VisualAge Distributed Technology: Technical Overview

Introduction

This paper describes the technology behind IBM′s* VisualAge Distributed feature, a set of Smalltalk classes and tools that are designed to support the development of distributed applications using the IBM Smalltalk environment. This technology is part of IBM′s overall strategy for distributed object-oriented development, which also includes the Distributed System Object Model (DSOM). This paper describes a technology, as opposed to products currently available.[1]

The paper first gives an overview of the advantages of VisualAge Distributed, followed by a description of the VisualAge Distributed architecture, and finally an overview of the proposed development process one would follow when using VisualAge Distributed.

The VisualAge Distributed feature provides:

- An easy-to-learn, high-level development paradigm with true local/remote transparency.
- An industrial-strength development environment with support for large-scale team development, with the distribution design and code changes kept synchronized in a central repository.
- Distributed garbage collection for efficient storage management.
- Compatibility with any products and tools built on IBM Smalltalk.
- Integration with the VisualAge* visual programming environment.
- Compatibility with the industry-standard Common Object Request Broker Architecture (CORBA), through IBM′s Distributed System Object Model (DSOM). CORBA is a standard developed by the Object Management Group (OMG).

With VisualAge Distributed, any Smalltalk programmer can create distributed applications after learning only a few additional concepts.

Why VisualAge Distributed?

Today′s software developers are facing increasingly complex and rapidly changing business requirements, as well as trends toward new technology such as graphical user interfaces (GUIs) and multimedia. Application developers must be able to develop and update their business applications quickly.

At the same time, developers must contend with the trend toward reducing costs by "right-sizing"—moving their information systems from centralized mainframe computers to networks of interconnected computers.

"Right-sizing" requires that processing be distributed. Developers must design their applications to run on networks of interconnected machines that share the processing load. Applications must make the best use of existing resources to meet the needs of the business, and to achieve performance, scalability, security, usability, and maintainability objectives.

[1] This paper represents IBM′s current technology direction and is subject to change or withdrawal.

Today's businesses are also finding the traditional client/server approach to be inefficient. Typically, most application logic in a client/server environment is located on the client machines, with the servers acting as centralized databases. This approach can lead to excessive network traffic and difficulty in maintenance. One solution to these problems is to put some of the application logic back on the server where it can access the data locally. This requires an approach much more flexible than the traditional client/server environment.

The VisualAge Distributed Solution

The Smalltalk language is an easy-to-learn and powerful development environment that can help corporate application developers move into the realm of object-oriented programming. Smalltalk is well suited to short development cycles, and lends itself to development of GUI interfaces as well as application business logic.

VisualAge Distributed provides all the support you need to adopt Smalltalk for development of industrial-strength distributed applications, including application logic located on a server.

VisualAge Distributed extends the basic Smalltalk model to support distribution of objects on different computers on a network. These objects can send standard Smalltalk messages to one another, regardless of their physical location. They can also freely send other Smalltalk objects as arguments, and receive objects as results. The different parts of an application can be located on any computer in the network that is running VisualAge Distributed.

VisualAge Distributed offers many benefits that ease the transition to object-oriented, distributed application development.

Smalltalk development model. Using the Distributed feature, IBM Smalltalk and VisualAge programmers can quickly develop applications that span networks, without having to learn the details of network communications, distributed application programming, or specialized interfaces. VisualAge Distributed extends the familiar Smalltalk tools—browsers, inspectors, debuggers, and workspaces—to the distributed environment. If you can use Smalltalk, you can use VisualAge Distributed.

High-level development paradigm. VisualAge Distributed enables developers to focus on solving the business needs of the enterprise, rather than upon the mechanics of distributed programming. VisualAge Distributed is based entirely on the Smalltalk model, providing a very high level of programming abstraction and removing much of the complexity of distribution from the task of program design.

Flexible distribution approach. The VisualAge Distributed model supports much more than a simple client/server approach. Applications can be "split" in many different ways, supporting both client/server and true peer-to-peer designs. The distribution of an application can be dynamically changed throughout the development cycle.

IBM Smalltalk foundation. The Distributed feature is based upon IBM Smalltalk, an industrial-strength object-oriented development environment. The industry-standard Smalltalk language provides a smooth transition to object-oriented development, and IBM Smalltalk provides an open and extensible programming environment that supports large-scale team development with version control.

IBM Smalltalk is the foundation for IBM′s VisualAge, an object-oriented visual programming environment. IBM Smalltalk applications are portable to any platform upon which IBM Smalltalk runs.

With VisualAge and VisualAge Distributed, Smalltalk programmers can quickly begin to build portable distributed applications.

Security. VisualAge Distributed is enabled to use the industry-standard Generic Security Service API (GSS-API) and currently supports the IBM product Network Security Program (NetSP), version 1.2. Used with NetSP, VisualAge Distributed provides transparent support for client authentication, message verification, and message encryption.

True server programming. VisualAge Distributed enables you to use Smalltalk everywhere in your client/server environment. It provides full support for true server programming:

- It is scalable to acommodate the demands of your business.
- Its security support provides for authentication of clients and servers.
- It supports concurrent execution of multiple client requests within a single server Smalltalk image.

VisualAge Distributed Architecture

VisualAge and the Distributed feature provide a two-level solution for distributed programming:

- Support for IBM′s CORBA-compliant SOM and DSOM.
- The Distributed Object Space, which takes advantage of "Smalltalk everywhere" applications to provide true local/remote transparency.

SOM/DSOM Support

The CORBA industry standard provides an object-oriented, language-neutral technology for distributed programming.

VisualAge Distributed coexists with IBM′s CORBA-compliant SOM and DSOM, providing a mechanism for distributed programming in heterogenous environments where some parts of an application are written in languages other than Smalltalk. SOM is an object-oriented programming technology for building, packaging, and manipulating binary class libraries. With DSOM, you can build a distributed application using any languages for which SOM bindings exist. Currently, this includes C, C++, and IBM Smalltalk.

VisualAge Distributed provides an interface through which Smalltalk applications can use objects implemented using SOM/DSOM. These objects can be developed in any language that has SOM bindings. With this interface, Smalltalk applications can create instances of SOM/DSOM objects and send messages to them.

The VisualAge Distributed DSOM support is implemented through the use of Smalltalk wrapper classes that represent DSOM classes. You can interact with these wrapper classes as though you are directly using the DSOM classes they represent.

For more information about using SOM/DSOM with IBM Smalltalk and VisualAge, refer to Appendix, "SOM and VisualAge SOMsupport" on page 11.

Distributed Object Space

When all parts of a distributed application are written in Smalltalk, VisualAge Distributed takes advantage of this situation with a technology called the Distributed Object Space. In effect, the Distributed Object Space extends the Smalltalk model across a network.

The Distributed Object Space is made up of multiple individual *object spaces*. An object space is a single instance of a running Smalltalk image, and it can exist on any computer attached to a network and running VisualAge Distributed. With this technology, objects can:

- Send standard Smalltalk messages to one another regardless of physical location.
- Send and receive other Smalltalk objects as arguments and results—even across the network— without having to "flatten" them into serialized data.
- Reside in different physical locations from one application version to the next, without requiring any change to the Smalltalk code.

The Distributed Object Space enables developers to use the full power of the Smalltalk language, including high-performance messaging and efficient storage management through automatic garbage collection.

Because the Distributed Object Space is based on Smalltalk, you can send Smalltalk objects as arguments to remote objects just as you would pass them locally, without any conversion necessary. The Smalltalk code

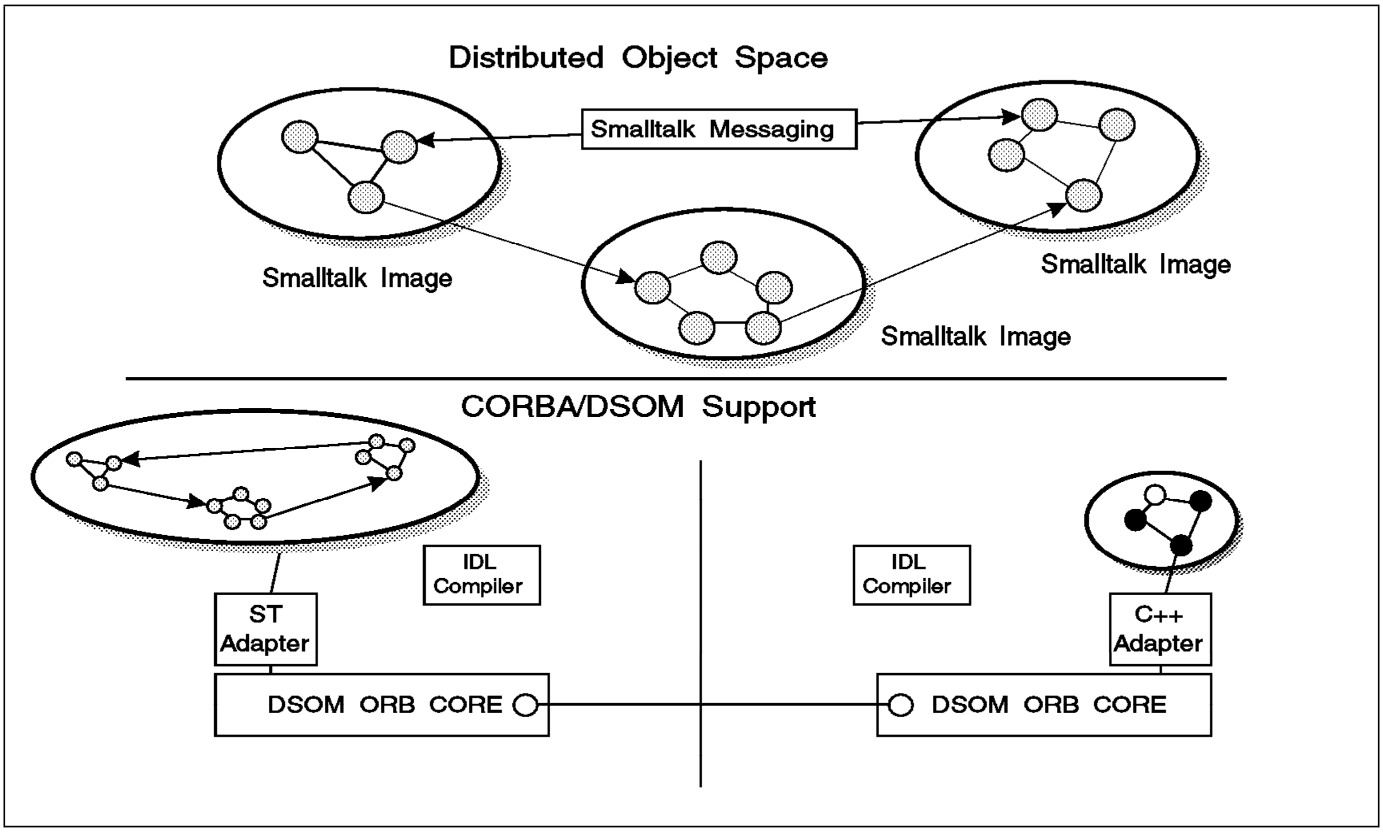

Figure 1. VisualAge Distributed distribution technologies. The Distributed Object Space provides local/remote transparency when all parts of the application are written in Smalltalk. The computers in the network can be running any operating-system platform supported by VisualAge Distributed. VisualAge Distributed also supports IBM's CORBA-compliant DSOM, for distributed programming in heterogenous environments.

used to send messages to remote objects is the same standard syntax used to communicate with local objects. Therefore, at development time, the programmer need not be concerned with the physical location of the objects in the application. In fact, the physical location of the application modules can be dynamically changed throughout the development process, without any change to the Smalltalk code.

Distributed Object Space runtime: The Distributed Object Space runtime environment provides all of the support necessary to implement distributed execution of Smalltalk applications with true local/remote transparency. The runtime environment provides an additional layer on top of IBM Smalltalk (see Figure 2 on page 6). It provides the following distribution services:

- **Messaging.** VisualAge Distributed provides all of the communications logic necessary to send Smalltalk messages between object spaces. Once configured with the necessary location information, VisualAge Distributed handles the low-level task of passing Smalltalk objects across a network connection.
- **Distributed garbage collection.** VisualAge Distributed extends Smalltalk garbage collection so that it is distribution-aware. Unused memory is freed when it is no longer being used by other local or remote objects.
- **Activation support.** VisualAge Distributed provides the necessary support to start any remote Smalltalk images that are required by the application. If an object in one object space sends a message to an object space whose image is not currently running, the VisualAge Distributed activator automatically starts it.
- **Name server support.** A name server is a directory of objects in a network. A name server contains *object references*, which provide information on the physical location of objects used by an application. By changing the object references, the developers can update the location information for the objects without having to change the Smalltalk code.

  If the name server cannot resolve a particular object name, it has the ability to relay the request to a second-level name server, and so forth. The VisualAge Distributed name server is designed to comply with the industry-standard OMG Naming Services Specification. The name server is also designed to be adaptable to other naming services, such as the Open Systems Foundation (OSF**) Distributed Computing Environment (DCE) directory service.

Distribution Toolkit: The Distribution Toolkit is currently planned as a set of tools to help developers and administrators design, develop, debug, optimize, and configure distributed applications from a single physical location. The Distribution Toolkit will itself be set of applications that run on top of the Distributed Object Space runtime support (see Figure 2 on page 6). The toolkit will include the following tools:

- **Remote Transcript and workspace windows.** A remote Transcript window is a local window that mirrors the System Transcript window of a remote object space, enabling developers to interact with a remote Smalltalk image as if it were local. Likewise, a remote workspace is a standard Smalltalk workspace from which objects in a remote object space are addressable.

- **Remote inspectors and browsers.** VisualAge Distributed extends the IBM Smalltalk inspectors and browsers to enable browsing and editing of remote objects and classes. Using these tools, you can browse other object spaces on the network; add, delete, and edit the classes that reside in those object spaces, provided you have authority to do so; and inspect and modify the contents of remote objects.
- **Distributed debugger.** VisualAge Distributed provides an enhanced Smalltalk debugger that shows distributed program execution as a single call stack. The debugger enables you to trace program execution that spans a network just as you would with an entirely local application. You can also make changes to remote code from within the debugger.
- **Name server UI.** VisualAge Distributed provides a graphical user interface you can use to browse and update the contents of the name server. This enables you to add, delete, and change object references and other named objects.
- **Distribution configuration editor.** This editor enables the developer to partition and distribute the modules of the application to the various object spaces on the network. Each module is a group of Smalltalk classes.[2] Using the distribution configuration editor, you specify which object spaces (images) each module should be loaded into. After you have specified the location for each module, you can also use this editor to automatically load the appropriate modules into the remote images.

  With the distribution configuration editor, you edit the *distribution matrix*, which designates the relationships between application modules and object spaces. (See Figure 3 on page 7 for an example of a distribution matrix.) For each application module on each object space, you specify one of the following options:

  – Do not load the module into this object space

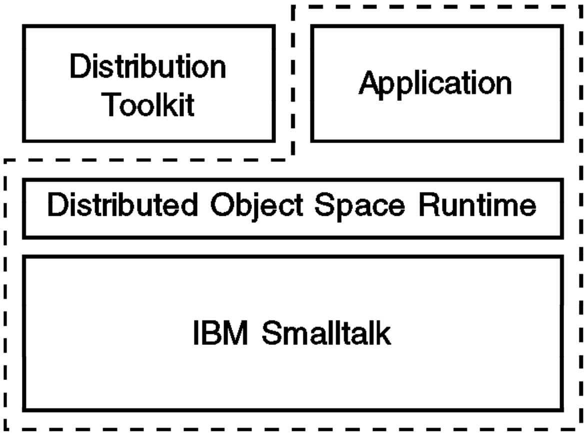

Figure 2. Distributed Object Space architecture. VisualAge Distributed provides the Distributed Object Space runtime support, which runs on IBM Smalltalk. In addition, the Distribution Toolkit will provide a set of development tools to help you distribute your application. The Distribution Toolkit is present only at development time.

[2] Each group of classes is actually called an *application* in IBM Smalltalk terms, although it may in fact be only a part of your complete distributed application. This is based upon the terminology of the IBM Smalltalk Professional environment.

– Load the module into this object space
– Load shadows for this module into this object space

A *shadow* is a local representation of an object that resides in another object space. The shadows provide local visibility for the remote module′s initial objects. Local objects send messages to the shadow objects, which in turn relay those messages to the true remote objects. Once the initial connection is made, the application can create and communicate with additional remote objects just as it would locally.

The distribution design of an application—the designation of which applications are loaded into which object spaces—is saved along with each version of the application. Developers can dynamically change the locations of the different modules of the distributed application to fine-tune performance.

The distribution design of an application is always kept in sync with any corresponding code changes. You can freely modify the design of the application to use a different distribution design without having to keep track of which versions of the code belong with each design. Each version′s code and distribution are stored together.

- **Remote Message Probe.** This profiles all messages traveling between object spaces. It shows you a stack containing every message sent from a particular object space. This lets you analyze your message traffic and evaluate the efficiency of your design.
- **Event Profiler.** The Event Profiler enables you to analyze the message traffic among the objects in your application. This helps you to decide the best way to split your application. For example, if two objects have a lot of traffic between them, they should probably be in the same object space, while two objects that seldom interact can generally be remote from one another.
- **Runtime configuration tool.** This tool is used to set up the components of a distributed application on the system upon which it will run. When an administrator installs a distributed application, the runtime configuration tool makes sure the VisualAge Distributed runtime environment can find all of the object spaces and objects that make up the application. Using this tool,

	Client Object Space	**Server Object Space**
Client module	Load module	Do not load
Server module	Load shadows	Load module
Common module	Load module	Load module

Figure 3. Example of a distribution matrix. Each application module is loaded into the appropriate object spaces. The client object space contains the client application module and shadow objects for the server module; these shadows enable the client objects to see the initial server objects as local. The server object space contains the server application module. Both object spaces are loaded with the module containing common objects used by both client and server.

the administrator configures the name server information to reflect the physical locations of the object spaces that make up the application.

VisualAge Distributed Development Process

The development of a distributed application with VisualAge Distributed would follow these steps:

1. **Design and develop the application.** Identify the business needs that must be accommodated by your application, and design the application to best meet those needs.

   During this design and development phase, you can concentrate on your business requirements without being concerned about the distribution design. In fact, you can build the complete, working application completely on a single computer and within a single Smalltalk image.

2. **Identify the "splits."** After the application is completed, tested, and functional, you can use the Event Profiler to identify the best places to split the application between object spaces. The Event Profiler analyzes the message traffic among the objects in your application, helping you determine which objects need to be local to one another and which objects can safely be remote.

3. **Split the application between object spaces.** Once you have determined the distribution design for your application, use the Distribution Toolkit to split the application into modules and distribute those modules among computers on the network. VisualAge Distributed provides all of the tools you need to load the different application modules into local and remote Smalltalk images. No change to your Smalltalk code is required when you split the modules among separate object spaces.

4. **Profile the application's performance.** Once you have distributed the application on the network, you can test the application and assess its performance. The Remote Message Probe, which measures the volume of message traffic traveling between object spaces, can help you identify performance bottlenecks.

5. **Tune your distribution design.** If you identify performance slowdowns resulting from too much network traffic, you can use the Distribution Toolkit to relocate the objects in your application. With the VisualAge Distributed tools, you can move objects from one module to another or move an entire module from one object space to another. Once again, you need not make any changes to your Smalltalk code in order to change an object's physical location.

   As you fine-tune your application's performance, you might also need to make some adjustments to the application itself in order to minimize message traffic. The VisualAge Distributed browsers enable you to view and modify the Smalltalk code in remote object spaces, meaning that you can continue to update your application without having to travel to each machine's physical location.

   Furthermore, the IBM Smalltalk team development environment stores the distribution design of your application—the location of the "splits"—along with the code for each version. This means that your code is always kept in sync with your distribution design. Each version of the application contains not only the code changes associated with that version but also any corresponding changes in the distribution design. You can also freely change your design to try different approaches, because you can always return to an earlier version.

6. **Debug the application.** After your application is distributed, you can continue to debug and make changes to it without disruption. You can edit local copies of the remote application modules, enabling you to test your updates before deploying them. When you have finished developing new versions, you can use the VisualAge Distributed tools to distribute the updated application modules to the remote images.

System Requirements

VisualAge Distributed is currently planned to have the following system requirements:

- An operating-system platform supported by IBM Smalltalk. Currently, IBM Smalltalk supports the following operating systems:
  - IBM OS/2,* Version 2.1 or later
  - DOS with Microsoft** Windows**

  As IBM Smalltalk becomes available on other platforms (such as AIX*), IBM intends to provide VisualAge Distributed on those platforms.
- IBM Smalltalk Team or IBM VisualAge Team
- A communications package supporting the Sockets API. Currently, VisualAge Distributed supports TCP/IP.

Getting More Information

For more information about IBM Smalltalk and VisualAge, call 1-800-IBM-2279 or 919-254-4760.

For education offerings, call:

- (US) 1-800-IBM-TEAC
- (Canada) 1-800-661-2131

To order VisualAge Family products, call:

- (US) 1-800-IBM-CALL
- (Canada) 1-800-555-SW4U, ext. 662

On CompuServe:** GO VISUAL.

Outside North America, contact your local IBM office.

Appendix. SOM and VisualAge SOMsupport

IBM′s SOMobjects* technologies, including the System Object Model (SOM) and Distributed Systems Object Model (DSOM), were designed to enable the creation of industrial strength, binary object classes that are truly reusable and are scalable to client server configurations.

SOM is a language-neutral object model that allows developers to package object classes in such a way as to provide an enhanced ability to reuse, modify and customize them within and among different language compilers. SOM, initially used in the development of the OS/2 Workplace Shell, will become a core technology for packaging object frameworks that are designed to be extended by others, such as the forthcoming OpenDoc compound document architecture from IBM, Apple**, Novell** and Wordperfect.** SOM has the unique capability to separate the object′s type from its implementation, resulting in a very flexible and dynamic model for developing object-oriented applications. DSOM, a scalable extension to SOM, provides transparent, distributed object services that are fully compliant with the Object Management Group′s CORBA specification.

IBM′s System Object Model (SOM) is not intended to replace existing object oriented languages (such as Smalltalk). Rather, it is intended to complement them so that application programs developed in different language and with different compilers can share common SOM object class libraries that are CORBA compliant. SOM also provides a rich set of object-oriented characteristics that can complement existing object-oriented languages (such as Smalltalk and C++), as well as procedural languages (such as C, COBOL & REXX).

By integrating the strengths of the IBM Smalltalk and VisualAge development environments with IBM′s SOM and DSOM technology, developers and end users will benefit from the following enhancements to the VisualAge development environment:

- **Reuse of object classes developed in other languages.** With currently available development tools, object classes developed in one language environment cannot be effectively reused in another language environment. The VisualAge SOMsupport will enable VisualAge developers to reuse and subclass object classes developed in other languages. Additionally, it will enable VisualAge developers to create Smalltalk object classes, conforming to the semantics of SOM′s Interface Definition Language (IDL), that can be reused by other languages (if they also support SOM). This can significantly increase the availability of object classes to be used in VisualAge as well as allowing VisualAge classes to be used effectively in other development environments (such as C or C++). In all, this will allow developers to reuse more code, thus resulting in less costly and risky development.
- **Support for CORBA Distributed Objects.** The Object Management Group′s CORBA specification (as fully implemented in IBM′s SOM/DSOM technology) was designed to provide a productive, flexible and dynamic client/server solution by exploiting the power of distributed object services. IBM′s DSOM is a scalable extension to SOM that provides local/remote distributed object services across a heterogeneous network.

VisualAge SOMsupport is planned to support the DSOM extensions and the ability to develop CORBA-based distributed objects (deployment across a workgroup requires the IBM SOMobjects Workgroup Enablers). VisualAge SOMsupport can significantly decrease the costs and risk associated in developing, deploying and adapting client/server applications while easing the migration from local objects to distributed objects.

- **Support for SOM-based OS Services.** Over time, an increasing number of operating systems services will be packaged as SOM object classes and frameworks. The first of these frameworks was the OS/2 Workplace Shell, which allows developers to customize, extend and integrate into the Workplace Shell desktop, as well as inherit behavior and characteristics (such as drag/drop) when developing new applications. OpenDoc, a compound document architecture being developed by IBM, Apple, Novell, and WordPerfect and supported by others, will be packaged as a framework of SOM object classes that can be used in developing document-centric, collaborative applications. VisualAge SOMsupport is positioned to exploit these frameworks as they are made available, which will help to automate the design and development of a new wave of document-centric, collaborative applications.

- **Industry Standard Class Definitions.** The SOM interface definition language fully conforms to the Object Management Group′s CORBA Interface Definition Language (IDL) standard, as does the SOM interface repository (providing runtime access to class information and definitions).

  The VisualAge SOMsupport Smalltalk bindings will evolve to be compliant with OMG standards as they are made available. VisualAge SOMsupport classes will, therefore, adhere to industry standards (CORBA IDL), allowing customers to retain investments in skills and code. It also provides a mechanism for interoperability, over time, with other CORBA-compliant object request brokers.

In summary, SOM, when integrated with the VisualAge development environment, can result in a more productive and powerful set of application development capabilities.

For more information on the SOMobjects Developer Toolkit, call 1-800-342-6672 in the US, or 1-800-465-7999 in Canada.

IBM®

This document contains information about products that are not available and describes planned technology only. This information represents IBM′s current intent, is subject to change or withdrawal, and represents only goals and objectives. User management is responsible for the choice of appropriate products and services to meet its needs.

This publication may refer to products that are announced but not currently available in your country. This publication may also refer to products that have not been announced in your country.

This publication could contain technical inaccuracies or typographical errors.

IBM′s VisualAge products and services are not associated with or sponsored by Visual Edge Software, Ltd.