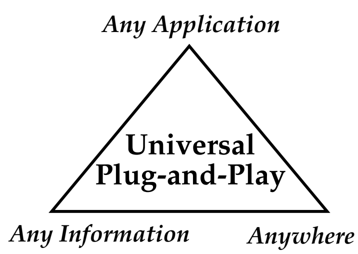
# *Introduction to the ToolTalk Service*

## *Executive Summary*




As computer users increasingly demand that independently developed applications work together, “interoperability” and “integration” are becoming important themes for software developers. These terms convey the idea of inter-application cooperation; of applications achieving more by working together than by working separately. The ToolTalk™ service, from SunSoft™, a network-spanning inter-application message system, has been designed to promote cooperation among independently developed applications.

Use the ToolTalk service when you want *universal plug-and-play* capability. The term *universal plug-and-play* means that any tool can be replaced by any other tool that follows the same protocol. That is, any tool that follows a given ToolTalk protocol can be placed (plugged) into your computing environment and can perform (play) those functions indicated by the protocol. Tools can be mixed and matched, without modification and without having any specific built-in knowledge of one another. Universal plug-and-play is built on the concept of describing applications by the functions the addressee needs to perform, rather than calling a specific type of application.

The ToolTalk service is completely distributed and network aware. Location issues are entirely transparent to a user. While it provides easy global access, ToolTalk hides all networking details. The ToolTalk service can function as a partner to Open Windows or can operate independently of any windowing system. In addition to this, ToolTalk messages can be retrofitted to any existing application.

The ToolTalk service is a standard feature of the Solaris 1.0 and 2.0 operating systems software platforms. All Solaris users have ToolTalk installed and available on their systems.

ToolTalk messaging is an API of the DOE Application Framework. If you adopt ToolTalk message compliant applications today, you can be assured of having made an excellent investment. You will be able to continue to use ToolTalk messaging in your existing applications with the DOE Application Framework as you enhance your applications to take advantage of DOE distributed object facilities.

## *Universal Plug-and-Play: The Application Integration Requirement*

Today, computer users have access to a wide variety of applications that are either purchased from software suppliers or developed in-house. Not only do users want to use applications they are already familiar with, they want their tools of choice to be integrated with other applications automatically. Typically, a company limits the choices available to its workers by standardizing on one or more applications of a particular type. Unfortunately, this results in little integration among applications. This is not the ideal solution. Without the best, highly integrated tools, an organization does not have an edge in the global market.

Users want any and all applications to integrate automatically and consistently. They need automated access to all data types including numerical, text, graphical, audio, video, and control information. And, they should not be required to know where the information is stored. This is the *universal plug-and-play* requirement of applications, a theme depicted in Figure 1.

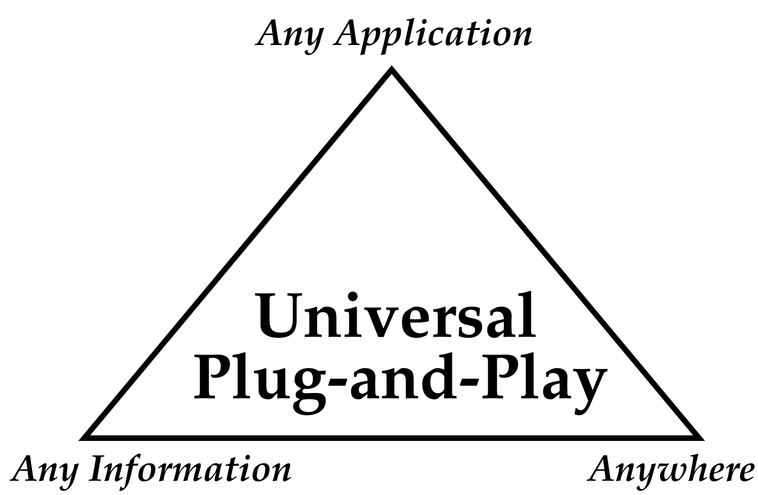



*Figure 1* Universal Plug-and-Play: The Application Integration Requirement

ToolTalk meets the universal plug-and-play integration requirement. It:

- Supports message passing between any applications
- Accesses any information types
- Integrates workgroups and provides global access while hiding the networking details
- Makes messaging entirely transparent to users
- Provides high performance and throughput

Applications that use the ToolTalk service cooperate by sending and receiving messages. The messages passed by cooperating applications form a small, well-defined message protocol that maximizes both application integration and autonomy. Many applications behave as though they are just a single application.

ToolTalk supports procedural and object-oriented messaging. It leverages the functionality of other applications and fits easily into the mix of procedural applications that dominate current usage.

If message protocols are observed, cooperating applications can be modified — even rewritten — without affecting one another.

Some advantages of the inter-application communication provided by the ToolTalk service are:

- Applications can do the tasks they do best and ask other applications to perform tasks for which they have little or no functionality.
- Inter-application communication is a small investment with the ToolTalk service. Existing applications are easily modified to send, receive, and use shared information.
- No knowledge about the application receiving the information is needed — including its location on the network. The ToolTalk service determines who receives the information by looking for applications who asked for the type of information being sent.
- Role addressing enables addressing to be generic. It describes the functions the addressee needs to perform, such as editing, instead of calling a specific named application. The appropriate application is then addressed.

## *The ToolTalk Service*

The ToolTalk service is one of the first commercial products in the new class of mechanisms designed to facilitate information exchange between applications from different developers. These mechanisms play a key role in application interoperability. Figure 2 summarizes how the ToolTalk service fits into a networked environment.

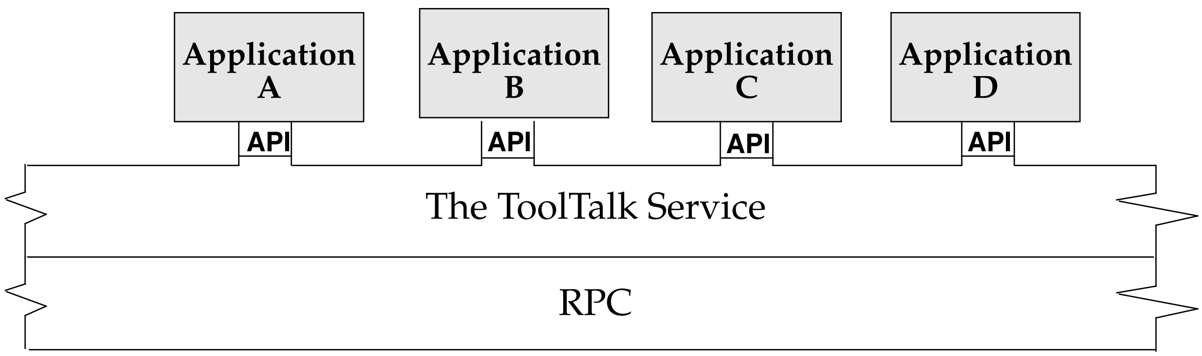



*Figure 2* Application Interoperability Facilities

The ToolTalk service is used in two ways:

- Applications use the ToolTalk service directly, calling functions from the ToolTalk API library to create, send, and receive messages. Examples of these message include messages to open, display, or close application windows.
- Applications could use a “service” built on top of the ToolTalk service. These application services would then use the ToolTalk service as a communication backbone. These types of services can provide linking and other capabilities for application users.

Both methods provide a unique opportunity for applications to bridge the communication gap and provide cross-application functionality to users.

The ToolTalk service runs on top of the ONC+/TI-RPC product.

## *Benefits of the ToolTalk Service*

Some highlights of the ToolTalk service capabilities are:

- Message addressing is very flexible. A message can be addressed to a specific process, to a particular type of process, or to a process that can perform a particular operation.
- Standard message sets allow developers to communicate between independently developed applications.
- Applications can send notice messages that announce events and request messages that ask for operations to be performed. Multiple processes may receive a copy of either kind of message for observation, but only one process executes a request.
- An application can register interest in receiving messages even though no process is running the application. It can further direct the ToolTalk service to start a process when a message is available, or to save the message until a process running the program is started in another way.

### *Benefits to Application Developers*

The ToolTalk service provides solutions meeting the requirements of multicast messaging primarily for developers of procedural-based, self-contained applications. ToolTalk takes independent applications that solve only portions of the end-user's problem and allows them to interoperate to solve whole problems. Instead of requiring developers to create an entire set of applications that solves a problem, developers can enhance and integrate existing applications to perform the functions needed to solve end users' problems.

By using message sets, little knowledge of other applications is needed to communicate through the ToolTalk service, and developers can concentrate on applications that "plug and play" with other vendor applications. This is particularly important because it means companies do not have to cultivate relationships with every vendor they need to communicate with, nor do they have to change their code every time a new application is added to the set of applications that interoperate.

## Benefits to End Users

As applications work together, communicating about events that have occurred or operations that need to be performed, the user's workflow becomes automated. Instead of performing one task with one application and then inputting the results into another application to perform the next task, users focus on solving an entire problem.

When applications communicate, end users gain greater freedom in choosing the application that best suits their needs. Today, a single company can provide a set of applications that may or may not interact. Users choose sets of applications based on trade-offs of the entire set of applications, not on the basis of whether the individual application meets the unique needs for the portion of the problem the application solves. As users gain the freedom to choose applications that interoperate, they can tailor their environment for their unique work situation.

## Using the ToolTalk Service

To communicate using the ToolTalk service, applications create, send, and receive ToolTalk messages. Sending applications create, fill in, and send a message; the ToolTalk service determines the recipient and delivers the message to the receiving application.

ToolTalk messages are simple structures that contain fields for address, subject, and delivery information. To send a ToolTalk message, an application obtains an empty message, fills in the message attributes, and sends the message. The sending application needs to provide the following information:

- Is the message a notice or a request? (that is, should the recipient respond to the message?)
- What interest does the recipient share with the sender? (for example, is the recipient running in a specific user session or interested in a specific file?)
- What operation should be performed or has been performed? (that is, are there arguments needed to perform the operation?)

To narrow the focus of the message delivery, the sending application can provide more information in the message.

An important ToolTalk feature is that senders need to know little about the recipients because applications that want to receive messages explicitly state how these messages should appear. This information is registered with the ToolTalk service in the form of *message patterns*.

Message patterns usually look very similar to the message protocols that applications have agreed to use. An application may add more types of messages they wish to receive beyond the list in the message protocol; for example, an application may want to observe all messages about a particular file.

Applications using the ToolTalk service to communicate usually have something in common — the applications are running in the same user session or they are interested in the same file or data. Applications register this interest in sessions or files, or both, with the ToolTalk service. This file and session information is used in conjunction with the message patterns by the ToolTalk service to determine who should receive a message.

For example, one application wants to inform all interested applications running in a user's session that an event has occurred. Because network window systems, such as the OpenWindows system, allow users to run applications on multiple machines, it is difficult for the sending application to know which applications should be notified and where they are running. Using the ToolTalk library functions, the sending application sends a message scoped to the session and the ToolTalk communication process delivers copies of the message to all applications in the session that have registered a pattern that matches the message, as illustrated in Figure 3.

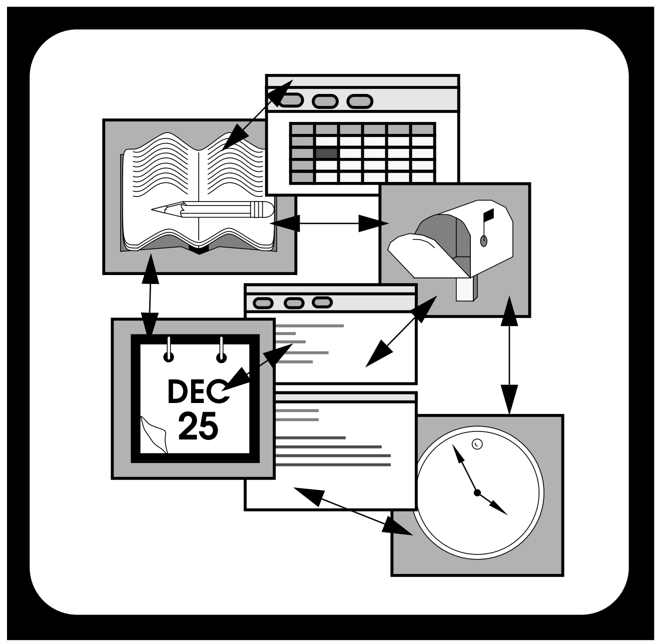



*Figure 3* ToolTalk Single User "Session-Scoped" Messaging Between Applications

In file scoping, the candidates for pattern matching are limited to those interested in a particular piece of information that is represented by a file. This type of scoping can integrate many applications that run on behalf of different users where the applications are working on the same information. Figure 4 shows file scoping.

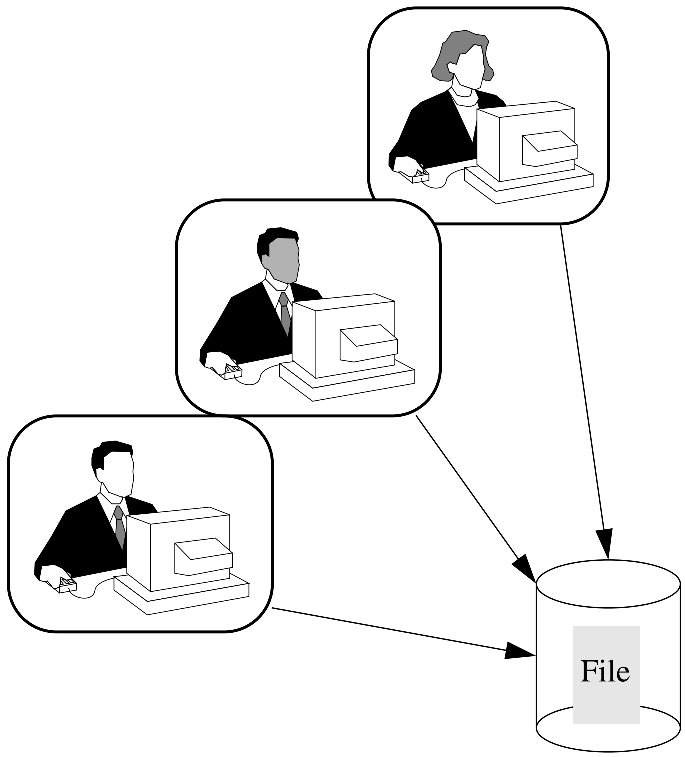



*Figure 4* ToolTalk Workgroup “File-Scoped” Messaging Between Applications

## *ToolTalk Messages*

Processes use the ToolTalk service to send each other messages. Each message consists of attributes such as those shown in Table 1. Within a message's scope, the ToolTalk service determines the actual receivers by matching attributes.

*Table 1* Basic Message Attributes

| **Attribute** | **Value** |
|---|---|
| Address | procedure or process |
| Class | request or notice |
| Operation | action to be performed, or event that has occurred |
| Arguments | input values for operation, output results |
| Scope | session, file, both, file-in-session |
| State | created, sent, handled |

Sending processes supply values for message attributes. Receiving processes specify, in the message patterns they register, the attribute values of messages they want to receive. The ToolTalk service compares message attributes to receiver patterns, and delivers copies of a message to the processes that have registered matching patterns.

### *Notices and requests*

Processes can send two classes of ToolTalk messages: notices and requests. A *notice* is informational, a way for a process to announce an event. Processes that receive a notice need not return any results to the sender. A *request* is a call for an action with the results of the action recorded in the message, and the message is returned to the sender as a reply.

### *Handlers and observers*

For each pattern describing a message it wants to receive, a process also declares whether it can handle the message or only wants to observe it. Both requests and notices can be handled or observed. Although many processes can observe a message, only one process can handle it; thus, a requested operation is performed only once. If the ToolTalk service cannot find a handler for a request message, it returns the message to its sender with a state of failed. Because notices do not return results, an unhandled notice is not an error.

### *Routing*

A notice takes a one-way trip, as shown in Figure 5. The sender creates a message, fills in attribute values, and sends it. The ToolTalk service matches message and pattern attribute values, and gives a copy of the message to one handler and to all matching observers. File-scoped messages are automatically transferred across session boundaries to processes that have declared interest in the file.

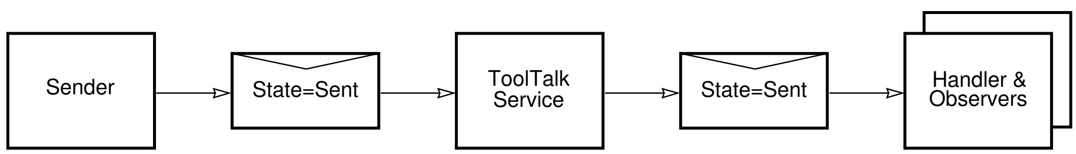



*Figure 5* Notice Routing

A request, as shown in Figure 6, takes a round-trip from sender to handler and back to sender: copies of the message take one-way side trips to observers. The ToolTalk service delivers a request to only one handler. The handler adds results to the message and sends it back. Other processes can observe a request before it is handled, after it has been handled, or at both times; observers absorb a request without sending it back.

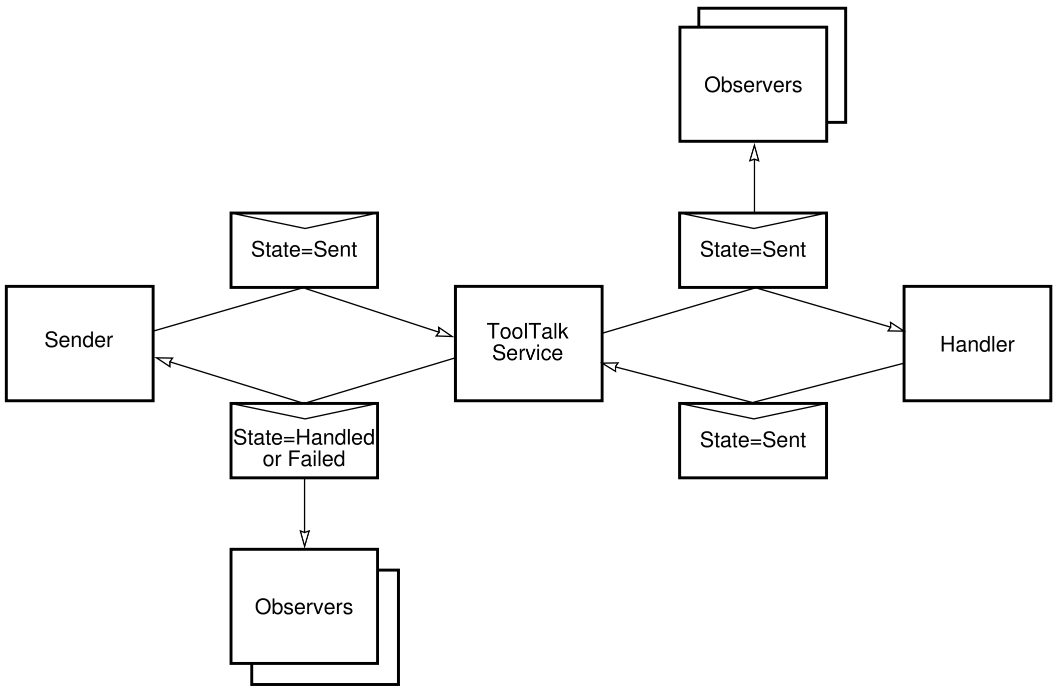



*Figure 6* Request Routing

## *Addressing*

Although messages are ultimately delivered to processes, they can be addressed to processes or procedures. Each ToolTalk process that receives messages has a process identifier (*procid*) that is unique across all running sessions. A sender that knows a receiver's procid can send a message to that process.

However, one process does not usually know another process's procid; more often a sender does not care which process executes an operation (request message) or learns of an event (notice message). A procedure message is received by any process that has registered a matching pattern. The sender specifies an operation and arguments; and lets the ToolTalk service finds one handler and all matching observers.

## *Standard Message Sets*

Applications usually are not developed concurrently. And, typically they are developed by multiple groups. Users often want to substitute their favorite applications, so they want the universal plug-and-play across all applications. ToolTalk's standard message sets provide integration benefits to users of compliant applications.

By having messages organized into discrete sets, you get a pre-arranged interface between applications with common interests. These sets are specific to an application domain; for example, accounting or computer aided software engineering. Figure 7 represents the relationships among the message sets.

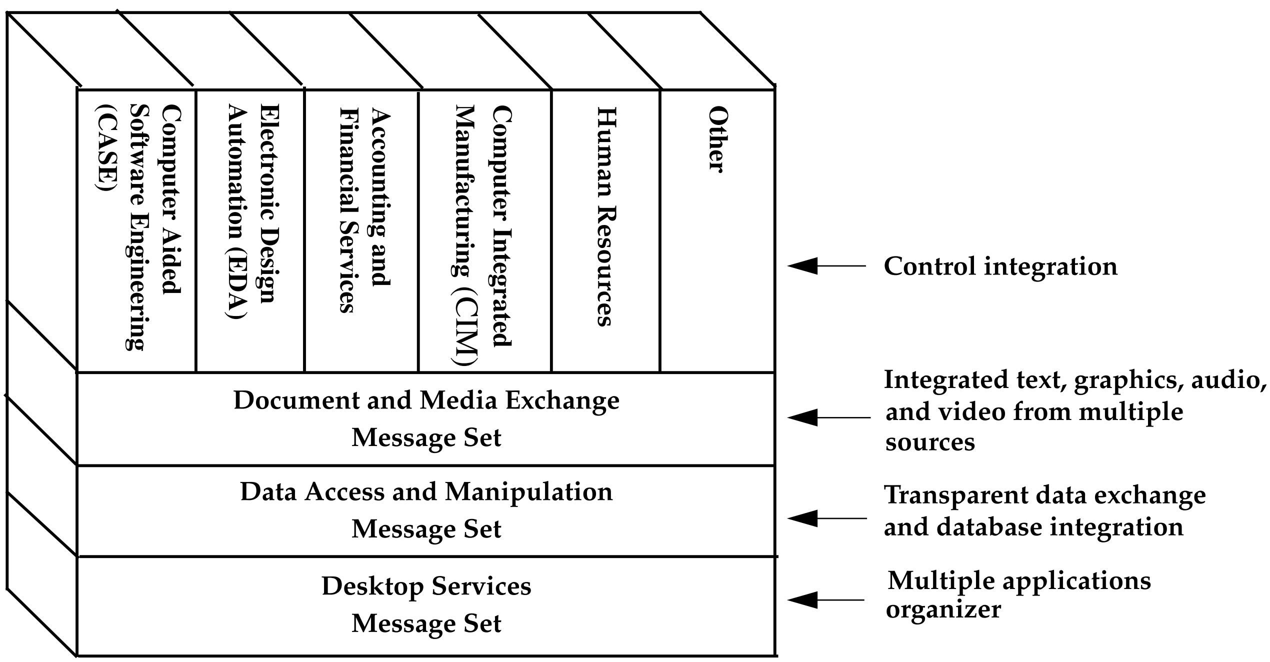



*Figure 7* Standard ToolTalk Message Sets and Integration Benefit Examples

As the result of extensive work to define message standards with leading software suppliers and end users, SunSoft provides these well-defined, highly functional, standard message sets to anyone. They are an open specifications in the public domain without royalty or license fees.

ToolTalk compliant applications realize many integration benefits from the various message sets, which are described below.

## *Desktop Services Message Set*

To achieve basic desktop integration, applications need to support a basic set of messages to enable inter-application control. With the Desktop Services Message Set, an application can manage the state of files, windowing, and processing for itself or any other application. A powerful messaging protocol that benefits both developers and users of desktop applications, the Desktop Services Message Set allows applications to interact easily with other desktop applications.

For integrated control of applications to take place, certain basic features are needed to launch, halt, control display appearance, and pass information regarding input and output data. All applications need to have these facilities so that other applications in the toolset can inter-change basic control information. This kind of functionality enables the development of smart desktops and integrated smart toolsets. Groups of applications can now call upon one another to perform tasks and to interact as one solution environment for the end user.

## *Data Access and Manipulation Message Set*

Users who access databases and manipulate information in spreadsheets and decision support applications need transparent integration. The Data Access and Manipulation Message Set makes data integration seamless and transparent to users.

## *Document and Media Exchange Message Set*

No single vendor provides a completely integrated solution that meets the complex needs of today's market. The ToolTalk Document and Media Exchange Message Set allows any application to incorporate text, graphics, audio, and video generated by another application. Virtually all applications can benefit from close integration of documentation and multimedia.

Using ToolTalk Document and Media Exchange Message Set, multimedia applications can communicate with one another in a transparent manner, both locally and over networks, regardless of data formats, compression technology, and other technical issues all of which have previously confined the use of this technology.

## CASE Interoperability Message Sets

To achieve control flow integration in an integrated CASE environment, applications need to support messages that enable inter-application control. The CASE Interoperability Message Sets are the common message sets that provide this functionality for CASE applications. A powerful messaging definition that benefits both developers and users of CASE applications, the CASE interoperability Message Sets allow applications to interact easily with other CASE applications. The CASE Interoperability Message Sets define a standard set of CASE functionality available through control integration services. Each message set describes an area of functionality in the CASE environment. The message sets are:

- Analysis and Design
- Browsing
- Building
- Debugging
- Editing
- Version Management

## Adopting Message Sets

It is easy to adopt ToolTalk messages. With help from a series of technical white papers, an experienced programmer can learn how to incorporate them into new or existing applications.

If you are a software vendor, you are not obligated to adopt all message sets. You can use some or all, depending on the functional integration needs of your customers. Your applications can then interoperate with any other application with the same message set(s).

## *Moving Toward Distributed Objects*

ToolTalk is a part of SunSoft's Project DOE. Through Project DOE, SunSoft has a major strategic commitment to deliver the power of the distributed object paradigm with an evolutionary growth path for current applications. ToolTalk can be used today, and ToolTalk message compliant applications will evolve smoothly into the future.

Typically, the procedural and object applications of today group together large collections of functionality and are known as "large grained applications." See Figure 8. The ToolTalk service integrates these applications through multicast messaging.

As SunSoft delivers to market the Project DOE system facilities that enable distributed objects, developers will be able to create new kinds of applications based on the distributed object model. This finer grained application construction model will allow the "plug-and-play" construction of applications from distributed object components.

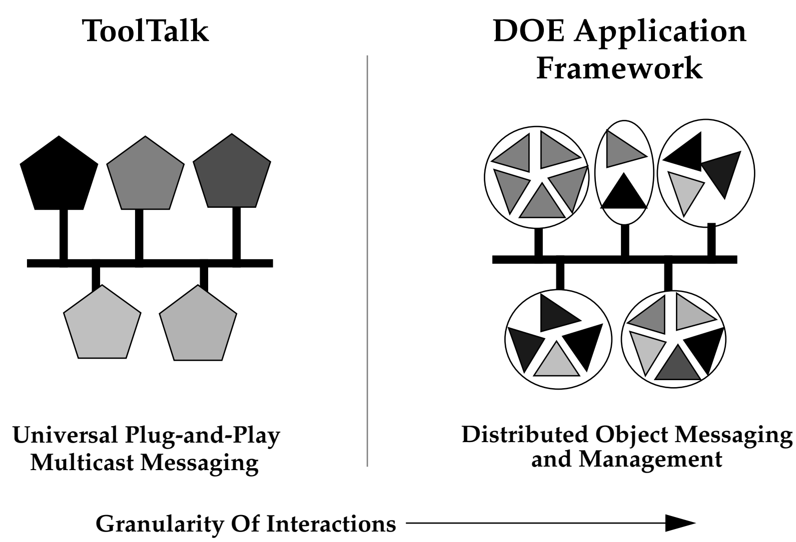



*Figure 8* ToolTalk's Large Grained Message Integration Leads to Finer Grained Distributed Object Integration

SunSoft will move the ToolTalk messaging API into the Application Framework and will support all current ToolTalk functionality. Figure 9 shows how an API migration to DOE protects your investment in ToolTalk messages.

From a ToolTalk functionality perspective, this move will be completely transparent to users and developers. Over time, the underlying ToolTalk implementation that users and developers do not see will be swapped out and replaced by the Distributed Object Management Facility (or DOMF) and certain common services.

This is great for developers and users because it preserves current application functionality while providing access to distributed solutions in the future. Investments made today to integrate existing large grained applications via ToolTalk will pay off today through increased user productivity. These investments will also prepare you for your evolution into distributed object applications.

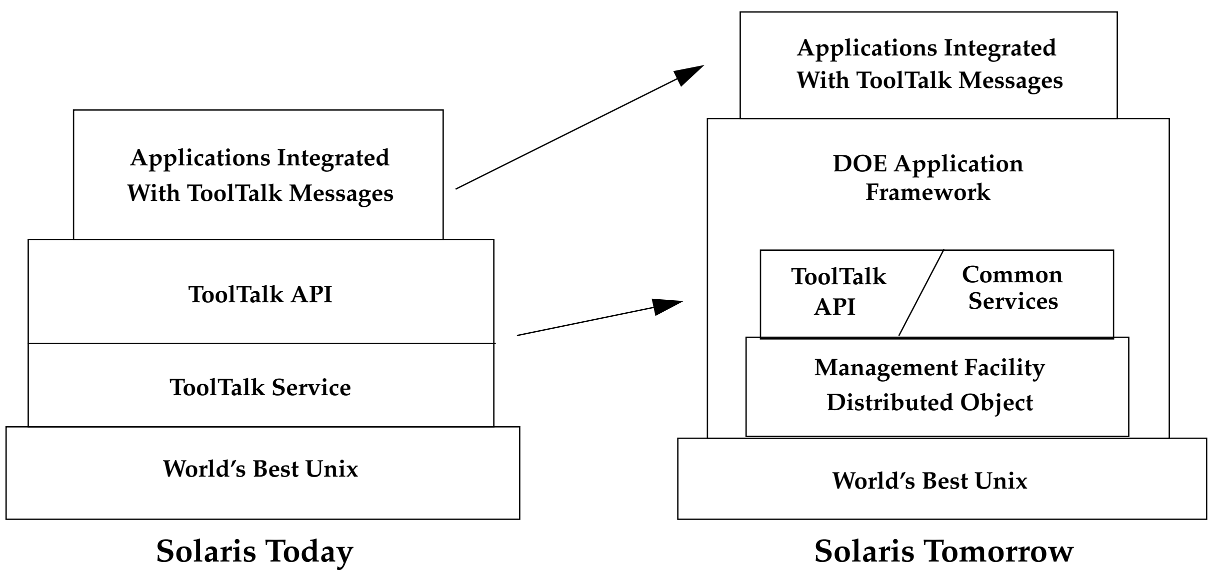



*Figure 9* API Migration to DOE

## *Summary*

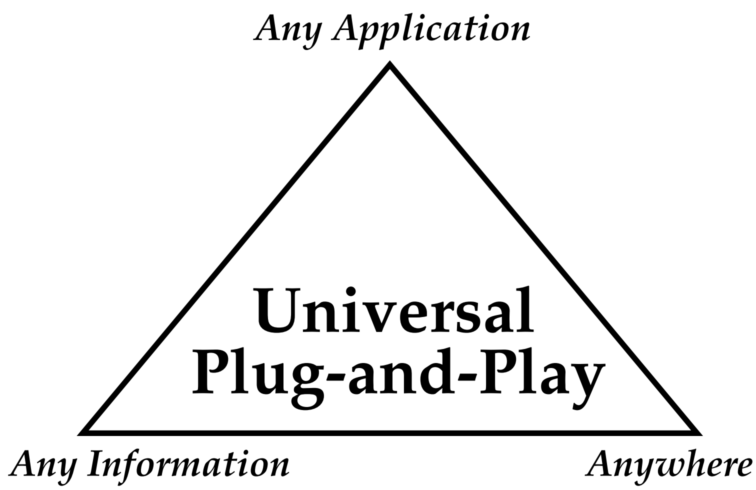
Today there is a need for applications to integrate automatically with other applications. By cooperatively using one another's facilities, interoperating applications offer users capabilities that would be difficult to provide in a single application. The ToolTalk service is designed to facilitate the development of interoperating applications that serve individuals and workgroups.

The ToolTalk service is the solution to the integration problem. It is enabling technology that provides the following:

- It supports the *universal plug-and-play* requirements of today's computer users. They gain greater freedom in choosing applications that interoperate with one another.
- It provides multicast messaging solutions to meet the integration requirements of developers of procedural-based, self-contained applications.
- Its standard message sets provide a pre-arranged interface between applications with common interests.
- It includes a transition path from today's procedure-oriented applications to distributed object applications of the future.
- It is a standard feature of the Solaris 1.0 and Solaris 2.0 operating environments. It is available at no extra cost.

# *Introduction to the ToolTalk Service*

*A White Paper*



# *Table of Contents*

SunSoft, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

For more information, call 1 800 227-9227.

Printed in USA 9/91 91031-0 1.5K