionality that is now available through the encapsulations and how to access those encapsulations.

Modifications to softinit can also be used to inform SoftBench to replace existing tools with new encapsulations. For instance, the standard e-mail tool that comes with SoftBench could be replaced with an encapsulation of a local e-mail application. SAIC has used this capability to replace the debugger that comes with SoftBench with an encapsulation of the GNU debugger, gdb. This provides SAIC's customers with a fully integrated debugger that runs on any machine that gdb supports, which includes most modern operating systems.

To further integrate a development environment, the SoftBench message connector can be used to automate repetitive tasks and enforce software development processes. The message connector works by monitoring the BMS for a desired message and then executing a user-supplied routine whenever that message is seen. For example, suppose company policy requires that a complexity analysis program be run on all source code when it is checked into the configuration manager. To meet that requirement with no human intervention, the message connector could be configured to monitor the BMS for a message from the configuration management tool indicating that a file has just been checked in. Then, it would run the analysis program on that file, perhaps e-mailing the results back to the developer who checked in the file.

For software development processes that require more intricate interactions than the message connector can provide, SAIC's SynerVision product can be used. It provides a nextgeneration process management environment that helps teams manage the software engineering process, including such tasks as writing new software, debugging programs, maintaining existing systems, and porting to new platforms. Also, because SynerVision fully supports the SoftBench environment, no new encapsulations need to be written for it.

Note that the steps described above for integrating a heterogeneous software development environment with SoftBench do not need to be implemented all at once. Instead, the built-in extensibility of SoftBench allows one to take a progressive approach wherein applications are encapsulated one at a time and added to the environment as they are completed. Such an approach can smooth the migration path from a legacy system to an open system by eliminating the need for a complete switchover to the new technology.

```
#
# $HOME/.softinit –– user customizations to
# SoftBench initialization

#
# Editor

#
# To use ”vi” as the editor, uncomment the
# following line
EDIT   TOOL  NET   *  %Local% softvisrv –scope
net -types %Types%

# To use ”softedit” as the editor, uncomment the
#following line
#EDIT  TOOL  NET   * %Local% softeditsrv –scope
net –types %Types%

# To use ”emacs” as the editor, uncomment the
#following line
#EDIT TOOL NET     * %Local% emacs

#
# Configuration Management
#

DM   TOOL  DIR     * teflon softdm –host %Host%
–dir %Directory% –file %File%

# To use ”RCS” as the CM tool, uncomment the
# following line
CM   TOOL NET * teflon softrcs –scope net

# To use ”SCCS” as the CM tool, uncomment the
# following line
#CM  TOOL NET * spike softsccs

#
# Debugger
#
# To use GDB as the debugger, uncomment the
# following line
#DEBUG TOOL FILE * teflon /usr3/stevew/
#DebugBackends/softgdb/softgdb –d 255 –1 /tmp/
softgdb.log –host %Host% –dir %Directory% –file
%File%

#
# Tandem stuff
#

# To startup RSHELLSRV in debugging mode,
#  uncomment the following line
#RSHELLSRV TOOL HOST * teflon /usr/rshellsrv/
#rshellsrv –d 255
1 /tmp/log.rshellsrv.%Host%.$USER –host %Host%

# To startup RSHELLSRV in standard mode,
#uncomment the following line
RSHELLSRV TOOL HOST * teflon /usr/rshellsrv/
rshellsrv –host %Host%
```

Fig. 3. *A SoftBench configuration file* softinit*. This file contains references to each new encapsulation.*

Benefits of Integration with SoftBench

By extending SoftBench as described above, a heterogeneous collection of computing systems with disparate, incompatible tools can be transformed into an efficient, tightly coupled software development environment with consistent, easy-to-use graphical user interfaces across all machines. Fig. 4 shows the result of an example integration.

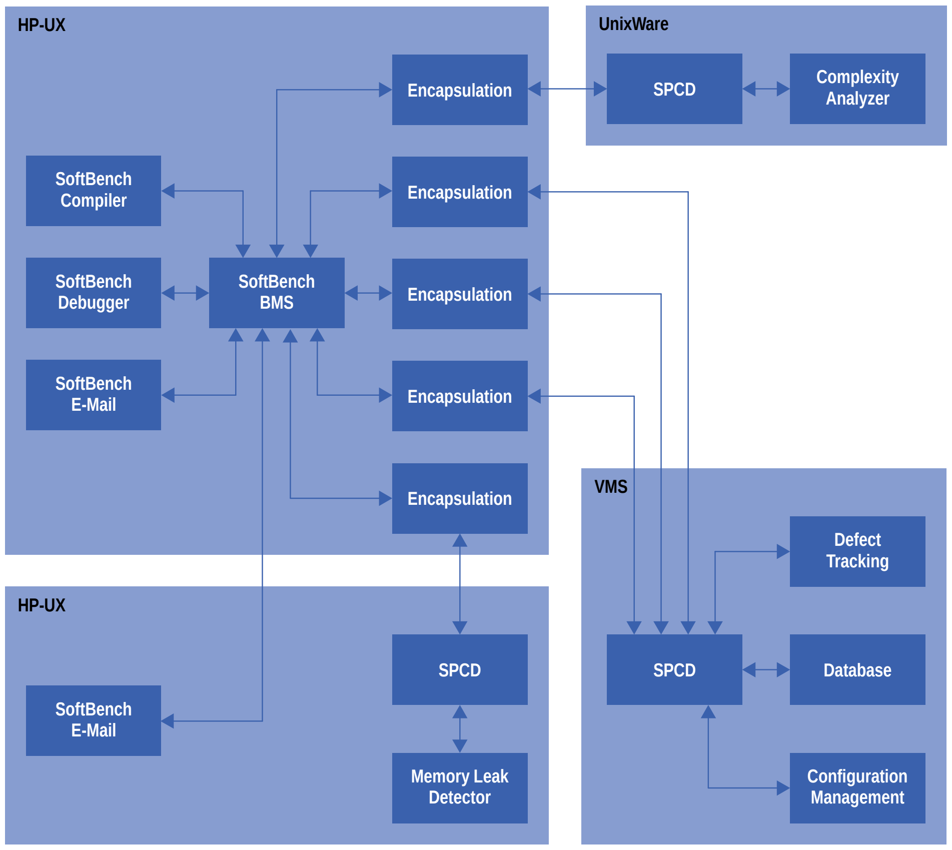

Fig. 4. *An example of an integration over several platforms. Note that there is an encapsulation for each application.*

Certainly, one of the biggest advantages of integrating with SoftBench is the realization of a standard, consistent user interface to all tools on all machines. This consistent interface minimizes the learning curve for developers by reducing the number of commands that they need to learn to use the environment. It also improves the efficiency of developers by simplifying their interactions with both applications and operating systems and by providing a means for data sharing between applications (e.g., cut and paste, drag and drop, etc.). In environments with legacy systems where developers have been using text-based terminals, the benefit of this graphical user interface can be enormous.

As discussed earlier, all SoftBench applications use the X Window System to display their graphical user interfaces. This provides the advantage that the complete software development environment is always available from any machine that has an X server. Furthermore, the environment looks and works exactly the same no matter what machine a developer uses, from a Macintosh PowerBook laptop running an X server to an HP 9000 workstation running HP VUE.

An environment integrated with SoftBench also provides the advantage that remote data access is transparent to the user. By using NFS and the automounter, SoftBench automatically retrieves data from remote machines without any user intervention. Developers only need to specify which machine contains the desired data, and SoftBench handles the rest. This benefits developers because they do not have to copy files back and forth between machines or know the intricacies of networked file systems.

Similarly, SoftBench provides the advantage that remote program execution is transparent to the user. By using SPCD, SoftBench can execute applications on remote machines without developer intervention. Developers no longer need to log into various machines to run the tools they need because SoftBench provides a centralized control center that places all tools at their fingertips. This lets the developer concentrate on the task at hand instead of worrying about logins, passwords, pathnames, and so on.

By providing transparent access to both data and applications, SoftBench allows resources to be spread across a distributed client/server topology without introducing complexity into its use. Developers get a unified view of their environment whether it contains one machine or one hundred, whether all their data is centralized on one server or distributed across many systems. In addition, machines can be added to (or removed from) the environment without impacting developers simply by modifying SoftBench to use (or stop using) the given machines.

Another advantage of integrating with SoftBench is the rich set of state-of-the-art software development tools that come with SoftBench. These tools benefit developers by simplifying and expediting the edit-compile-debug cycle. The tools automate processes such as checking source files into and out of configuration management, building executables, and displaying errors found by the compiler. In addition, the tools can provide a graphical view of source code, allowing a developer to quickly learn unfamiliar code or find errors in program flow.

Furthermore, by encapsulating local and third-party applications in the environment, developers will have access to those applications as easily as if they were standard SoftBench tools. This benefits developers because they do not have to know on which host the applications exist or how to start them. Instead, the developer can start an application simply by selecting it from the list of applications in the SoftBench tool manager. In fact, by customizing the environment with the message connector, many applications can be started automatically.

As discussed earlier, the message connector and SynerVision can save developers time and effort by automating repetitive tasks and by enforcing software development policies such as ensuring that required tasks always occur and that those tasks are executed in the proper order. By enforcing well-defined policies, SoftBench can help increase the efficiency of the software development process and improve the quality of the finished product.

Conclusion

Software development in a heterogeneous computing environment can be a difficult proposition. Varying hardware and software platforms, incompatible tools, and inconsistent user interfaces are just a few of the trouble spots. However, Hewlett-Packard's SoftBench product can be used to solve these problems by providing a standard upon which to integrate the disparate components of such an environment. By porting SoftBench's SPCD to each operating system involved, all machines become equally and consistently accessible from SoftBench. Then, by encapsulating the applications on those systems, the applications become fully integrated SoftBench tools capable of interacting with other SoftBench tools.

Acknowledgements

SAIC's success in deploying SoftBench to solve its customers' problems is the result of a collaboration of exceptional talent at both SAIC and Hewlett-Packard. I would especially like to thank Vern Badham, Winn Rindfleisch, and Dave Romaine for their invaluable contributions to these projects and for their help in writing this article. I would also like to thank Curt Smith for inviting me to join his team at SAIC and John Dobyns for allowing me to continue my encapsulation work after I left SAIC. Lastly, I extend special thanks to Dick Demaine and the Software Engineering Systems Division of Hewlett-Packard for making SoftBench possible and for their continuing support of SAIC's work.

References

1. B.D. Fromme, "HP Encapsulator: Bridging the Generation Gap," *Hewlett-Packard Journal,* Vol. 41, no. 3, June 1990, pp. 59-68.
2. C. Gerety, "A New Generation of Software Development Tools," *Hewlett-Packard Journal*, Vol. 41, no. 3, June 1990, pp. 48-58.
3. J.J. Courant, "SoftBench Message Connector: Customizing Software Development Tool Interactions," *Hewlett-Packard Journal,* Vol. 45, no. 3, June 1994, pp. 34-39.

HP-UX 9.* and 10.0 for HP 9000 Series 700 and 800 computers are X/Open Company UNIX 93 branded products.

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open is a registered trademark and the X device is a trademark of X/Open Company Limited in the UK and other countries.

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation.

▶ Go to Next Article

▶ Go to Journal Home Page