

2.0 The Mission and Future of Integration

Application integration is pervasive because it is required, not because it is easy. Although expensive and difficult, application integration is undertaken because an enterprise has no choice. Forces outside of the IS organization usually motivate the enterprise to integrate applications. Business reasons for application integration include:

- Regulatory compliance — for example, compliance with the Basel accords in Europe, or with Sarbanes-Oxley or the Health Insurance Portability and Accountability Act in the United States
- B2B collaboration — for example, via electronic data interchange (EDI) or UCCnet
- Mergers and acquisitions
- Deploying new packaged applications
- Desire for a single view of customer
- Implementing self-service portals — for customers or employees
- Improving data quality

In 1998, the most-common business factor motivating integration was installing a packaged application, such as enterprise resource planning (ERP). In 2000, it was implementing customer relationship management (CRM). In 2004, it's industry-specific regulation.

Enterprises usually don't undertake integration projects to:

- Clean up messy and inefficient processes in the IS organization
- Consolidate disparate middleware products into a few strategic choices
- Improve documentation
- Provide flexibility for future application changes
- Conform to emerging IT standards
- Use the latest technology
- Reduce costs in the IS organization (usually by eliminating the need for some IS staff)

Even if they believe that integration technology and systematic integration design practices can bring these benefits (which they can), most IS managers won't — or can't — invest today to reap these largely unquantifiable benefits years into the future. As a result, although enterprises often undertake integration projects, they usually do so on a piecemeal basis and only in response to urgent business needs.

For most enterprises, the crucial issue is how to integrate, not whether to integrate. Although most modern business strategies require effective application integration, little consensus exists about how best to achieve it. Most enterprises are still learning about the design patterns, software tools and organizational practices that are most effective for application integration.

Furthermore, software vendors continue to make fundamental changes in the design of their integration middleware products as they learn more about the nature of the integration problem. This further complicates the task of enterprise architects, who must deal with shifting technology as they gain experience with systematic integration through trial and error.

This chapter addresses the key factors associated with meeting the application integration challenge today, and how this challenge will evolve in the future. The following Key Issues frame the analysis in this chapter:

- What changes will service-oriented architectures (SOAs) and event-driven architectures (EDAs) bring to the design practices used for application integration?
- How will integration software technology evolve?
- How will breakthroughs in standards, design concepts and organizations affect the cost and complexity of application integration?

2.1 How SOAs and EDAs Will Affect Application Integration

Key Issue: What changes will SOAs and EDAs bring to the design practices used for application integration?

2.1.1 The Nature of Application Integration

Tactical Guideline: *To meet the requirements of modern business, an enterprise needs:*

- An application integration strategy
- An application integration team or competency center
- A comprehensive middleware infrastructure

Application integration aims to make independently designed application systems work together. This broad definition encompasses tightly coupled request-reply exchanges among interdependent systems as well as simple, arms-length, batch file transfers among relatively autonomous application systems.

IS organizations find integration daunting and difficult because application developers always have to reconcile disparate information architectures based on different data, process and object models. Moreover, in most cases, developers must make different technologies interoperate, including:

- Operating systems
- Programming languages
- Application platforms
- Database management systems (DBMSs)
- Other middleware

Today, stand-alone applications — that is, applications that don't interact or communicate with other application systems — are almost nonexistent. Virtually every application system links to one or many other systems — even if only using simple batch file transfers.

Therefore, every application development (AD) project also is, in part, an integration project. Conversely, because every project includes some new development of logic and data, and stems from a desire to improve the way the business runs, no "integration" project deals solely with application integration.

2.1.2 The Three Types of Relationships Among Enterprise Applications

Application integration involves one or more of three relationship patterns (see Figure 2-1):

- *Data consistency* integration aims to get multiple application systems — often controlled by different business units — to agree on facts. This goal is an issue when two or more databases hold information on the same subject.
- *Multistep process* integration involves a sequence of related activities, each conducted by an application system or person.
- *Composite applications* combine the logic or data from two or more application systems — where the combined systems are independently designed, and heterogeneous in their information architecture — to support the work of one instance in a business process (such as a purchase order or insurance claim). Composite-application integration represents the most closely knit and hardest-to-implement of the three integration patterns.

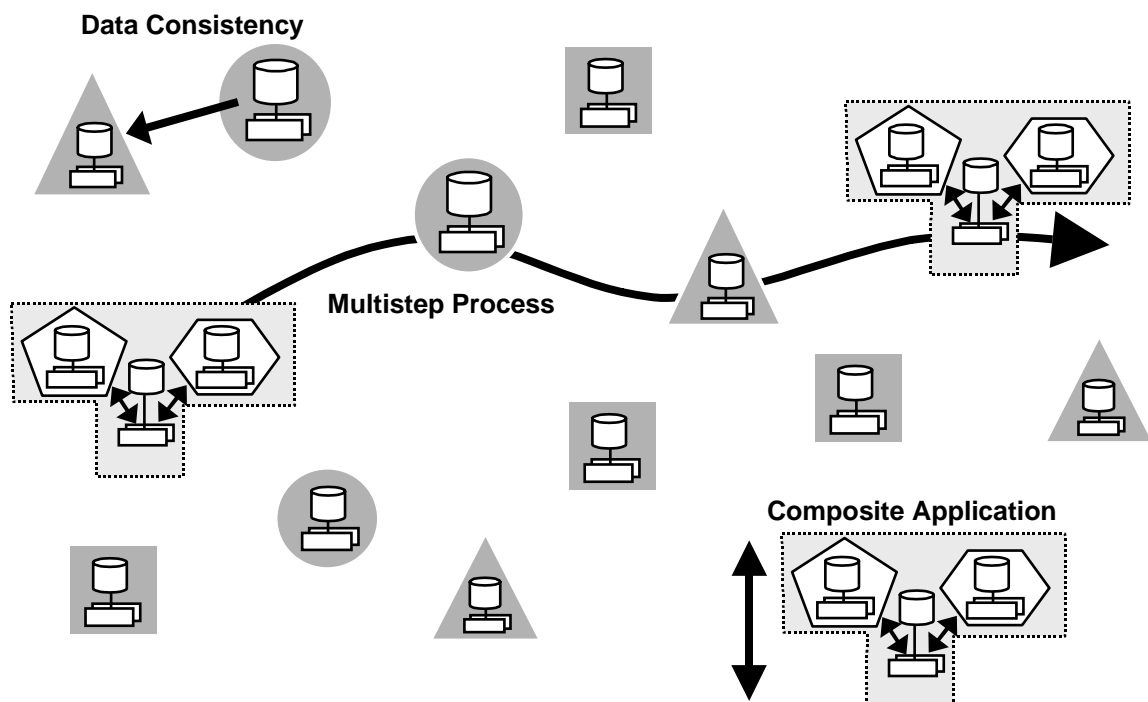
Action Item: Gain familiarity with the three basic integration patterns, and be prepared to implement all of them to address the needs of different parts of the business process. Real-world business problems will require using a mix of all three patterns — no single pattern is right or wrong.

2.1.3 Multistep Processes and Business Process Management

Tactical Guideline: Modern integration strategies should aim to optimize the end-to-end flow of a multistep business process.

Multistep processes aren't new — business operations used them even before the introduction of computers. The idea of thinking and designing “straight through” across multiple business units and their respective application systems isn't new, either. Business process re-engineering has tried to accomplish that since the late 1980s.

Figure 2-1: Enterprises Integrate Applications Using Three Kinds of Relationships



Source: Gartner

However, business managers, analysts and software engineers implement end-to-end process optimization more often today, and the relevant software tools have improved considerably. Enterprises often use business process management (BPM) engines to automate the business rules that govern the handoffs among systems, thereby reducing errors and making the flow of data more flexible and easily reconfigured (see Figure 2-2).

As business process modeling tools become more integrated with BPM engines, developers don't have to manually transfer the process design from their design and simulation tools to the runtime BPM engine. Furthermore, workflow-type BPM tools are automating human activity steps — particularly for exception and approval handling — on a broader scale than ever before. (For more information on BPM technologies, see Chapter 5.0.)

Finally, the use of new business-activity monitoring (BAM) features in BPM products, as well as in special-purpose process monitoring products (from vendors such as Bristol Technology, MQSoftware, Nastel Technologies, Network Managers & Integrators and Systar), has improved process monitoring.

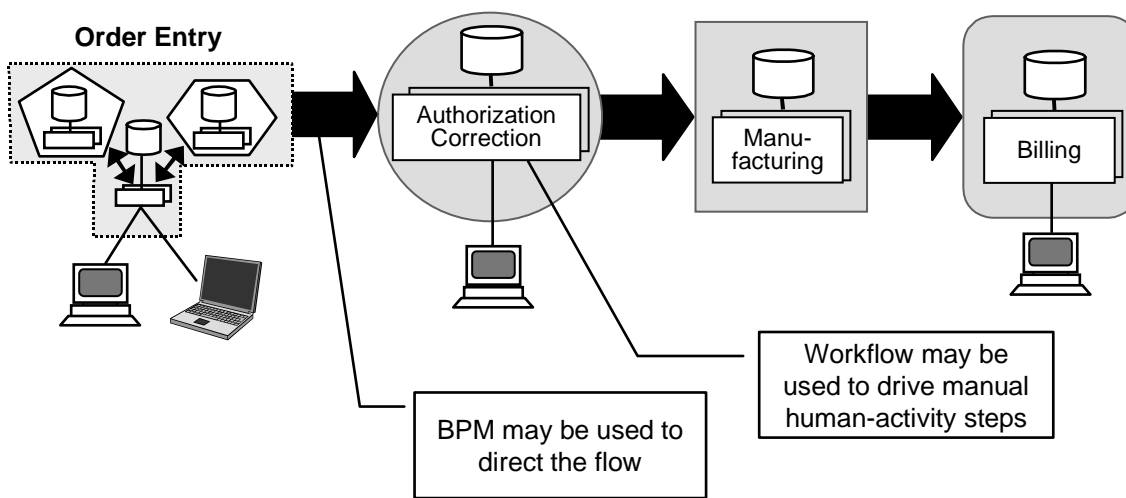
Action Item: To ensure that the process design correctly responds to business priorities and metrics, follow a development process in which business analysts and software engineers work together from the earliest design stages.

2.1.4 Composite Applications

Tactical Guideline: Composite applications differ significantly from multistep processes because they integrate vertically among the tiers within one step of a business process, rather than horizontally across multiple steps.

To create a composite application, a developer must implement some new custom code — whether written in a third-generation language (3GL), or generated by a fourth-generation language (4GL) or modeling tool — to complement the legacy or purchased applications that provide back-end services. The legacy or purchased applications generally continue to service established end users directly through their own presentation tiers, while servicing the new end users of the composite application.

Figure 2-2: Multistep End-to-End Process Example



Source: Gartner

BPM business process management

In some cases, a microflow BPM engine is used to orchestrate the dialogue among the applications within the composite application. Microflow differs from long-running BPM or workflow services because the interaction is short — typically lasting only a few seconds.

One can implement the interactions between the new logic and the legacy or packaged code at one or more of several tiers (see Figure 2-3). If the connection goes:

- Into the presentation tier, one may use screen scrapers or integration servers
- Into a business logic or data logic tier, one may use:
 - Remote procedure calls (RPCs)
 - Object request brokers (ORBs)
 - Message-oriented middleware (MOM)
 - Web services
- Directly to the DBMS used by the legacy or purchased package, one may use:
 - Open Database Connectivity

- A database gateway
- An enterprise information integration (EII) tool

(For more information on composite applications, see Chapter 6.0)

Action Item: When implementing composite-application relationships, ensure that architects and developers use different design concepts and tools than the ones they use for multistep-application relationships.

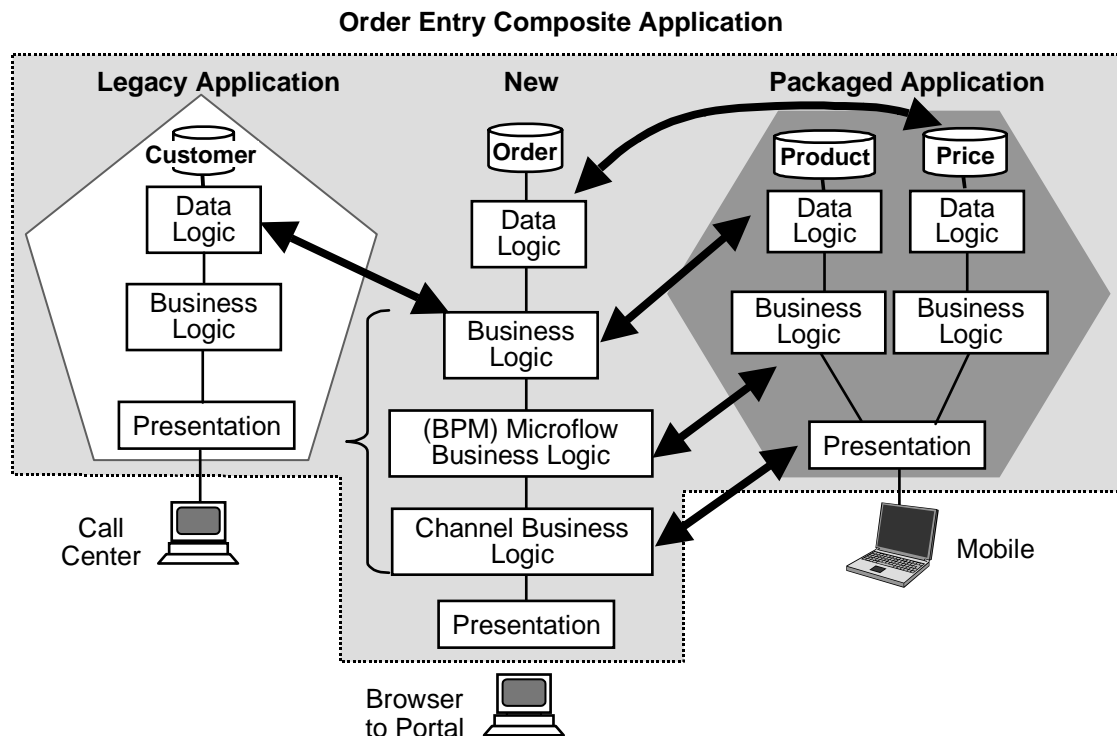
2.1.5 Using a Service-Oriented Architecture

Tactical Guideline: Applications designed from conception using an SOA can be incorporated into a composite application more easily than can monolithically designed (non-SOA) applications.

Two main principles are essential in an SOA:

- Modularity — breaking big jobs into smaller tasks

Figure 2-3: Linking Composite Applications Vertically Across Logic Tiers



Source: Gartner

BPM business process management

- Encapsulation — using a clearly defined interface to insulate the internal parts of a component from outside contact

These two principles make SOA applications easier to develop and maintain than traditional monolithic applications, because developers focus on their respective parts of the application without having to know the details of work done by others. With minimal coordination, developers using an SOA can create, test and modify components — as long as developers maintain the same application programming interface (API) across all components (that is, respect the interface contracts).

An SOA clarifies design and facilitates reuse — even within an application system in which all clients and servers are new and written by one development team. An SOA is even more helpful when developing a composite application that will combine parts of different systems (see Figure 2-4). Developers should identify logically

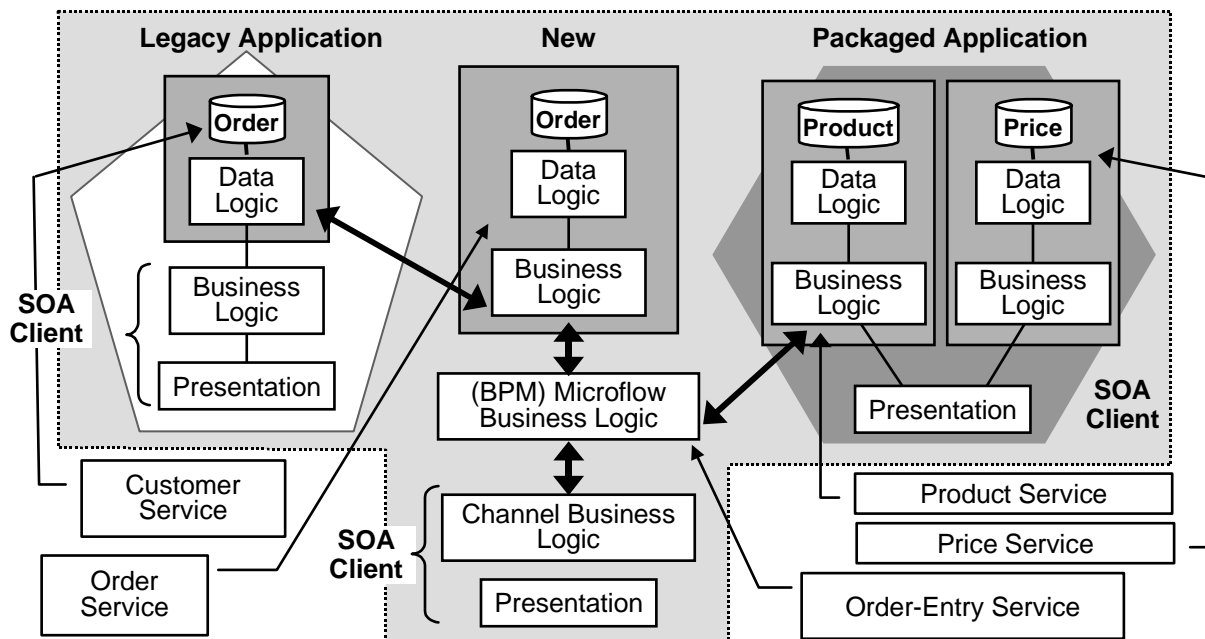
reusable functions and package them as server modules with documented interfaces.

Generally, an SOA should use Extensible Markup Language (XML) as the interface encoding language. Most SOA applications and wrappers developed through mid-2009 will use Web Services Description Language (WSDL) to document the interfaces. Developers can use Simple Object Access Protocol (SOAP) to convey a request message and the response in cases where SOAP meets quality-of-service requirements.

However, an SOA isn't the universal remedy for all integration challenges because it doesn't provide good support for data consistency or multistep processes, which constitute the majority of integration scenarios.

Action Item: When developing new application systems that include components that may act as servers in future composite applications, design them from the outset to be “composable” (that is, build to integrate).

Figure 2-4: Using an SOA to Organize Application Logic



Source: Gartner

BPM business process management
SOA service-oriented architecture

2.1.6 The Role of Event-Driven Architecture

Tactical Guideline: Although compatible, EDA and SOA are distinct concepts, each with its own advantages and limitations. Enterprises need both.

As mainstream enterprises adopt SOAs on a broad scale, they also see the need for another design approach: EDA (see Figure 2-5).

SOA and EDA have many things in common because they both represent ways to combine multiple software components into large distributed applications. However, they structure relationships among components differently, which makes them appropriate for different purposes.

In simplistic terms, SOA uses directed, generally bidirectional, request-response communication to delegate work to a subordinate procedure. By contrast, EDA uses unidirectional messaging to communicate

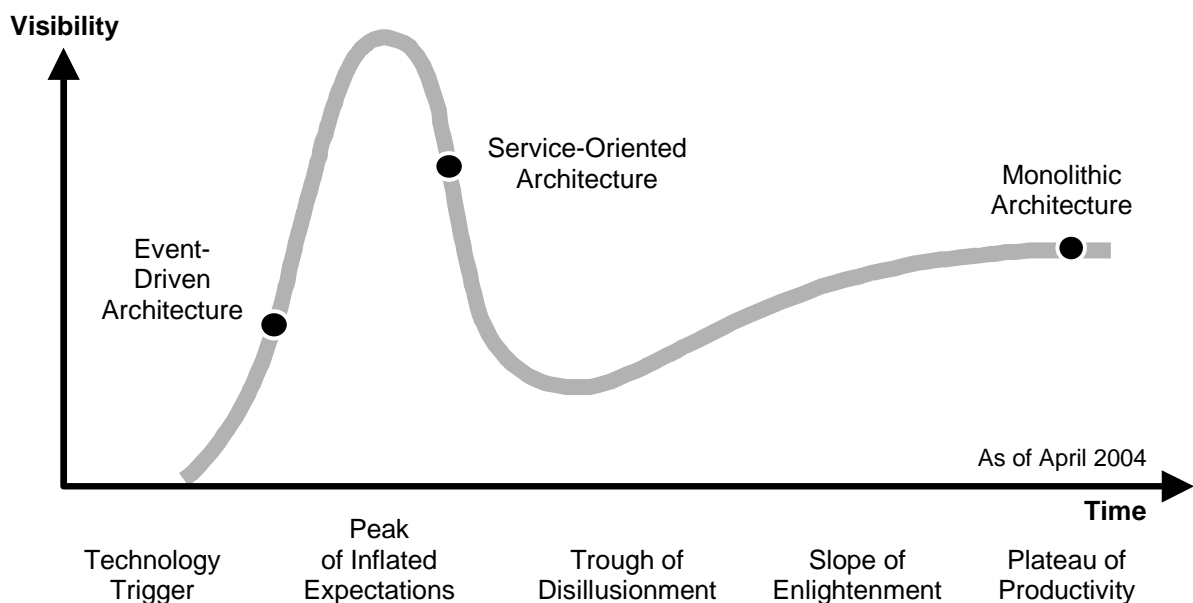
between two (or among more) largely independent peer procedures. Applications often combine SOA and EDA constructs.

The principles of event-driven programming have been known for decades and applied at a technical level within system software and graphical applications. Except in a few specialized areas, however, complex event processing hasn't been used as a design concept for application systems.

EDA represents the next "big thing" because it provides unprecedented power to support dynamic and multifaceted business processes. EDA will complement, not replace, SOA. Although architects can implement both types of architecture using Web services, neither EDA nor SOA requires the use of Web services.

Action Item: Because enterprise agility will depend on SOA and EDA, ensure that IS managers and architects understand the strengths and limitations of these architectures.

Figure 2-5: The Evolution of Architecture in Business Applications



Source: Gartner

2.2 Expected Changes in Integration Software

Key Issue: How will integration software technology evolve?

2.2.1 Enterprise Service Buses

Strategic Planning Assumption: More than one-half of large enterprises will use enterprise service buses (ESBs) by the end of 2006 (0.7 probability).

Raw communication protocols (such as TCP/IP, SMTP, SOAP and FTP) and traditional middleware products (such as RPCs, ORBs, SOAP stacks, traditional application servers and MOM) lack features that SOAs and event-driven applications need. Vendors have responded by developing a new form of middleware — ESBs (see Figure 2-6).

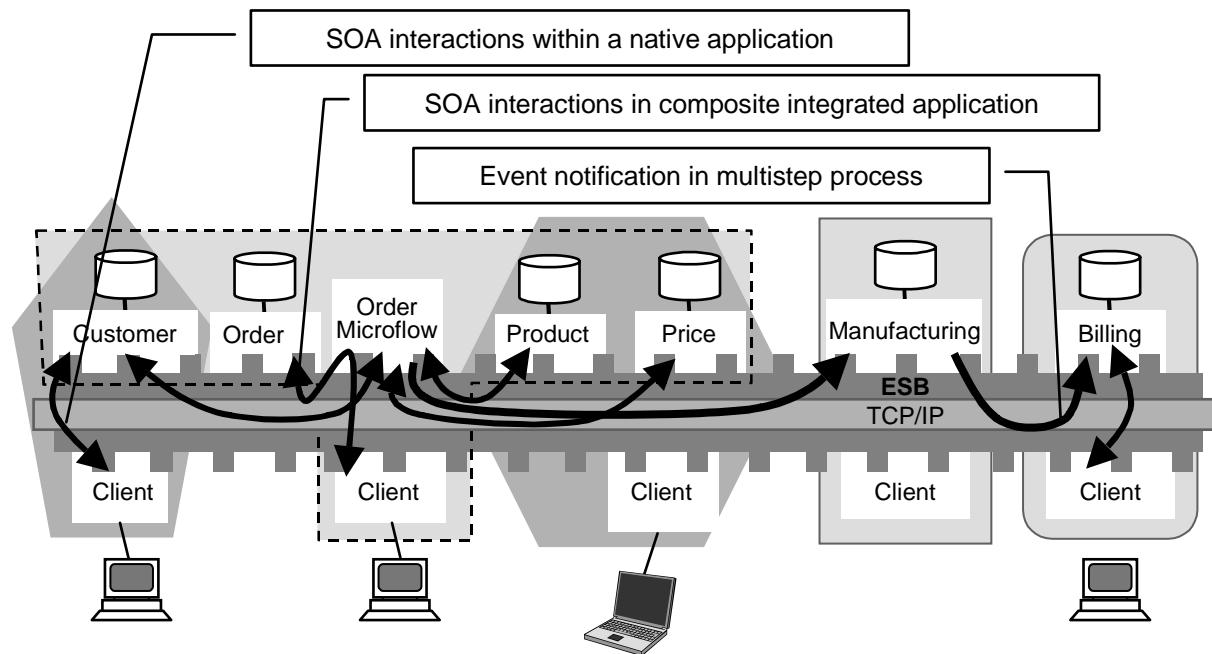
ESBs — which will operate within a single application system as well as among multiple, separate application systems — support all of the following features:

- Web services standards
- Messaging
- Basic transformation
- Intelligent routing
- Interface repositories
- Associated development and management facilities

Although vendors sometimes will sell ESBs as independent products, they usually will bundle ESBs into:

- Application servers
- Application platform suites (APSs)

Figure 2-6: SOA and Event-Driven Applications Will Use ESBs as Middleware



Source: Gartner

ESB enterprise service bus
SOA service-oriented architecture
TCP/IP Transmission Control Protocol/Internet Protocol

- Operating systems
- Integration suites

Before the emergence of commercial ESB products, enterprises met this functional need through DCE (Distributed Computing Environment), CORBA (Common Object Request Broker Architecture) or COM (Common Object Model) — or they constructed a rudimentary version of an ESB by using MOM and writing a super-API layer to provide additional functions. However, the drawbacks associated with these approaches limited their long-term acceptance.

Now, ESBs can support “pluggable” applications better than ORBs, MOM and other previous middleware. Using an ESB, an enterprise can add, delete, move or modify components of a composite application or a multistep process dynamically — without stopping processing.

For the first time, the industry has come to a consensus on a set of specifications that standardize a small part of these functions. ESBs support Web services by implementing SOAP and various aspects of WSDL and UDDI (Universal Description, Discovery and Integration).

Many ESBs offer communication mechanisms, such as guaranteed delivery and publish-and-subscribe, which will be added to Web services standards. ESB vendors that don't support these functions soon will do so.

ESBs also may provide other value-added services, such as:

- Message validation
- Sophisticated transformation
- Content-based routing
- Security
- Service registration and discovery
- Load balancing
- Failover
- Logging

Some services are built into the ESB core; others run as plug-in modules. ESBs support XML and other message formats.

Only a few leading-edge enterprises have deployed commercial ESB products; however, use of ESBs will accelerate quickly when large vendors — such as BEA Systems, IBM and Microsoft — roll out their ESBs during 2005 and 2006.

Action Item: If your enterprise is a mainstream (Type B) one, plan to incorporate ESBs into its IT strategies during 2004, and to selectively implement new applications on ESBs in 2005.

2.2.2 The Two Emerging Types of ESBs

Strategic Planning Assumptions:

- *By the first half of 2005, IBM will ship its WebSphere ESB product (0.8 probability).*
- *By 2006 (0.4 probability) or by the end of 2007 (0.8 probability), Microsoft will ship Indigo in the Longhorn operating system.*
- *By the second half of 2005, more than 90 percent of ESBs will support multiple styles of communication and multiple protocols (0.8 probability).*

Tactical Guideline: *To remain competitive, every major application server, APS, MOM and integration suite vendor must have a public ESB strategy by the end of 2004 and an ESB available by the end of 2006.*

In 2004, two types of ESBs exist:

- Those based on SOAP and Hypertext Transport Protocol (HTTP)
- Those that support Web services and other communication mechanisms

SOAP/HTTP ESBs — sometimes called Web services brokers, Web services management tools or “fabrics” — originally worked only with SOAP and HTTP. Although they still use SOAP, some also support transports other than HTTP. These ESBs intercept messages within the local SOAP runtime environment or in an intermediate proxy server to provide value-added services. Many vendors offering these products also provide Web services development tools and management capabilities.

Examples of SOAP/HTTP-centric ESBs include:

- Blue Titan Software's Network Director
- Cape Clear Software's 4 Server
- Digital Evolution's DE Management Server
- Hewlett-Packard's Talking Blocks SOA
- Systinet's Server and Gateway
- webMethods' Fabric

Multiprotocol ESBs support SOAP/HTTP as well as additional protocols, and usually follow the Java Message Service standard for areas that don't yet have a Web services standard. These products also provide various value-added services. Examples of Multiprotocol ESBs include:

- BEA Systems' X-Bus (future)
- Fiorano Software's ESB
- IBM's WebSphere ESB (future)
- Iona Technologies' Artix
- Kenamea's Web Messaging Platform
- KnowNow's Event Routing Platform
- Microsoft's Indigo (future)
- PolarLake's Jintegrator
- SeeBeyond's elnsight ESB
- Software AG's EntireX
- Sonic Software's ESB
- SpiritSoft's Spiritwave
- Tibco Software's ESB (future)
- WebV2's Process Coupler

By 2006, the two types of ESBs will largely converge for the following reasons:

- Web services standards have become more open to protocols beyond SOAP and HTTP.
- Standard SOAP documents can be sent over private protocols when HTTP isn't scalable, reliable or efficient enough.

- WSDL can describe official Web services, even if SOAP isn't used, which enables pure SOAP ESB vendors to implement other protocols without violating their principle of standards compliance.

2.2.3 Supplementing ESBs With Integration Suites

Strategic Planning Assumption: *By the end of 2006, all major integration suites will have an ESB core (0.7 probability).*

Although an enterprise can use a plain ESB to implement a new, basic SOA or event-driven application, the ESB doesn't include all of the features that more-sophisticated applications or application integration scenarios require. Therefore, most large middleware vendors will offer some or all of the features of a full integration suite as an add-on to their ESBs. These features may be bundled or sold separately.

An integration suite — a superset of MOM or an ESB — includes features that go beyond those found in a basic ESB, such as:

- A BPM engine
- B2B features, such as electronic data interchange (EDI) and trading-partner management
- Adapters and adapter-building tools
- Sophisticated transformation services
- Portals
- Packaged integrating processes
- BAM
- Complex event processing

Most integration suites originally used MOM as their backbone because Web services weren't mature enough when the suites were designed. Although these integration suites support Web services, they generally use them as a supporting system, rather than as the core communication infrastructure. Although vendors are incrementally redesigning their integration suites to incorporate Web services and other ESB features in a more-native fashion, the degree of integration varies significantly among vendors.

By the end of 2006, most integration suites will have an ESB core (see Figure 2-7). Enterprises will usually have the option to buy the ESB separately or have it bundled with the suite.

Action Item: To detect the subtle but important differences in product strengths and limitations, examine ESB and integration suite architectures at a detailed level.

2.2.4 The Use of Integration Suites

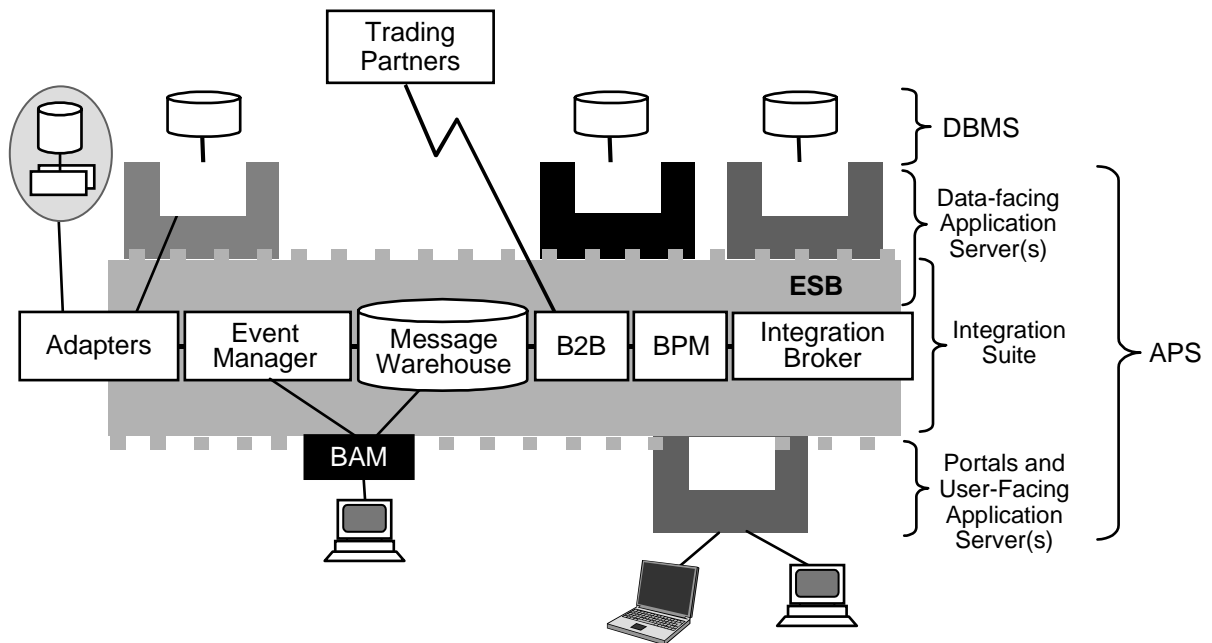
Strategic Planning Assumption: By the end of 2008, large enterprises will use integration suites in approximately 40 percent of their AD projects — up from 10 percent of their AD projects in the first half of 2004 (0.7 probability).

Nearly 90 percent of application integration is done without involving an integration suite, although this is changing (see Figure 2-8).

More than 85 percent of enterprises with more than \$1 billion in revenue use an integration suite in at least one application project. However, these suites are used in only about 10 percent of AD projects overall — most projects (60 percent) still use only point solutions for integration, such as:

- ESBs
- DBMS gateways
- Adapter tools
- EDI tools
- Screen scrapers
- Integration servers
- Transformation engines
- Extraction, transformation and loading (ETL) tools
- EII tools

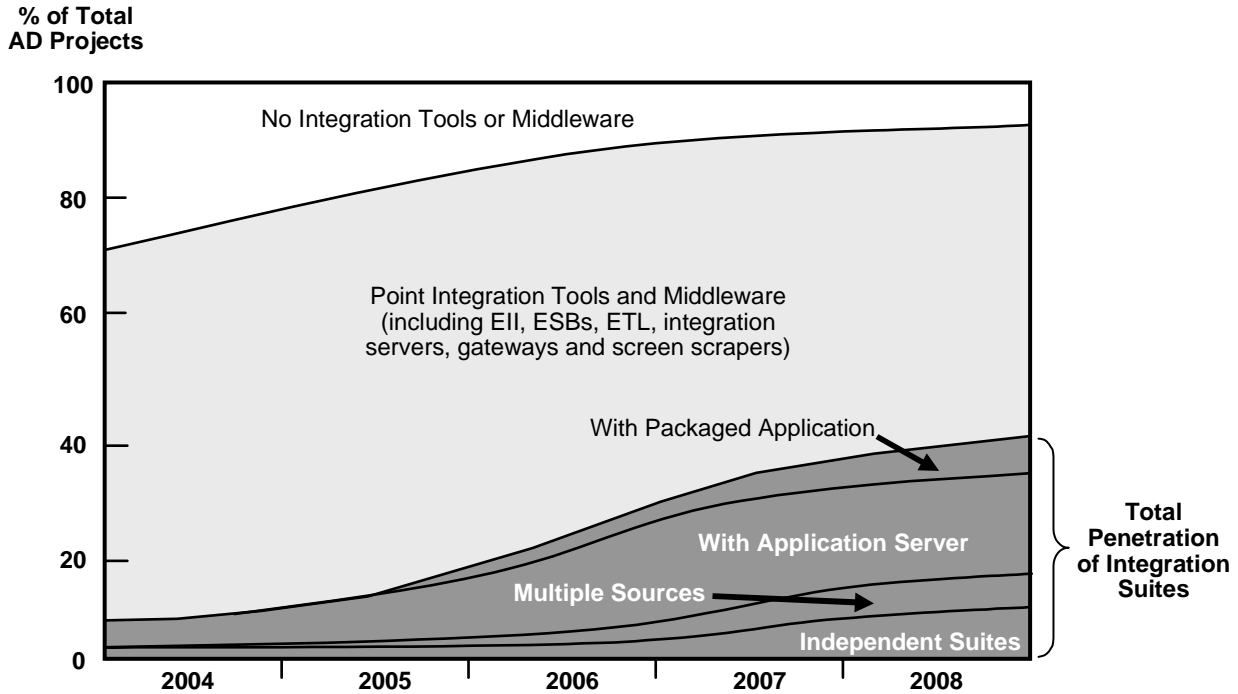
Figure 2-7: Next-Generation Integration Suites Will Have an ESB Core



Source: Gartner

APS	application platform suite	BPM	business process management
B2B	business-to-business	DBMS	database management system
BAM	business activity monitoring	ESB	enterprise service bus

Figure 2-8: Integration Technology Use in AD Projects, 2004 to 2008



Source: Gartner

AD application development **ESB** enterprise service bus
EII enterprise information integration **ETL** extraction, transformation and loading

- Transaction delivery networks

The remaining 30 percent of projects use no integration-specific technology, relying on:

- Basic programming languages
- Simple forms of middleware, such as application servers and MOM
- Traditional IT tools, such as file transfer utilities

Because they're embedded in packaged applications and APSs — as well as sold separately — integration suites are proliferating. Enterprises often buy the integration suite from the vendor that supplies the application server or APS — for example, from BEA, IBM, InterSystems, Microsoft or Oracle. Use of such vendor-supplied suites accounts for approximately 6 percent of all projects in 2004, and will likely increase to more than 20 percent in 2008.

Many projects use a mix of point solutions and suites. Although products from integration specialists — such as Fiorano Software, SeeBeyond Technology, Sonic, Tibco, webMethods and Vitria — are used in approximately 4 percent of projects in 2004, this number will increase to more than 10 percent in 2008.

Action Item: Ensure that enterprise's architecture team and integration competency center develop guidelines for when to use each kind of integration technology.

2.2.5 Development- and Deployment-Centric Integration

There are two main approaches to integration — development-centric and deployment-centric. The following two sections examine the characteristics, advantages and drawbacks of each.

2.2.5.1 The Development-Centric Integration Approach

Development-centric integration:

- Combines software development and application integration as tightly as possible
- Designs new server-based application logic using an SOA (generally)
- Tends to use one vendor's tools for the new code and the integration (often packaged as an APS, which includes the development tools, application server, portal, ESB and BPM tools)
- Coordinates the metadata for development and integration
- Uses one set of tools for monitoring and management

This strategy works best in a homogeneous environment in which most or all of the new applications and services will use the same APS. Legacy and packaged applications can be connected to the new APS-based application using adapters.

This approach is suited for building composite applications — which, by nature, require a significant amount of new logic. Development-centric integration delegates many of the data consistency and multistep-processing interactions to other tools, such as those for third-party file transfers or ETL.

Development-centric integration doesn't work well when the AD groups are autonomous and use different software tools. Similarly, it offers diminished benefits when an enterprise will buy its applications — not build them — because of the diverse software technology typically associated with multiple purchased applications. In the worst case, developers will face the complex challenge of combining multiple application servers, development tools, BPM engines, ESBs and other integration tools into a single composite application.

Action Item: Consider development-centric integration strategies when:

- The enterprise is likely to create a high amount of new custom code.

- Most of the development teams will use the same software tools for development and the same application servers.

2.2.5.2 The Deployment-Centric Integration Approach

Strategic Planning Assumption: *By the end of 2005, more than one-half of very large enterprises will have three or more integration suites in production (0.8 probability).*

Deployment-centric integration:

- Separates the development of new applications from the development of integration
- Focuses on diverse IT environments in which applications use a broad mix of application servers and development tools — conditions that make it nearly impossible to:
 - Coordinate all development metadata in one repository
 - Have one set of management tools
- Uses Web services tools, MOM, ESBs and integration suites that are specifically designed to be neutral with respect to different application servers and development tools
- Addresses data consistency, multistep-process relationships, and composite applications involving multiple platforms
- Focuses on the ongoing maintenance and reconfiguration of the application (after code is written) rather than just on the initial AD

Deployment-centric integration works best when an enterprise standardizes on one ESB and integration suite for a particular business process or business unit. Although the entire enterprise ideally would use the same backbone, such uniformity rarely happens.

This strategy doesn't work well when most applications use new custom code and all developers use the same platform. In such conditions, bringing in an ESB or integration suite that is not part of the application server unnecessarily complicates things.

Action Item: Use deployment-centric integration strategies when:

- Most applications are purchased or legacy systems.
- The enterprise uses different application servers.

2.2.6 Data Integration

Tactical Guideline: A successful application integration strategy must include data integration tools and data-oriented design patterns to complement message-based communication.

ESBs, Web services, MOM, RPCs, ORBs, sockets and other message-based communication mechanisms account for less than one-third of the data moved for integration purposes. Data integration accounts for approximately two-thirds of all application integration (measured in bytes) — especially when one considers the major role file transfers play (with or without integration brokers, EDI engines or ETL tools).

Enterprises commonly use ETL tools to load information into a data warehouse or data mart. However, they also can load information into operational data stores (ODSs) or application databases. Most enterprises will need both ETL tools and integration brokers because they serve different purposes.

There are two data-centric ways (and multiple message-centric ways) to achieve the concept of “a single view of the customer” (or a single view of anything):

- The first data-centric strategy is to create a physical ODS, an approach that Ascential Software and SAP call “master data management” and Oracle calls a “single source of truth.”
- The second approach is to create a virtual view of federated data at runtime, a strategy that some vendors call “EII” (although EII is really a goal rather than a technology, and should encompass all forms of data integration rather than just virtual data views). Most virtual federated views are used in read-only business intelligence applications; however, the converse is not true — most business intelligence applications are implemented using a physical data warehouse or data mart, rather than a virtual view of data.

(For more a more thorough examination of data integration approaches, see Chapter 9.0.)

Action Item: Recognize that a comprehensive integration strategy must incorporate batch and real-time communication mechanisms, and a mix of data-centric and message-centric tools and designs. No one size fits all.

2.3 The End of Application Integration: Often Predicted, Never Achieved

Key Issue: How will breakthroughs in standards, design concepts and organizations affect the cost and complexity of application integration?

Since the field of application integration emerged as a discipline in the mid-1990s, people have sought to find ways to eliminate the problem of integration. Although no approach has succeeded at eliminating the problem, major advances have been made toward making integration less costly and onerous. Integration isn’t impossible. However, it’s never really finished, either.

Web services don’t represent the end of application integration. Rather, they will accelerate the adoption of systematic integration strategies and tools by reducing the cost of connecting into end-point applications and by standardizing some aspects of the interactions.

When pundits proclaim that application integration is finished, this just means that integration products designed in the 1990s are obsolete and have been replaced by newer models — not that the need to make independently designed applications work together has disappeared or will ever disappear.

The following seven myths are typical of some of the common false claims concerning the demise of the need for integration, or for certain integration tools and technologies.

- **Myth No. 1:** Enterprise application architecture planning can prevent the need for integration. *Reality:* No enterprise can afford to rewrite all of its applications from scratch, so enterprises will always have to cope with heterogeneous application systems one way or another.

- **Myth No. 2:** Packaged application suites can eliminate the need for integration. *Reality:* Packaged software drives the need for more application integration. When a new package is installed, it must be linked with other packages and with legacy applications.
- **Myth No. 3:** Application servers will eliminate integration suites. *Reality:* Application server vendors see the need for integration suites and BPM engines, and they have added these products to their portfolios of offerings. Application servers do not compete against integration suites; rather, application server vendors selling their own integration suites compete against integration specialists who sell integration suites but not the rest of the software infrastructure.
- **Myth No. 4:** Web services will eliminate integration suites. *Reality:* Integration suites have been redesigned to use Web services to reduce costs and expand use.
- **Myth No. 5:** Web services will eliminate adapters. *Reality:* Adapters are still needed for transformation and process reconciliation, even when the applications have native support for Web services.
- **Myth No. 6:** Java Connector Architecture (JCA) will eliminate adapters. *Reality:* JCA is used in some adapters in an attempt to reduce cost, but Web services are becoming more widely used than JCA.
- **Myth No. 7:** Web services (SOAP/HTTP) will eliminate proprietary MOM. *Reality:* SOAP/MOM and WSDL/MOM will probably be used more than SOAP/HTTP, particularly in demanding applications.

All modern integration tools leverage standards such as XML, Web services and (often) Java, wherever practical. Although the basic requirements of application integration — such as semantic transformation, content-based

routing, publish-and-subscribe, BPM and workflow — haven't changed, development tools, metadata management, security, integration standards, administration and monitoring have substantially improved during the 2000-through-2004 period.

Most enterprises will find that their biggest challenge lies in finding cost-effective ways to deal with the volatility among vendors and the rapid pace of change in integration tools.

Action Item: Ensure that business leaders understand the benefits of systematic application integration strategies, and the business process improvements that can be enabled through the application of modern technology and standards.

2.4 Recommendations

- Recognize that, to meet modern business requirements, an enterprise needs:
 - An integration strategy
 - Integration software
 - An integration competency center (see Section 3.2.1)
- Apply BPM in selected multistep and composite applications.
- “Build to integrate” all new, enterprise-class business applications using SOA and basic forms of EDA.
- Prepare for EDA — it will be the next “big thing.”
- Add an ESB to the IT strategic plan and application architecture.