HP Distributed Smalltalk: A Tool for Developing Distributed Applications

An easy-to-use object-oriented development environment is provided that facilitates the rapid development and deployment of multiuser, enterprise-wide distributed applications.

by Eileen Keremitsis and Ian J. Fuller

HP Distributed Smalltalk is an integrated set of frameworks that provides an advanced object-oriented environment for rapid development and deployment of multiuser, enterprise-wide distributed applications. Introduced in early 1993, and now in its fourth major release, HP Distributed Smalltalk leverages the ParcPlace Smalltalk language and the VisualWorks development environment. Together, HP Distributed Smalltalk and VisualWorks enable rapid prototyping, development, and deployment of CORBA-compliant applications.†

In the global marketplace, corporate information technology needs are increasingly demanding because worldwide competition requires geographically dispersed operations, changing markets require agility to remain competitive, pressure to improve return on investment requires strong cost controls, timely access to complete information is crucial for business success, and finally, corporate users require access to both legacy and newly developed information sources and applications.

HP Distributed Smalltalk helps answer these business needs by supporting:

- Easy on-demand access to information and services across the enterprise
- Dynamic interaction of distributed people and resources
- Greater application flexibility and ease of use
- Insulation from differences in operating environments
- An architecture that supports an evolutionary approach including legacy system integration
- Industry standards that will allow application interoperability across languages, high productivity, and code reuse.

Customers can take advantage of HP Distributed Smalltalk's easy-to-use development environment to create distributed solutions to compete effectively in the global marketplace. For example, with HP Distributed Smalltalk, customers might build on the sample Forum application (described later) so that their geographically dispersed users can simultaneously annotate a shared document. Also, customers might use HP Distributed Smalltalk to create three-tiered database access applications that extend the advantages of existing client-server architectures for better isolation between user interfaces, data manipulation models, and legacy and new data.

† CORBA, or Common Object Request Broker Architecture, defines a mechanism that enables objects to make and receive requests and responses. HP Distributed Smalltalk's implementation of this architecture is described later in this article.

Three-tiered applications are the most efficient and scalable form of software design for building complex applications. They carefully separate the user interface (tier one) from the business rules governing the application (tier two) and the persistent storage for the information in a database (tier three). Each tier can reside on a different machine in a network, making best use of the network resources. HP Distributed Smalltalk contains objects that enable the straightforward construction of these applications.

Using HP Distributed Smalltalk

An application written in HP Distributed Smalltalk is able to respond to service requests from remote systems. Remote entities that request services of an application do not have to be written in HP Distributed Smalltalk as long as they are in a system that implements the standard ORB (object request broker) and common object services from the Object Management Group (OMG). See "Object Management Group" on page 86 for a description of these items.

In many cases an HP Distributed Smalltalk application's component objects are distributed across several systems. These distributed objects can interact seamlessly so that end users are unaware of where the objects are located.

An overview of the process of running an HP Distributed Smalltalk application is shown in Fig. 1. For incoming requests to the service provider, the ORB translates requests from the implementation-neutral Interface Definition Language (IDL) to the local language (ParcPlace Smalltalk) and forwards them to the correct local object for processing. To complete the request, the service provider's ORB takes return values, translates them to IDL and forwards them to the remote ORB from which the request was received.

Not only does HP Distributed Smalltalk support distributed application delivery but it also provides an environment for distributed application development, which includes:

- A complete implementation of the Object Management Group's latest standards
- A rich suite of tools for application development and administration including simulated remote test support, a remote debugger, and an IDL interface browser and generator
- A user interface environment and sample applications that developers can reuse or extend, or simply use to become familiar with the system.

Object Management Group

The Object Management Group, or OMG, is a nonprofit international corporation made up of a team of dedicated computer industry professionals from different corporations working on the development of industry guidelines and object management specifications to provide a common framework for distributed application development.

OMG publishes industry guidelines for commercially available object-oriented systems, focusing on areas of remote object network access, encapsulation of existing applications, and object database interfaces. By encouraging industrywide adoption of these guidelines, OMG fosters the development of software tools that support open architecture, enabling multivendor systems to work together.

To define the framework for fulfilling its mission, in 1992 OMG published its Object Management Architecture Guide. This guide provides a foundation for the development of detailed interfaces that will connect to the elemental components of the architecture. Fig. 1 shows the four main components of this architecture:

- The object request broker (ORB) enables objects to make and receive requests and responses in a distributed object-oriented environment.
- Object services is a collection of services with object interfaces that provide basic functions for creating and maintaining objects.
- Common facilities is a collection of classes and objects that provide general-purpose capabilities useful in many applications.
- Application objects are specific to particular end-user applications.

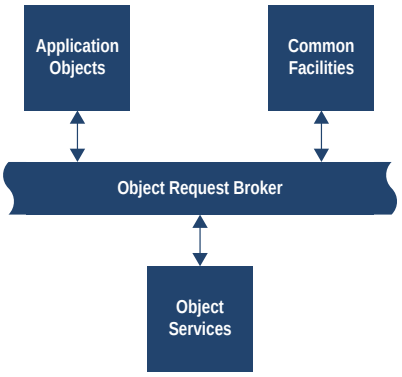

Fig. 1. The object management architecture.

The application objects, object services, and common facilities represent groupings of objects that can send and receive messages. The software components in each of these primary components have application programming interfaces that permit their participation in any computing environment that is based on an object technology framework.

In addition, because HP Distributed Smalltalk is an extension of VisualWorks, developers are able to do their programming in a language they already know (ParcPlace Smalltalk) using the VisualWorks application builder.

VisualWorks is an implementation of the Smalltalk programming language and environment. It provides an excellent environment for building standalone and simple client/server applications that are 100% portable between many of the major computing platforms and operating systems. HP saw an opportunity to enhance the capabilities of VisualWorks to be the basis for next-generation applications by adding objects that enable VisualWorks systems to communicate directly using a standardized set of communications facilities.

Framework

The HP Distributed Smalltalk framework is an environment that encompasses everything from communication with other systems through database access to the object-oriented ParcPlace Smalltalk language and a rich suite of developer's tools, all seamlessly integrated to facilitate distributed application development.

The major components of HP Distributed Smalltalk are shown in Fig. 2 and briefly defined below:

- HP Distributed Smalltalk ORB. This is a full implementation of the Object Management Group's Common Object Request Broker Architecture (CORBA).
- Remote Procedure Call (RPC) communication. This component supports efficient and reliable transfer of messages between systems.
- HP Distributed Smalltalk object services. This includes all standard object services required by distributed systems, as well as support for creating and maintaining objects and the relationships between them.

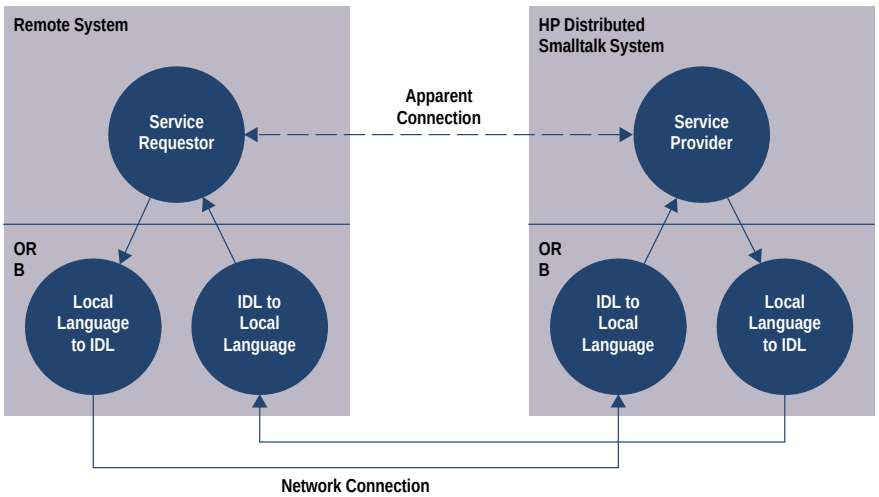

Fig. 1. Overview of the HP Distributed Smalltalk process.

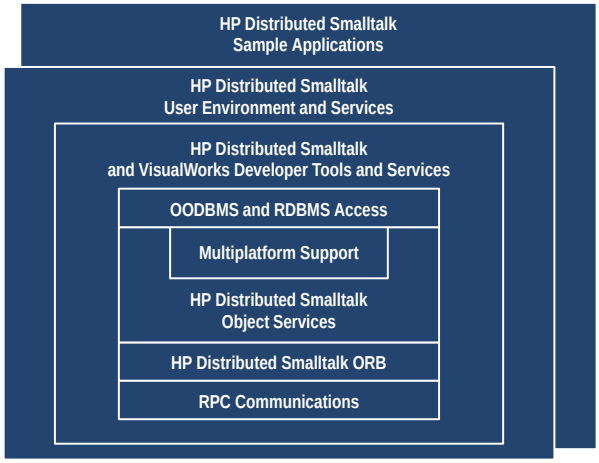

Fig. 2. The major components of HP Distributed Smalltalk.

- Multiplatform support. HP Distributed Smalltalk applications that run on one platform (hardware and operating system combination) can run, without porting, on any other supported platform.
- OODBMS and RDBMS access. HP Distributed Smalltalk provides database access directly to HP's Odapter† and Servio's GemStone as well as to Sybase and Oracle (via VisualWorks). HP Odapter can be used to provide access to a variety of other database systems.
- HP Distributed Smalltalk developer tools and services. This level of the framework provides support specifically designed for developing, testing, tuning, and delivering distributed applications. HP Distributed Smalltalk incorporates a rich development environment, application builder support, and the ParcPlace Smalltalk language.
- HP Distributed Smalltalk user environment and services. These services include a reusable demonstration user interface and desktop environment support for users' work sessions and normal desktop activity.
- HP Distributed Smalltalk sample application objects. These objects provide developers with example code that can be reused or extended, or can provide a source of ideas for developing alternate applications.

The following sections provide more detailed descriptions of the components that make up HP Distributed Smalltalk.

HP Distributed Smalltalk Object Request Broker

HP Distributed Smalltalk is a complete implementation of CORBA, the Object Management Group's specification of an object request broker. HP Distributed Smalltalk's compliance provides the basis for object and application interoperability.

CORBA specifies core services that are required of an object request broker to support interoperable distributed computing. The CORBA specification includes the following core services.

Interface Definition Language Compiler. OMG has defined the Interface Definition Language, or IDL, to be independent of other programming languages. Interfaces for objects that can provide distributed services are written in IDL so that they are accessible to service requesters that might be written in Smalltalk, C, C++, or another language.

OMG recently approved the IDL-to-Smalltalk language binding proposed by HP and IBM. This is important because it allows users to build distributed systems using multiple languages where appropriate, allowing a Smalltalk object to be able to request services of a C++ object or vice versa.

Interface Repository. This service provides a registry of distributable object interfaces for a given system. Any object that remote objects can access has an interface in the interface repository. For example, when objects on two or more systems at different locations collaborate in an application, they interact by sending messages to their interfaces. Since external clients have access to an object's services only through the object's interface, the implementation of the object is private. This privacy provides a variety of benefits, including security, language independence, and freedom to modify the implementation of how a service is performed without external repercussions.

HP Distributed Smalltalk ORB Support. The object request broker (ORB) is the key to providing support for distributed objects. By providing an ORB on each system, HP Distributed Smalltalk makes the location of any object transparent to clients requesting services from the object.

When a message is sent to a local object, the activity is handled normally. When a message is sent to a remote object, the remote object's local surrogate (created automatically by the ORB) intercepts the message, then uses the ORB to locate the remote object and communicate with it (see Fig. 3). Results returned to the calling object appear exactly the same, whether the message went to a local or remote object.

An ORB's responsibilities include:

- Marshalling and unmarshalling messages (translating objects to and from byte streams for network transmission)
- Locating objects in other images or systems
- Routing messages between surrogates and the objects they represent.

While a request is active, both client and server ORBs exchange packet information to track the course of the request

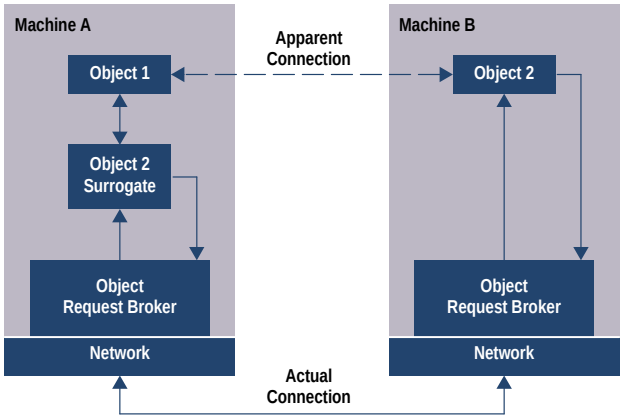

Fig. 3. HP Distributed Smalltalk handles remote access so that a request to a remote object appears the same as a request to a local object.

† HP Odapter is a complementary product from Hewlett-Packard that provides an efficient and scalable link between objects implemented in an object-oriented language such as Smalltalk or C++ and the entities in an Oracle relational database.

and resolve any network or transmission errors that might occur.

Object Services and Policies

Object services extend the core ORB services to support more advanced object interaction. HP Distributed Smalltalk implements OMG's Common Object Services Specification (COSS), which extends CORBA to provide protocols for common operations like creating objects, exporting and destroying objects (life cycle), locating objects (naming), and asynchronous event notification. Additional object services and policies provide efficient interaction between finer-grained distributable objects.

Naming.† There is a standard for assigning each object a unique user-visible name. Names are used to identify and locate both local and remote objects.

Event Notification.† This is a service that allows objects to notify each other of an interesting occurrence using an agreed protocol and set of objects.

Basic† and Compound Life Cycle. There are standard ways for objects to implement activities such as create and initialize, delete, copy, and move both simple and compound objects, externalize (prepare for transmission to remote systems), and internalize (accept objects transmitted from remote systems). Compound objects, built from simple objects, can include application components, anything that appears on a user's desktop (such as a document, a mail handler, or a graphics toolbox), complete applications, and so on.

Relationships: Containment and Links. Links allow networked relationships among objects. Objects can be linked together with various levels of referential integrity (determining how to handle situations when one of the parties to the link is deleted), and in one-to-one, one-to-many, and many-to-many relationships.

Together with links, containment establishes and maintains relationships between objects. Each object has a specific location within some container. Containers are related hierarchically. HP Distributed Smalltalk provides objects that implement a generic distributed container. Programmers can use these objects to build specific implementations such as an electronic mail envelope (containing components of a message) or a bill of sale (containing information about items in a shipment) with minimal extra programming.

Properties and Property Management. Properties are part of an object's external interface (owner, creation date, modification date, version, access control list, and so on). They are a dynamic version of attributes.

Application Objects and their Assistants. Application objects are relatively large-grained compound objects that end users deal with (e.g., a file folder or an order entry form). Application assistants are lightweight objects that implement most of the policies and participate in most of the services that desktop objects need to participate in. Application assistants function as the developer's ambassador into the object services. Application assistants can be stored and activated efficiently and provide the basis for future transaction support.

† This service is specified in COSS 1.0.

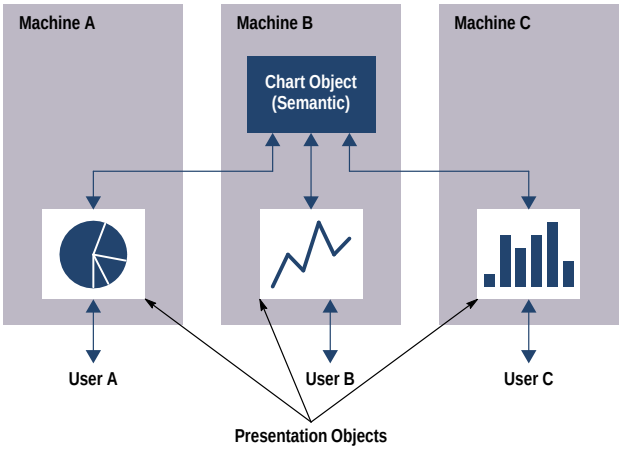

Fig. 4. The bulk of user interaction is with local presentation objects, minimizing and condensing the need to propagate semantically relevant changes over the network. Here for example, a user might choose to look at a chart (semantic object) as a pie, line, or bar chart presentation object.

Presentation/Semantic Split. A logical split between distributed objects, the presentation/semantic split provides an efficient architecture for distributed applications. Local presentation objects handle the bulk of user interaction, while a semantic object (which can be anywhere on the network) holds a shared persistent state of the object (see Fig. 4).

By using the presentation/semantic split, the designer can choose what part of the application should be shared and what should be unique to each user. Applications that might use the presentation/semantic split include a team whiteboard where all behavior is shared but each user can write comments, or a common document with pages that are unique to each user so that all users can read at their own pace. A variety of sample applications included with HP Distributed Smalltalk provide illustrations of how to use the presentation/semantic split.

While use of the presentation/semantic split is optional, it facilitates and optimizes distributed application development and execution. Advantages of using the presentation/semantic split include:

- Acceptable performance levels even over wide area networks
- Association of a single semantic object with multiple presentation objects, a critical feature in distributed computing environments where it is common for many users to work with the same application
- Application access independent of local windowing systems
- Better code reusability.

The HP Software Solution Broker described on page 93 is a good example of using the presentation/semantic split in an application.

Developer Services

HP Distributed Smalltalk also extends VisualWorks with services that support development and test of distributed applications.

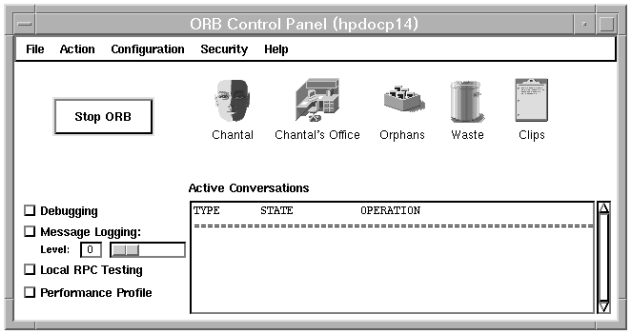

Fig. 5. The control panel provides an easy-to-use interface to administrative and developer services.

Control Panel. The technical user interface to HP Distributed Smalltalk for administrators and developers is invaluable for testing and maintenance (Fig. 5). The control panel provides:

- Controls to start and stop the system cleanly
- Support for local RPC testing (simulated distribution)
- Tracing facilities to log network conversations between objects
- Performance monitoring.

Interface Repository Browser and Editor. The interface repository browser provides an iconic view of the contents of the interface repository where publicly available interfaces are specified (see Fig. 6). It is organized hierarchically so that developers can explore and edit interfaces and construct requests to use the interfaces.

Shared Interface Repository. In HP Distributed Smalltalk, users can share an interface repository on a remote system so they do not have the overhead of keeping a copy of all of the interfaces on every system. The product also supports version management of interfaces, which is very important in large-scale, evolving distributed systems.

Remote Context Inspector and Debugger. This service is an extension that allows debugging on remote images when appropriate. It supports object inspection and debugging for

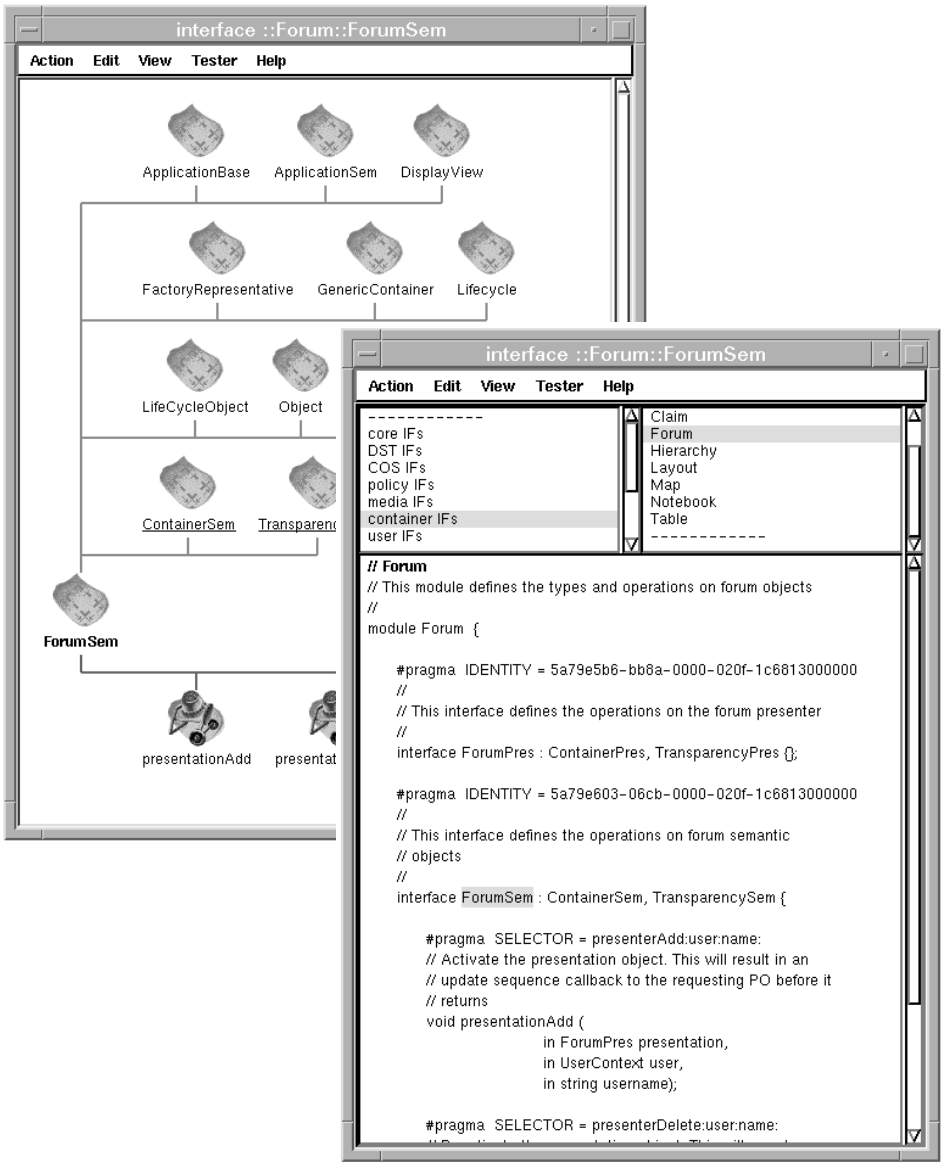

Fig. 6. The interface repository browser can be used to view or edit interfaces that remote clients can use to call local objects.

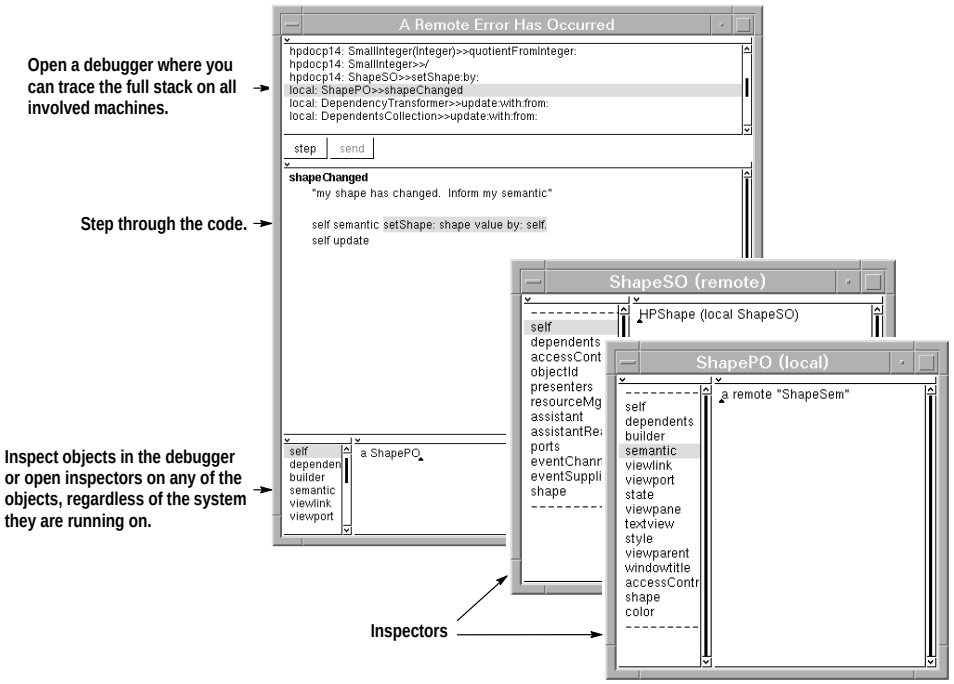

Fig. 7. Screens associated with a remote debugger.

the entire distributed execution context, including communication between images. Fig. 7 shows using the debugger to step through code and inspect objects that might be located anywhere in a distributed environment.

Stripping Tool. To prepare an application for delivery, developers use the HP Distributed Smalltalk stripping tool to remove unneeded classes and interfaces and seal source code when application development is complete. The stripping tool's user interface suggests likely items for removal (see Fig. 8).

User Services

User services allow developers to build a desktop or office environment and control activities during a session.

System Objects. HP Distributed Smalltalk supports a variety of system objects: user, session, clipboard, wastebasket, and orphanage.

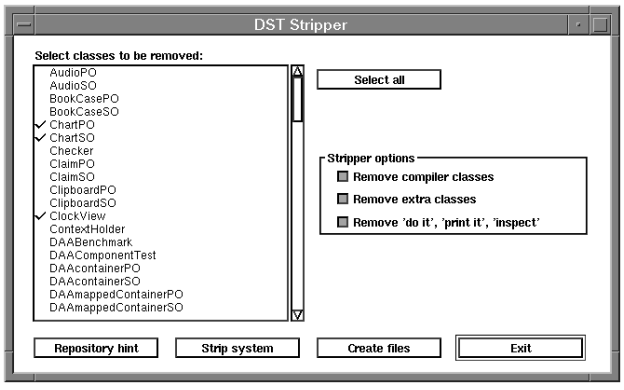

Fig. 8. Interface for the HP Distributed Smalltalk stripping tool.

- User object. This object contains information held about end users of the system including who they are, how to contact them, and so on. User objects may be included by reference in other objects. For example, a user might include a business card in a memo that would enable the receiver to get in touch with the sender.
- Session object. All the information required about the state of a user's environment, including user login, preferences, layout, and so on are contained in a session object. The session object also supports the notion of workspaces, with the potential for developing richer workspace environments. It has no icon on the desktop but it interacts with and supports other application objects.
- Clipboard. This is a container for objects that are being cut, moved, or copied from one location to another.
- Wastebasket. This container receives objects that users throw away. The wastebasket can be cleared when it gets too full.
- Orphanage. This is a container for holding objects that are no longer needed.

Security. Developers can use or extend HP Distributed Smalltalk's access control services in the applications they build, setting controls for host systems, users, or both. Host-system access control lets developers determine whether an image can receive messages from another system. User-level access control lets a developer determine whether a given user has any one of several kinds of privileges (e.g., read or write privilege) for a given object.

Developers can administer access control programmatically or from the default user interface.

Example Code

While all HP Distributed Smalltalk code is available to read, reuse, or extend, the default user interface and certain sample applications may be the best place to start.

Fig. 9. The screen for presenting the office metaphor and some typical objects in an office.

User Interface. HP Distributed Smalltalk uses and provides support for a user interface based on an office metaphor which is designed for easy use and understanding. In the default user interface, all the objects a user works with locally (folders, file cabinets, documents, and so on) are contained in an office. All offices on the same system are in the same building. Users can navigate between buildings to access objects in other offices. Fig. 9 shows a typical office and some of the objects available in an office.

Sample Applications. Sample applications illustrate the use of distributed objects. For example, the Forum (Fig. 10) provides a shared window in which several users can view and annotate a picture or document. The Notebook is a place to store both local and remote objects on a desktop.

Users can also build their own objects from any of the simple objects available, including a table, chart, input field, picture, and text window (see Fig. 11). The sample applications can be extended and customized to create a variety of simple distributed applications.

Creating Applications

HP Distributed Smalltalk allows VisualWorks programmers to create distributed applications quickly and easily. Building on the benefits of Smalltalk and VisualWorks, HP Distributed Smalltalk users can build CORBA-compliant applications either from scratch or by modifying existing applications. Like any Smalltalk application, the distributed development process is iterative and designed for dynamic refinement.

Development. Distributed application development is a four-step process.

1. Design and test the application objects locally.

2. Define the object interfaces and register them in the interface repository.

3. Use HP Distributed Smalltalk's simulated remote testing tools (which actually use the ORB to marshall and unmarshall object requests) to verify the interfaces specified in the interface repository.

4. Track messages and tune performance.

Distribution. Once an application is developed, tested, and tuned locally, it is easy to set it up for distributed use.

5. Copy the application classes to the Smalltalk images they will run on.

6. Update the interface repositories in these images.

The application can then run in the fully distributed environment without further change. Except for actual packet transfer, the distributed application is identical to the simulated remote application developed, tuned, and tested during development.

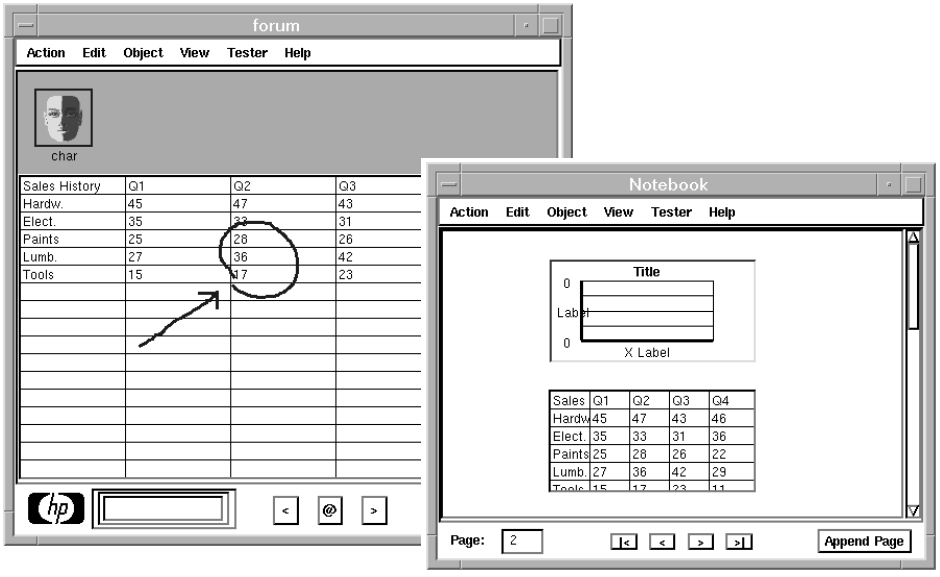

Fig. 10. User interfaces to the sample applications Forum and Notebook.

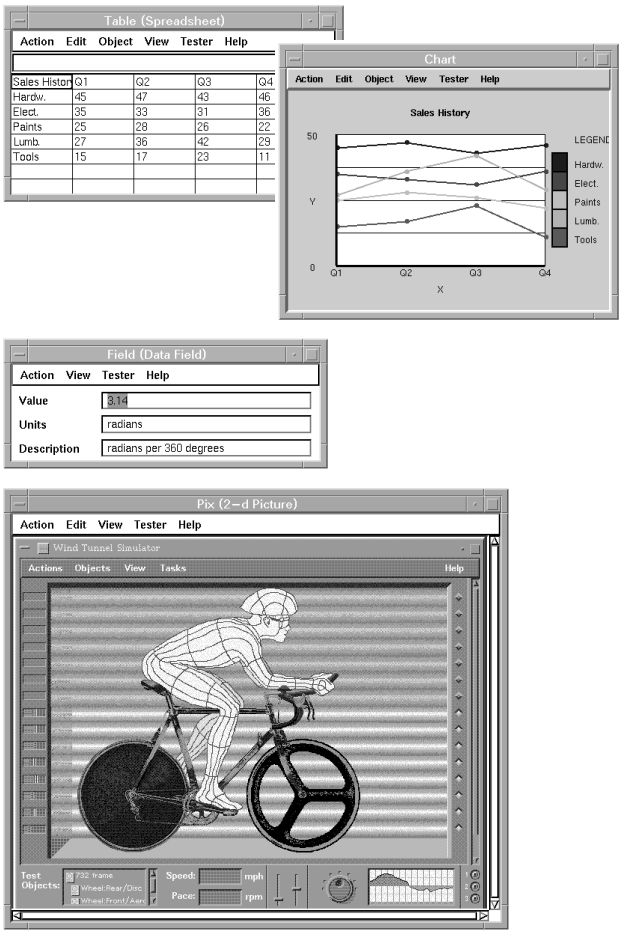

Fig. 11. Sample objects provided with HP Distributed Smalltalk.

Delivery. Once the application is tested, developers can deliver it to their users by stripping the environment of unneeded objects and tools. Once stripped, the application looks exactly the same as applications developed in other languages and can be executed on any supported platform, including: HP-UX,* SunOS/Solaris, IBM AIX, Microsoft® Windows, Microsoft Windows NT, or IBM OS/2. Support for these platforms is available under a run-time license from Hewlett-Packard.

Acknowledgements

We would like to thank the program team manager for HP Distributed Smalltalk, William Woo, for his unflagging support and encouragement. Dr. Jeff Eastman deserves immense credit for designing and building the first implementation of the project. We would also like to thank the HP Distributed Smalltalk program team in Cupertino, California and Fort Collins, Colorado for all their contributions: Lynn Abbott, Jerry Boortz, Jim Borchert, Kevin Chesney, Jürgen Failenschmid, Warren Greving, Robert Larson, Lynn Rowley, Terry Rudd, and Brent Wilkins.