The ABCs of Enterprise Application Integration

By
Jason R. Longo
Technical Director, PRL Scotland Ltd.

The key to developing a Enterprise Application Integration (EAI) Architecture is recognizing that there are different levels of integration, each with its own particular requirements and considerations. The successful implementation of consistent, scalable, reliable, incremental and cost effective solutions depends on the standards and methodologies that are defined for these levels. Consider the following four questions:

1. How is information to be shared within an Application?
2. How is information to be shared between Applications within an Enterprise (A2A)?
3. How is information to be shared between Enterprises (A2B, B2A and B2B)?
4. How is information to be shared directly with Customers (C2B and B2C)?

In answering these four questions an architecture emerges that is complete and covers the broad spectrum of integration requirements.

1. Applications Development - CBD

When writing the standard for new application development there is complete freedom to select the exact technologies and methodologies that are best suited to the enterprise. In most situations, Component Based Development (CBD) is the approach of choice. It promotes the development of tightly coupled, well defined, and highly reusable components. If the standards are adhered to throughout the enterprise, CBD can significantly reduce the cost of building and maintaining applications.

Microsoft’s Component Object Model (COM) was designed specifically for this purpose. The CORBA (Common Object Request Broker Architecture) standard as implemented in several different ORB products also fits in this space as does Enterprise JAVA Beans (EJB). Most proprietary component development environments rely on one of these three standards for inter-component communication.

Components written to the same standard utilizing any one of these technologies can easily interoperate in a flexible and scalable environment. Mixing these technologies, however, will result in an environment that requires another level of integration.

2. A2A (Application to Application)

Certainly, two applications, running on the same computer, written to the same standard can communicate easily using COM. This is especially true if the components are designed and maintained by the same organization so that they can be upgraded and released in such a manner as to ensure inter-application integrity. Unfortunately, the vast majority of application-to-application integration does not fit this perfect world. CORBA and EJB are better suited for communicating between different computers, but both have their short comings. While CORBA itself is a standard, the products that implement it (ORBs) are anything but standard. They are difficult to implement and do not communicate well with each other. None of these three technologies were designed to seamlessly work with the others.

A true EAI architecture must recognize that the applications that run within an enterprise were written to different standards, utilize different technologies and are implemented across distributed heterogeneous computer platforms. The installed base of applications consists of a mix of 3rd party solutions (old and
new), entrenched legacy systems and possibly a few applications written to the new component based standard. Clearly, the majority of A2A integration is not going to be accomplished as simply as having one component communicate with another solely via COM, or CORBA, or EBJ.

We think component-based interfaces can be used for some kinds of integration, and they will be. For example, SAP and its BAPI interfaces, and CORBA. But by itself, it’s not enough because these object request brokers don’t have all the features of the message brokers.

Roy Schulte, GartnerGroup

Messaging middleware products such as IBM’s MQSeries, Microsoft’s Message Queue Server (MSMQ) and BEA’s MessageQ were all designed specifically to handle the complexities of exchanging information within a distributed computing environment. Rather than being based on a synchronous component invocation, they reply on the concept of guaranteed message delivery to implement asynchronous, time-independent communications. Again, there are different technologies designed to address different integration issues.

The complete A2A architecture must embrace the new component standard for application development, middleware products for cross-platform communication, as well as all of the technologies and standards that have already been deployed within the Enterprise. Businesses can not afford to replace or re-engineer all of their mission critical applications. A quality message broker is an excellent solution. It provides the means to bridge the various islands of technologies and standards. Utilizing intelligent routing, content translation and transformation, process flow control, and middleware brokering, a message broker can seamlessly pass information from one standard to another, invoking the appropriate technologies to ensure real-time, inter-process communication.

Enterprises that thrive in the future will be those that can rapidly assimilate packaged applications and reuse existing applications in new ways. They understand that systems built to change are ultimately more valuable than systems built to last. The key to their success will be how to modularize their application portfolio and organize the connections among the systems.

Roy Schulte, GartnerGroup

At the heart of the I/O Exchange integration environment is the Interface Engine. It is a fully featured, next generation message broker. It combines the capabilities of a standard message broker with a middleware broker, and adds the unique capacity to accept component plug-ins. These plug-ins are written and designed to provided services within a distributed computing environment, but can be accessed directly by the Interface Engine without the involvement of middleware. This incredibly flexible and scalable approach to service provision and component invocation mixes the best features technologies such as COM and CORBA with messaging products such as MQSeries, MSMQ and MessageQ. The Interface Engine can employ parallel processing and asynchronous, time-independent processing hand in hand with synchronous component invocation.

The link between a specific application and the Interface Engine is handled by an I/O Exchange Application Wrapper. Application wrappers are written to bridge the technology and standards of the application with I/O Exchange. There are a number of standard Wrappers available as well as the tools for writing new Wrappers. Applications communicate with the Interface Engine via their Wrappers. The Interface Engine communicates with Applications via their Wrappers. Applications do not communicate directly with Applications. The technology and standards of one application no longer need to be distributed to any other application. Applications are no longer tightly coupled. They are no longer
bridged via one-off, single-use, interface specific programs that need to be maintained each time either application changes. They are now loosely coupled via the multi-use Wrappers and the table driven Interface Engine.

All Application Wrappers are written using the I/O Exchange Super API which makes programming with any of the leading middleware products as easy as seven simple calls. The API also has the advantage of allowing any component written using it to be managed and monitored using the I/O Exchange Manager.

3. B2B (Business to Business)

When sharing information between businesses, the problems confronted with A2A integration are only compounded. Each enterprise has its own set of applications, technologies, and standards. One enterprise rarely has any say over the specific standards and technologies another chooses to implement.

When you are integrating multiple enterprises in an Internet context, it's necessary for that integration to be loosely coupled. Historically, tightly coupled technology was used to build e-business applications, whether it was Component Object Model (COM) or Common Object Request Broker Architecture (CORBA). And the type of technology that is typically used to provide loosely coupled integration is messaging technology.

Mike Gilpin, Giga Information Group

Clearly, what made the combination of components, middleware and I/O Exchange a powerful architecture for A2A, is even more applicable to B2B. Since all I/O Exchange components can communicate, out of the box, using XML, it is a natural fit with Microsoft’s BizTalk and other XML-based B2B gateways. Since any interface defined within the Interface Engine can be invoked via COM (the COM class can even be generated automatically), it is a natural fit with Microsoft Web Server and ASP. Since the I/O Exchange Super API is available for JAVA, writing servlets and EBJs that communication with back-end systems is no different than using I/O Exchange for A2A solutions. This gives us a single, standardized methodology for providing A2A, A2B, B2B and B2A solutions.

The Architecture now flows from Application A (new or old, component-based or otherwise) to its Wrapper to the Interface Engine to web gateway A to web gateway B to the Interface Engine B to Application B via its Wrapper. Each application can be implemented using the latest standards, purchased from a best-in-class 3rd party supplier, or written thirty years ago in COBOL. Each interface is table driven and maintained in a Windows-based Interface Repository. If applications change or are replaced, only the Wrappers and Interface Engine definitions need to be changed. Applications, whether local or owned by another enterprise residing on the other side of the world, are no longer tightly linked and bound together. The Wrappers (which only need to be changed if the technology of the wrapped Application changes) and Interface Engine provide an insulation layer that ensures application integrity and significantly reduces the cost of interface development and maintenance.

4. B2C (Business to Customer)

The answer for question 4 is the same as for question 3 but for slightly different reasons. Here the emphasis is on flexibility and ease of change. In order to compete, businesses must be able to constantly update their customer offerings, and provide personalization. The architecture must be adaptable enough to support ever evolving input devices and technology.
Again the combination of web servers and I/O Exchange put the information customers require at their finger tips. The memory resident nature of the Interface Engine definitions means they can be updated on the fly, in real time, without impacting system availability. Businesses are better armed to react to the ever changing demands of an increasing sophisticated and demanding consumer.

Conclusion

COM, EJB and other such technologies are excellent for building new, component based applications. They are not, however, particularly suited for enterprise-wide integration. If used for this purpose, they result in tightly coupled applications that must conform to the same standards to communicate. Messaging middleware products provide the transport for supporting a looser coupling, but can be difficult to program and lack the features of a broker. Web gateways provide the means to do “dock-to-dock” delivery of information, but lack the facilities to effectively integrate with the back-end applications. What is missing is the bit in the middle. The bit that can use components and messaging, that can invoke or be invoked by web servers, that can bridge the islands of technology and standards that exist within any enterprise.

An EAI architecture has been outlined above that addresses new application development via CBD; application-to-application integration employing components, messaging, and I/O Exchange; business-to-business integration leveraging off the foundations of solid A2A integration by adding web servers and gateways such as BizTalk to the I/O Exchange backbone; and business-to-customer integration by tailoring existing B2B approaches and incorporating new technologies to accommodate customer specific personalization. The architecture is flexible, scalable, and designed to embrace new standards and technologies as they immerse. It is designed to help IT implement and support robust, change resilient systems that meet the requirements of an ever changing business.