By Max Dolgicer & Gerhard Bayer

MESSAGE Brokers vs. APPLICATION Servers FOR INTEGRATION

It comes as no surprise that a series of recent surveys by Morgan Stanley, the prominent investment bank, determined that EAI remains among the two top IT priorities for major corporations. Application integration and application security have held the two top spots since January 2002. The same survey found that development of custom e-business applications remains a key approach to support requirements, despite the trend toward deploying packaged applications.

In this article, we'll share our views on how two major middleware technologies—application servers and message brokers (hereinafter referred as integration brokers)—are being used to facilitate application integration in different types of projects.

Since over the last decade or so, our company, International Systems Group (ISG), Inc., has performed many application development and integration projects across a variety of major industries, our views are based on real-world project examples. However, it's important to emphasize that our views have also evolved ›

gradually in line with the middleware technology evolution as well as the shifting business and technical requirements and IT policies of many client organizations.

We'll examine the evolution of middleware technology in order to provide a context for understanding how the divide between application servers and integration brokers emerged. We'll also consider how segregation on the "supply" side (i.e., commercial software products) is mirrored on the "demand" side (the needs of IT projects) and we'll explore decision factors for finding the right approach based on your requirements.

Technology Evolution, Not Revolution

When EAI established itself as a distinct market niche, an interesting question emerged: Are integration projects so different from application development projects? A few years ago, the answer was clearly yes. EAI projects did seem different from development projects because of their dissimilar goals, the organizational changes they usher in, costs, vendor/product choices, skills required, and often even the business goals driving the initiatives. (Think for a moment about the skills required to implement an EAI project using any of the integration brokers compared to the skills required to roll out a major, new J2EE application.) This common view has been challenged quite often, mostly due to numerous facts, including:

- A number of strategic, large-scale EAI projects have been replaced by less ambitious and more tactical projects where just a few applications need to be integrated (i.e., application integration [AI]).
- A bad economy, long implementation cycles for integration projects, and our realization that integration without coding is simply an unrealistic goal despite some vendor attempts to convince us to the contrary.
- Finally, the advancements that vendors on both sides (i.e., integration vendors vs. application platform vendors) made to position their product to accommodate both types of projects have challenged the common view even further.

Regardless of the challenges, we will attempt to address when integration brokers are a more suitable choice for integration projects and when application servers can do the trick. A brief review of how middleware used for application development has evolved as opposed to middleware used for application integration will provide the context for this comparison.

Evolution of Middleware for Application Development

The '90s saw a fundamental shift from procedural programming approaches, such as COBOL, to the object paradigm, dominated by C++ and standards such as Common Object Request Broker Architecture (CORBA) and Component Object Model (COM). However, these technologies required a large amount of handcrafting services that fall into the category of "plumbing." Companies had to build and maintain a code base that dealt with low-level middleware services that would provide functionality to the business application layer. Much of this functionality resembled concepts typical for container-based systems such as Enterprise JavaBeans (EJB) or Microsoft's Transaction Server (MTS).

Due to increased pressure on IT to deliver solutions at low, predictable costs, companies increasingly have bought into advanced commercial middleware solutions to develop and deploy new e-business applications. Java 2 Enterprise Edition (J2EE) is widely accepted and many enterprise architectures now incorporate application servers. Though J2EE has become a platform of choice for mission-critical, line-of-business applications, many organizations are comparing it to Microsoft's .NET in hopes of selecting the best choice for different types of business application requirements.

Evolution of Middleware for Application Integration

EAI approaches have evolved differently from the development discipline. While application development has seen convergence on concepts, such as object-orientation and components, and the emergence of standards, such as CORBA and later J2EE, the integration space hasn't had such an evolution. Companies are still using a mixed bag of tricks for integration projects, including:

- Batch-oriented technologies, such as file transfer and extract, transform, and load (ETL) tools
- Message-oriented middleware (MOM) combined with custom developed adapters for near real-time integration
- Commercial integration broker products.

For the most part, there haven't been any standards that would provide focus to the plethora of products. XML has certainly become an important part of integration technology, but it deals only with a small subset of the issues to be addressed. A few vendors have adopted additional XML-based standards such as XML PATH Language (XPATH) and eXtensible Stylesheet Language Transformation (XSLT), while other vendors are already abandoning those to roll out more efficient solutions.

The independent evolution of middleware technologies has culminated in two, well-established key types of commercial middleware products:

- Application servers
- Integration brokers.

Today, application servers and integration brokers still represent a significant polarization. They don't share a common technology base, and one is built around

business integration journal takeaways

BUSINESS

- Integration project business sponsors with help from their IT architects need to determine project focus: Does the business require the implementation of new processes, or is the goal simply to propagate business events among different business domains and their supporting applications.
- From a budgetary perspective, the business needs to know the scope of an integration project (i.e., true EAI vs. just application integration (AI); a mismatch between integration scope and selected products can result in unnecessary spending.

TECHNOLOGY

- Application servers and integration brokers are converging technologies and the functional overlap is increasing. However, their distinctive qualities are still significant, and the decision of one or the other can usually be well-founded.
- A key project characteristic to watch out for is the degree of integration required: Do you need true EAI or just AI; in the latter case, a plain vanilla application server may be sufficient.
- Often, the amount of new business logic to be developed determines another dimension of project focus (i.e., development focus vs. integration focus), which, in turn, determines the technology choices, and application servers being a logical choice for development-focused projects.

standards while the other has mostly grown from proprietary implementations, although that is changing, too.

Application Servers

Application servers provide a component model and a container-based hosting environment, including a wide range of middleware services, along with tightly coupled deployment tools and somewhat less tightly coupled development tools. This technology relieves application developers from the need to develop complex, lower-level middleware services and lets them concentrate more on coding business rules. The benefit is faster time-to-market and increased ability to solve real business problems. This concept found its most prominent expression in the set of J2EE standard specifications and the supporting implementations from vendors such as BEA, IBM, Sun, and Oracle, and, to a lesser degree, open source products such as JBoss.

Today, most J2EE-based application server products include support for distributed transactions, security, persistence, load balancing, application-level clustering, and many additional services. Together, these form a comprehensive application platform for highly demanding applications. They facilitate reliability, scalability and availability, while at the same time automating some of the tasks application developers would have to worry about. Figure 1 shows how parts of a J2EE application server play together:

- Visual tools for development and deployment configuration
- A framework with prebuilt, partially generated code that's combined with custom business code
- A deployment environment to host the applications.

Application servers are now transcending their original purposes of providing a comprehensive environment for application development and deployment. Recently, application servers have started to embrace application integration challenges. They exploit their strength as a standards-based, central hub for new business components to also function as an integration hub. The integration capabilities have evolved from proprietary connectors that facilitated point-integration with legacy systems and packaged applications to new technologies based on J2EE standards. These include support for Java Messaging Service (JMS), the Java 2 Connector Architecture (J2CA), and Web services.

Integration Brokers

Initially, integration brokers originated as hub-and-spoke systems, where the broker product forms an integration hub and applications that need to be integrated reside on the spokes. Typical integration broker products include:

- Adapter technology to establish connectivity to legacy and packaged applications
- Some form of asynchronous MOM system to propagate events (and, of course, the data that represent these events)
- A rules engine to govern intelligent routing and data transformation.

The integration hub approach simplifies integration of additional applications, since it reduces a total number of interfaces needed to be integrated. As Figure 2 shows, companies have built a patchwork of connections between custom legacy applications, packaged software and data sources, using various interface mechanisms. As the number of connected applications grows, this approach becomes too costly in terms of development and maintenance. Assume an interface has been implemented such that events are propagated from application A to application D. If application E is to receive the same information, a new interface has to be developed, probably using different means of connectivity. This continues for every application that enters the integration scenario. The hub approach reduces this effort, since it decreases the number of interfaces to be integrated. In addition, the prepackaged adapters shield developers from the low-level details of how to establish connectivity.

The commercial integration broker products from companies, such as IBM, SeeBeyond, TIBCO, webMethods and a few others, have matured and evolved into comprehensive (and complex) product suites. Today, many integration vendors have built out their adapter offerings and most provide a universal transport layer that supports and includes:

- Proprietary MOM

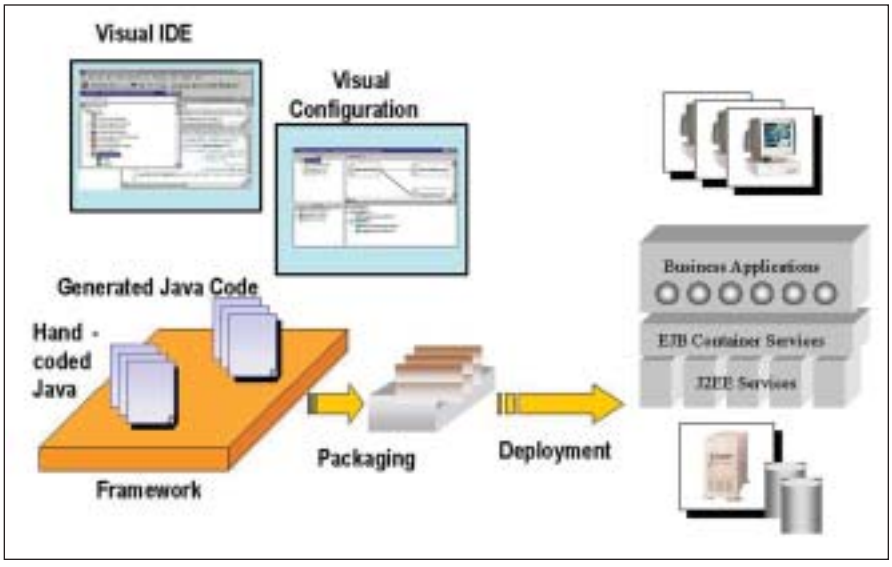

Figure 1: Application Server "Parts"

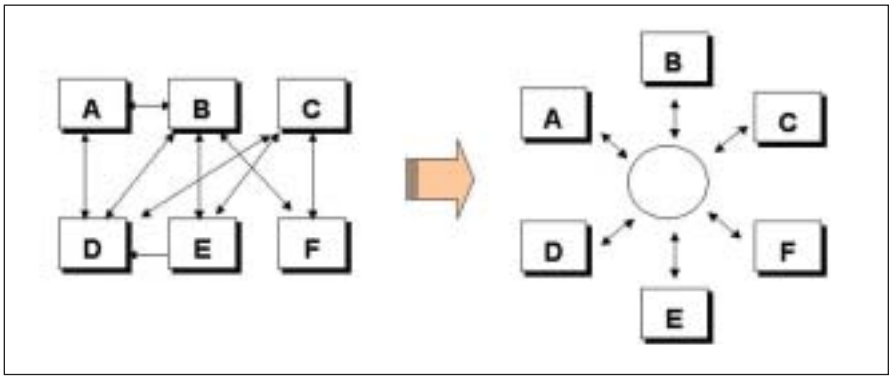

Figure 2: Point-to-Point vs. Hub-and-Spoke Integration

- Simple Object Access Protocol (SOAP)
- JMS
- Other forms of file transfer.

This facilitates different communication models (e.g., bulk transfer, synchronous and asynchronous, publish/subscribe, and message queuing). These vendors have extended their product suites with business process management (BPM), workflow, and support for Web services. Figure 3 shows typical building blocks of the integration broker environment.

Traditionally, integration brokers have focused on the concept of propagating business events. Events that become available from one application silo are forwarded to other applications based on predefined business rules. Generally, this concept applies to an environment that doesn't involve interaction with the user. However, most integration broker products now also support user-driven applications through front-end capabilities such as Java Server Pages (JSP). This shouldn't be confused with the capabilities typically employed by application servers to build and deploy new e-business applications. Most integration brokers don't provide a standards-based container that's geared to manage newly developed business logic such as an EJB container does. There are no standard interfaces between business logic components and a container that would facilitate portability and support automatic, container-managed services such as persistence, transactions, and security. This is an important distinction that will be discussed in the concluding section.

Integration brokers try to reduce the application integration effort by providing prebuilt functionality common to many integration scenarios. The value proposition rests on reuse (in terms of the middleware infrastructure and the integration logic) across multiple applications and integration initiatives. However, some companies struggle to achieve a positive ROI in an appropriate time, which is one of the reasons the number of ambitious, large-scale integration projects has been declining.

Now let's take a closer look at the demand side (the needs of IT projects). Since supply and demand are obviously interrelated, the segregation pattern on the product side often is mirrored on the project side.

The Traditional Project Divide

Traditionally, IT projects have been segregated between development- and integration-focused projects. This was due to significant differences in:

- Business objectives
- The technology base for existing systems vs. new applications
- Solutions provided by vendors
- Skillsets required for implementation and maintenance.

In fact, development and integration projects often rely on separate budgets.

Development-Focused Projects

The primary objective of typical development-focused projects is implementing new component-based e-business applications. We refer to this as the development-centric approach, since the development of new logic is the overriding concern; while integration with existing systems is usually important, in most cases it is treated as a secondary concern. These projects follow the well-known rigor of the application development life cycle (definition of the application architecture, design and implementation of business components, testing and acceptance). There are proven development methodologies and tools to manage all phases of this life cycle. For example, object-oriented design and development are widely accepted and, although programming languages such as Java and C# differ in their implementation, the basic principles are the same. At the design and modeling level, Unified Modeling Language (UML) is widely used; it has been standardized, and many developers are familiar with tools such as Rational's Rose suite.

Even in a development-focused project, there are always application integration requirements. However, typically, these requirements aren't on the scale of large EAI initiatives but rather what is often referred to as "AI," (application integration).

AI typically entails integration of a rather limited number (three to five) of integration points (i.e., interfaces to legacy applications, databases, packaged applications, etc.) and not hundreds of interconnected interfaces. These integration points will be implemented through custom-built adapters, or wrappers, around the legacy systems or packages that need to be incorporated into the overall new system. In such projects, the emphasis is on a systematic application development and deployment, and this is where the bulk of the project work is centered. The integration work on such projects is not a major focus, thus making any major investments in purchasing a comprehensive integration platform—in addition to the development platform—cost-prohibitive.

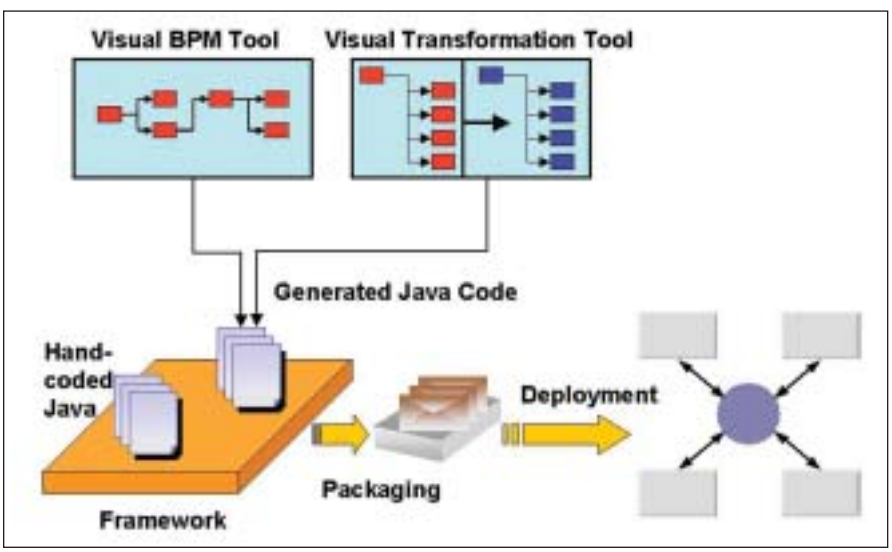

Figure 3: Typical Building Blocks of Integration Broker

Integration-Focused Projects

The primary objective of typical integration-focused projects is large-scale integration of legacy and packaged applications with the intent of integration proliferation to the enterprise level. This usually entails creation of a center of excellence (COE) for integration and agreement (at least perceived) on what corporate standards, including integration products, will be implemented. We refer to this as an integration-centric approach since integration of existing systems is the overriding concern while development of new business logic is a secondary concern.

There's really no typical integration project. They vary widely in scope, requirements, and the approach that companies take. Many solutions have evolved through one-off design of interfaces in a

point-to-point fashion, as opposed to a hub-and-spoke architecture or any other systematic approach that could scale.

There are still many projects that use batch file transfer for most integration points and where message-based, near real-time integration is rarely used. Such implementations will be replaced with near real-time integration only as business drivers demand. Sometimes, the business side doesn't require the capabilities of today's integration brokers even though such technology is capable of transforming a company into a real-time enterprise (RTE). Then there are those companies where the business drivers really mandate moving toward an RTE.

Gartner defines an RTE as a company that "competes by using up-to-date information to progressively remove delays in managing and executing its critical business processes." An RTE does that by streamlining and automating processes, focusing on parallel activities, and rapidly redistributing workload across available resources.

Integration brokers seem to be the obvious choice, at least for now, and many RTEs are successfully using this technology. However, we're also witnessing that some organizations struggle to achieve a positive ROI within a reasonable timeframe—and management responds accordingly by halting or slowing further investments.

Some reasons these projects don't meet ROI goals include:

- High upfront costs to introduce an integration broker product into an organization
- High personnel costs due to the low availability of skills (which is mostly a byproduct of the proprietary nature of this technology)
- Difficulties in establishing an efficient reuse strategy across different application domains.

All this is compounded by organizational challenges in outlining a clear strategy for managing the launch of the technology and selecting a methodology governing development processes that spans multiple application teams in a large organization.

How to Choose the Right Approach

A discussion of all the factors that could enter into the decision-making process in selecting the right approach for integration is beyond the scope of this article, and there are no black and white guidelines to follow. The remainder of this article identifies some broad architectural concerns your organization may want to consider.

From an architecture perspective, many integration projects come in two basic flavors: business event propagation and aggregation of business functionality. The first architecture applies to a scenario that consists of either a data-sharing solution or a multistep business process. In both cases, events originate in one application and are propagated to other applications to eliminate manual processes and reduce latency until the applications have a consistent view of where the business is.

While sharing data across applications could also be achieved using database-centric technology (e.g., replication tools or ETL), the requirements for many event propagation scenarios dictate that a heterogeneous mix of data stores and applications need to be involved in the information exchange. In fact, there are many cases when it's undesirable to access data stores directly, but better to send a business request to the application that governs the data and have it apply consistent updates to the data it owns.

A multistep business process adds some overarching business logic that controls how events are propagated. However, the fundamental assumption remains that data is simply forwarded across applications; there are no responses constructed and typically there's no end user waiting at a presentation tier to formulate an answer. The business event propagation architecture has been a typical domain for most integration broker projects. That makes it an attractive product choice compared to other alternatives, including application servers.

On the other hand, aggregation of business functionality is an architectural "super pattern" that's more suitable for application servers. Instead of simply forwarding data, applications cooperate to create an aggregated response to a request. This implies that we can't just "string" applications together through adapters and MOM systems. We also need to implement a controller (i.e., some new business logic that manages the aggregation process). Finally, an aggregation of business functionality super pattern will most likely involve an end user, thus requiring tight integration with an appropriate presentation tier.

All this typically necessitates a standard-based container that's geared to manage, and most importantly, host newly developed business logic—such as what an EJB container does.

Application servers provide this kind of component-hosting environment—including standard interfaces between business logic components and a container that facilitates portability across products from different vendors—as well as support for a wide range of automatic, container-managed services. These include services such as distributed transactions, location, instantiation, and persistence, as well as integration between the container and front-end pieces that collectively represent the presentation tier.

Conclusion

In summary, many factors, including what we call "money, religion, and politics," will enter into your decision-making process for selecting the right technology for integration projects. This decision is becoming even harder as application server platforms become more sophisticated and offer more integration functionality. A shining example of this is WebLogic Integration Platform 8.1, which is not only built on the new WebLogic Server 8.1 stack, but also provides a serious attempt to unify development and integration as a single discipline. The product also provides a plethora of new integration functionality that, until very recently, was available only via integration brokers.

To that end, intelligent organizations will continue to follow the evolution of these two distinct technologies. If their convergence continues, the word "distinct" may no longer apply. Stay tuned. **bij**

About the Authors

Max Dolgicer is an internationally recognized expert and managing director of International Systems Group (ISG), Inc., a leading consulting firm that specializes in large-scale integration projects, using a variety of middleware technologies. He has managed and played key technical roles in many of ISG's major engagements for *Fortune 500* clients.
e-Mail: mdolgicer@isg-inc.com
Website: www.isg-inc.com

Gerhard Bayer is a senior integration consultant with ISG.
e-Mail: gbayer@isg-inc.com
Website: www.isg-inc.com