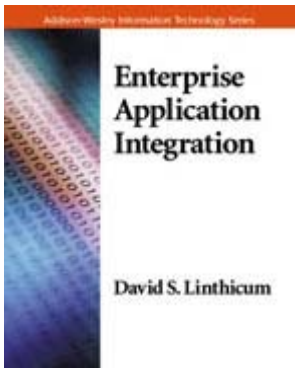


David S. Linthicum

Publisher: Addison Wesley

First Edition November 05, 1999

ISBN: 0-201-61583-5, 400 pages

Preface

Enterprise Application Integration, or EAI, is a buzzword that gives a name to the informal process that's been going on for years—the integration of various applications so that they may share information and processes freely. However, with a new, intense focus on EAI from both the vendor and analyst community, the opportunity finally exists to get ahead of the curve on a problem—the integration of applications—that costs the Fortune 1000 over $100 billion per year.

Forester Research estimates that more than 30 percent of IT dollars are currently being spent linking systems together for the common good. Perhaps it's time to consider whether that is too great a cost.

The EAI twist to this old story is not so much about system integration as it is about rethinking the technologies and approaches that would allow EAI to become a short-term and cost-effective reality. EAI also represents a technology-business philosophy that focuses on the business issues at hand and suggests that all systems existing either inside or outside an enterprise should be free to share information and logic in ways inconceivable in the past. EAI is the nexus of technology, method, philosophy, and desire to finally address years of architectural neglect.

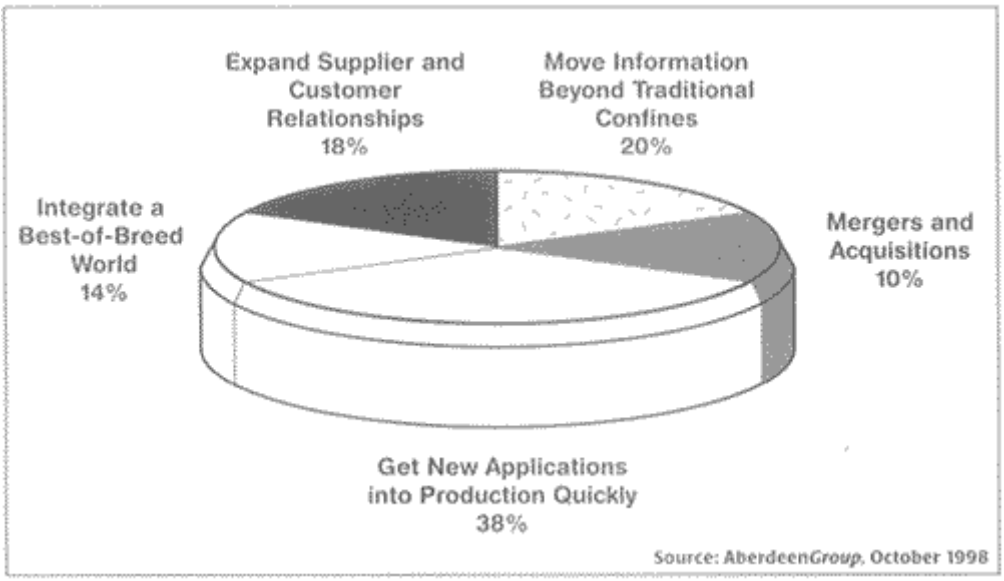
Make no mistake—this is an emerging space. Now is the time to ask the tough questions. Now is the time to determine the real value of applying EAI in an enterprise. It's time to question which technologies add value, and to determine which problems they solve. And now is the time to reflect on some of the architectural and design decisions that have been made over the years.

> They have computers, and they may have other weapons of mass destruction.
>
> —Janet Reno, Attorney General, February 2, 1998

As corporate dependence on technology has grown more complex and far reaching, the need for a method of integrating disparate applications into a unified set of business processes has emerged as a priority. After creating islands of automation through generations of technology, users and business managers are demanding that seamless bridges be built to join them. In effect, they are demanding that ways be found to bind these applications into a single, unified enterprise application. The development of Enterprise Application Integration (EAI), which allows many of the stovepipe applications that exist today to share both processes and data, allows us to finally answer this demand.

Interest in EAI is driven by a number of important factors. With the pressures of a competitive business environment moving IT management to shorter application life cycles, financial prudence demands that IT managers learn to use existing databases and application services rather than recreate the same business processes and data repositories over and over (see Figure 1.1). Ultimately, finances are a prime concern. The integration of applications to save precious development dollars creates a competitive edge for corporations who share application information either within the corporation or with trading partners.

Figure 1.1 The need for application integration



The vast majority of corporations use several generations of systems that rely on a broad range of enabling technology developed over many years. Mainframes, UNIX servers, NT servers, and even proprietary platforms whose names have been forgotten,

constitute the technological base for most enterprises. These technologies, new and old, are all providing some value in the enterprise, but their value is diminished if they are unable to leverage other enterprise applications. Moreover, the need to integrate those systems with packaged systems has been intensified by the popularity of packaged applications such as SAP, PeopleSoft, and Baan.

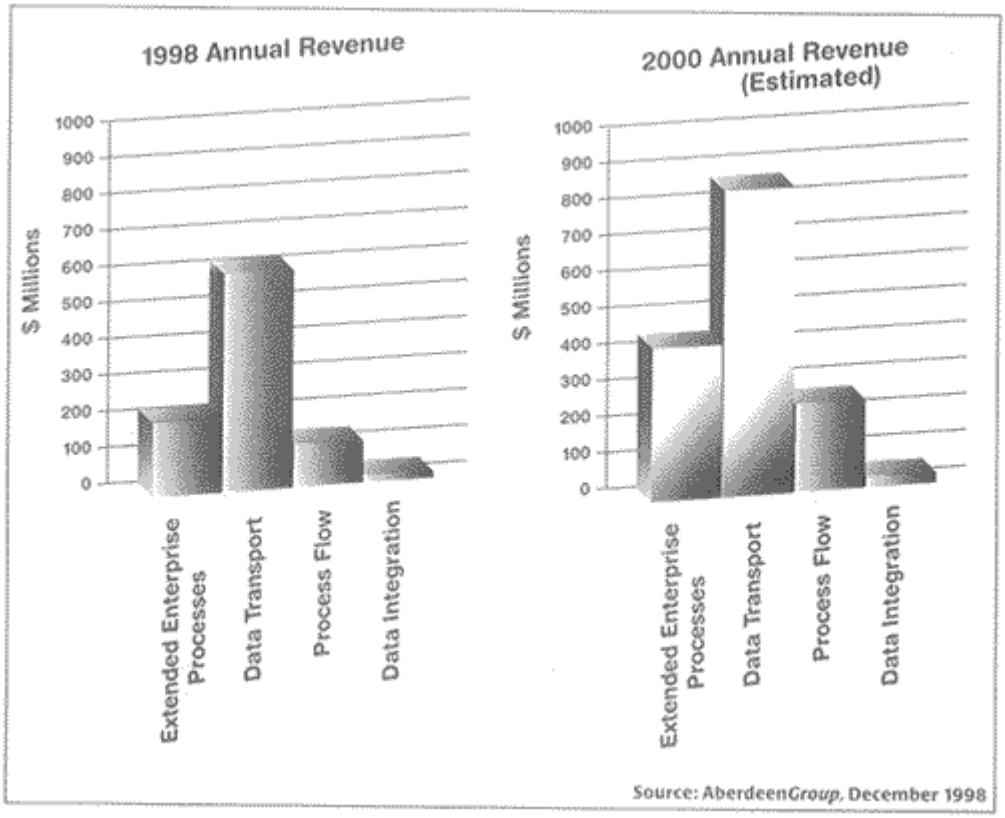
The case for EAI is clear and easy to define. *Accomplishing* EAI, however, is not. The idea of EAI is something we've been wrestling with over the last several years as the need for a comprehensive integration system has grown more urgent. Forester Research estimates that up to 35 percent of development time is devoted to creating interfaces and points of integration for applications and data sources. Most problems with developing software derive from attempts to integrate it with existing systems. Certainly that has been a significant problem in creating traditional client/server systems—what was inexpensive to build was expensive to integrate and became difficult to maintain.

What Is EAI?

So, if EAI is the solution, what exactly is it? EAI is not simply a buzzword dreamed up by the press and analyst community. It is, at its foundation, a response to decades of creating distributed monolithic, single-purpose applications leveraging a hodgepodge of platforms and development approaches. EAI represents the solution to a problem that has existed since applications first moved from central processors. Put briefly, EAI is the unrestricted sharing of data and business processes among *any* connected applications and data sources in the enterprise.

The demand of the enterprise is to share data and processes *without* having to make sweeping changes to the applications or data structures (see Figure 1.2). Only by creating a method of accomplishing this integration can EAI be both functional and cost effective.

Figure 1.2 The technology that drives EAI is growing quickly.



Now that you know what it is, the value of EAI should be obvious. EAI is the solution to the unanticipated outcome of generations of development undertaken without a central vision or strategy. For generations, systems have been built that have served a single purpose for a single set of users without sufficient thought to integrating these systems into larger systems and multiple applications.

Undoubtedly, a number of stovepipe systems are in your enterprise—for example, inventory control, sales automation, general ledger, and human resource systems. These systems typically were custom built with your specific needs in mind, utilizing the technology-of-the-day. Many used nonstandard data storage and application development technology.

While the technology has aged, the value of the applications to your enterprise likely remains fresh. Indeed, that "ancient" technology has probably remained critical to the workings of your enterprise. Unfortunately, many of these business-critical systems are difficult to adapt to allow them to communicate and share information with other, more advanced systems. While there always exists the option of replacing these older systems, the cost of doing so is generally prohibitive.

Packaged applications such as SAP, Baan, and PeopleSoft—which are natural stovepipes themselves—have only compounded the problem. Sharing information among these systems is particularly difficult because many of them were not designed to access anything outside their own proprietary technology.

Applying Technology

If EAI articulates the problem, then traditional middleware has sought to articulate the solution—sort of. Traditional middleware addresses the EAI problem in a limited manner.

The primary limitation is that middleware that uses **message queuing** or **remote procedure calls (RPCs)** only provides point-to-point solutions—that is, linkage between system A and system B. Unfortunately, any attempt to link additional systems quickly becomes a complex tangle of middleware links. Worse still, traditional middleware demands significant alterations to the source and target systems, embedding the middleware layer into the application or data store.

For example, when integrating a custom accounting system running on Windows 2000 with a custom inventory control system running on the mainframe, you may select a message-queuing middleware product to allow both systems to share information. In doing so, however, you generally have to alter the source system (where the information is coming from) with the target system (where the information is going to) in order to make use of the middleware. This is due to the fact that the point-to-point middleware layer only provides a program interface, and thus the programs must change to accommodate the middleware. This is costly and sometimes risky.

What's more, as we use the same or similar technology, as with the previous example, to integrate other applications inside an enterprise, the number of point-to-point solutions may grow to accommodate information movement between various systems. The end result is software pipes running in and out of existing enterprise systems, with no central control and central management, and a limited ability to react to change. The end state looks more like an ill-planned highway system that was built by many small integration projects but with little strategic value.

An additional complication to this scenario is that IT managers must perform integration projects inside fluid environments using rapidly advancing technology. In seeking to integrate links, the manager may also encounter additional problems such as:

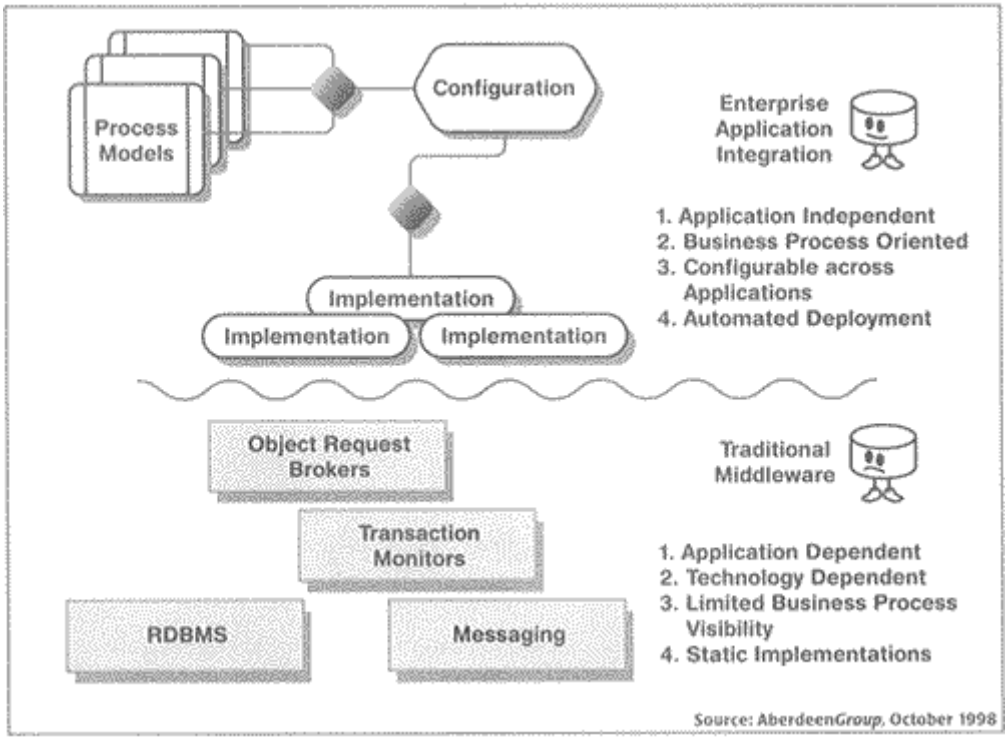
- A variety of legacy systems that contain mission-critical applications
- Several packaged applications with both proprietary and open frameworks
- A hodgepodge of hardware and operating system platforms
- A hodgepodge of communication protocols and networking equipment
- Geographically disbursed applications and databases

In addition to these structural limitations, the economics of traditional middleware have placed EAI out of the reach of most IT organizations. Even a simple dual-application linking is financially daunting, running as high as $10 million according to the Aberdeen Group.

Given these significant limitations, it follows that EAI represents a very different method of application integration than that using traditional middleware (see Figure 1.3). EAI provides a set of integration-level application semantics. Put another way, EAI creates a common way for both business processes and data to speak to one

another across applications. More importantly, we approach this old problem with a new set of technologies designed specifically for EAI.

Figure 1.3 EAI versus traditional middleware



So, keeping this information in mind, we can focus on the following differences between traditional approaches and the vision of EAI:

- EAI focuses on the integration of both business-level processes and data, whereas the traditional middleware approach is data oriented.
- EAI includes the notion of reuse as well as distribution of business processes and data.
- EAI allows users who understand very little about the details of the applications to integrate the applications.

How Did Things Get This Bad?

EAI answers the problem of integrating systems and applications. But how did the problem come about? The answer depends to a great extent on perspective. In the early days of computing, information was processed on centralized platforms. As a result, process and data existed in a homogeneous environment. Integrating applications (both processes and data) within the same machine rarely created a problem beyond some additional coding. As technology developed, platforms changed. Smaller and more open platforms, including UNIX and Windows NT (in addition to new programming paradigms such as object-oriented and component-based development), challenged traditional mainframes, once the backbone of IT.

Whether correctly or not, traditional IT leveraged the power of these newer, more open platforms. A number of factors contributed to this. Users wanted applications that were driven by newer and more attractive graphical user interfaces. Misreading or ignoring this desire, traditional legacy systems lagged in adopting "user-friendly" graphical user interfaces, abandoning the field to the newer platforms. Perhaps more importantly, the press trumpeted the claim that the cost of applying and maintaining the newer platforms was less than traditional systems. Sometimes this claim was correct. Often it was not.

In the rush to incorporate these new systems, most enterprises applied minimal architectural foresight to the selection of platforms and applications. IT managers made many of their decisions based on their perception of the current technology market. For example, when the UNIX platforms were popular in the early 1990s, UNIX was placed in many enterprises, regardless of how well it fit. Today, the same is true of Windows NT.

Clearly, the installation of these systems has been a textbook example of "management-by-magazine." Rather than making sound, business-driven decisions, or evaluating all possible solutions, decisions were made to implement the "coolest" technology.

While acknowledging that smaller and more open systems can hold a very important place in the enterprise, this decision-making process lacked the perspective and foresight that might have minimized, or possibly avoided, many integration problems. The success of these smaller, more open systems was that they met the requirement of commodity computing but with a steep price—the need for integration with the existing, older system.

The need for EAI is the direct result of this architectural foresight, or rather, the lack of it. Until recently, architecture at the enterprise level had been virtually nonexistent. Information technology decisions tended to be made at the department level, with each department selecting technology and solutions around its own needs and belief structures. For example, accounting may have built their information systems around a mainframe, while human resources leveraged the power of distributed objects, and research and development might have used distributed transaction processing technology. The end result could have been a mixing and matching of technologies and paradigms. As a result, the enterprise as a whole was left with a "system" that is nearly impossible to integrate without fundamental re-architecture—and a significant investment of precious funds.

To address the problem of architectural foresight, many organizations have created the role of the enterprise architect. This person or office is responsible for overseeing a centralized architecture and making sure that technology and solutions selected for the enterprise are functionally able to interact well with one another. As a result, departments will be able to share processes and data with a minimum of additional work.

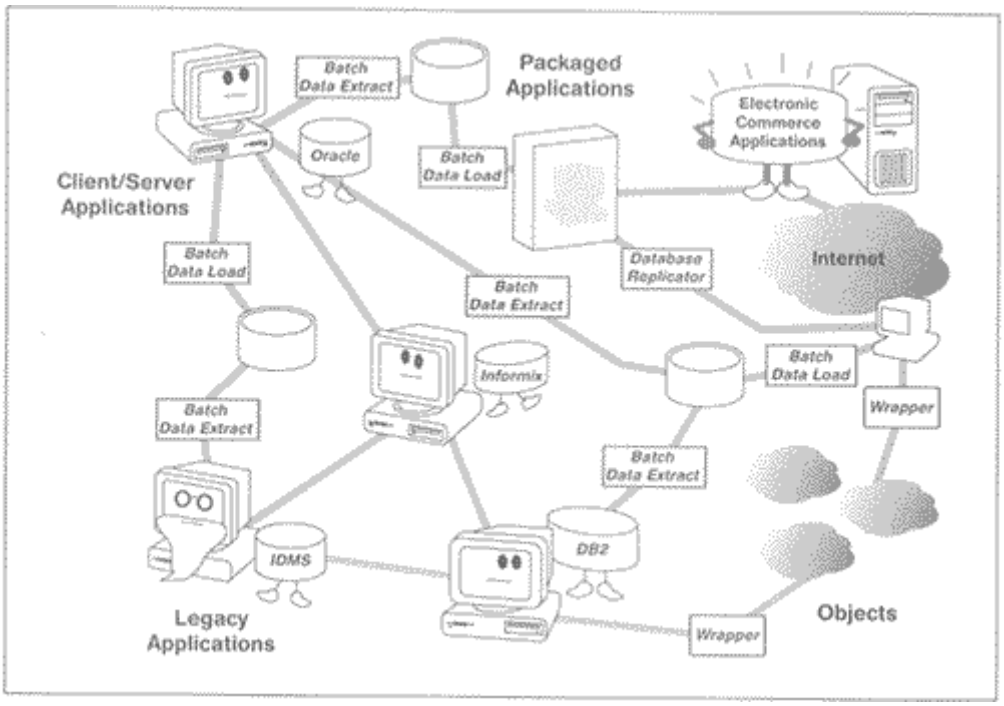
Enterprise architects will be called upon to make some unpopular decisions. They will be charged with making decisions regarding the commitment to a single programming paradigm and technology religion. If the enterprise goes to distributed objects, it must

do so as a unified whole. Tough, centralized decisions must be made. The idea of selecting technology for technology's sake is today a proposition that is simply too expensive. Corporate America is finally shutting down the playground that was once information technology.

Chaos Today, Order Tomorrow

The establishment of enterprise architects is a significant and positive development. Often, in addition to developing a central long-term strategy for the future, the role of the architect is to coordinate the technology that is already in place.
Most enterprises leverage many different technologies. The integration of these technologies is almost always a difficult and chaotic proposition. Traditional middleware technology, such as message-queuing software, ties applications together, but these "point-to-point" solutions create single links between many applications, as we mentioned previously. As a result, the integration solution itself may become more expensive to maintain than the applications it's connecting.
When using the point-to-point approach, integrating applications comes down to altering each application to be able to send and receive messages. This can be accomplished with any number of **message-oriented middleware (MOM)** products (e.g., IBM's MQSeries). While this is easily managed within the context of integrating two applications, integrating additional applications demands additional pipes. If you have successfully integrated application A with application B and would like to include applications C and D, you will have to create a pipe between each involved application. In no time, the process will grow so complex as to render it almost unmanageable. (See Figure 1.4.)

Figure 1.4 Enterprise chaos



This "solution" is absurd. However, traditional middleware leaves no other choice but to leverage this type of architecture.

Many enterprises that needed to integrate applications have implemented such integration architectures within their enterprises and continue to maintain them today. While it is easy to criticize these enterprises, they have done the best they could, given the available technology. Unfortunately, even their best efforts have resulted in chaos.

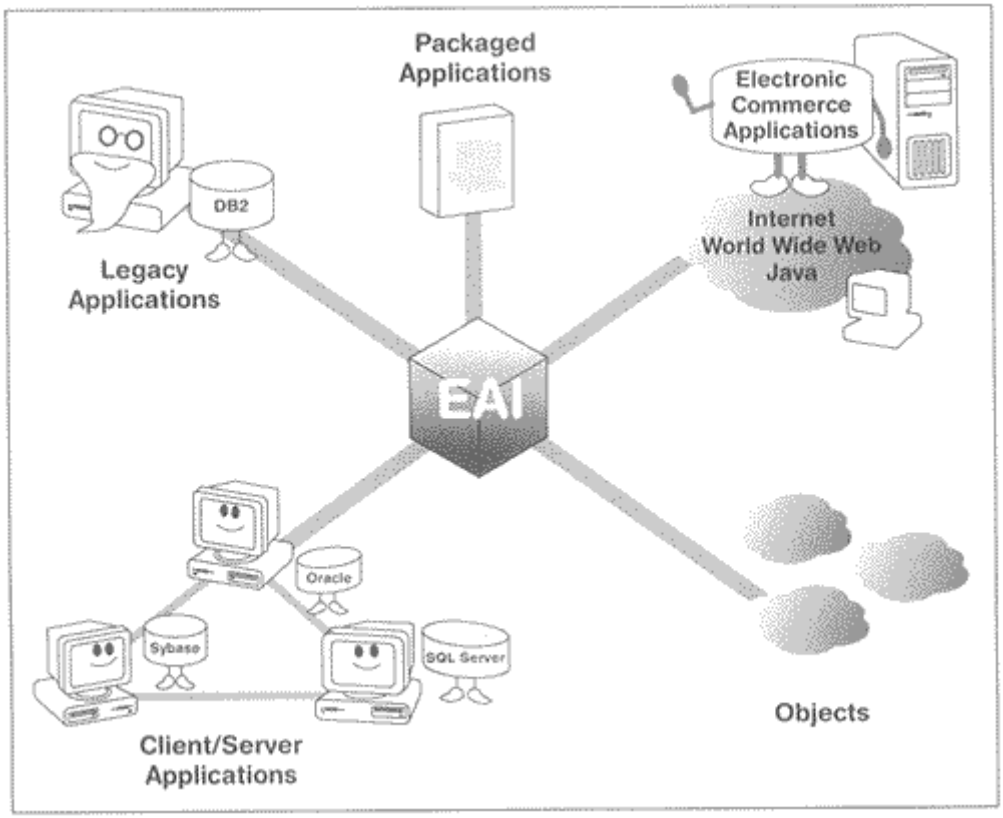
The question is, If such chaos exists today, what can tomorrow hold?

The "order solution" to this chaos is twofold. First, the enterprise needs to understand the large-picture architecture. Understanding the processes and the data that exist within the enterprise and how each exists within the context of the other is the necessary first step in moving forward. Then, understanding the architecture, the enterprise can determine which applications and data stores need to share information, and why. With this basic requirement established, it is simply a matter of determining the correct architecture and enabling technology to solve the problem.

Sounds easy, doesn't it? It is—on paper. The truth of the matter is that enterprises are much more complex and difficult to contend with than one may think. Obstacles, such as interdepartmental turf battles and constantly changing business requirements, have little to do with technology but a lot to do with hindering solutions. Because a sound EAI solution often requires a change in organization, the psychological considerations of the workforce and the flow chart structures of the enterprise must be handled appropriately.

Second, new technology must be leveraged to solve the EAI problem. For example, point-to-point middleware, while providing value within the enterprise, does not—and cannot—provide the ultimate solution. **Message brokers** provide a vital "middle ground," offering real promise while other technologies are being created. These brokers are able to move messages from any type of system to any other type of system by changing the format of the messages so that they make sense to the target system. Message brokers also assure that messages are delivered in the correct sequence and in the correct context of the application. (See Figure 1.5.)

Figure 1.5 Using EAI technology to bring order to an enterprise

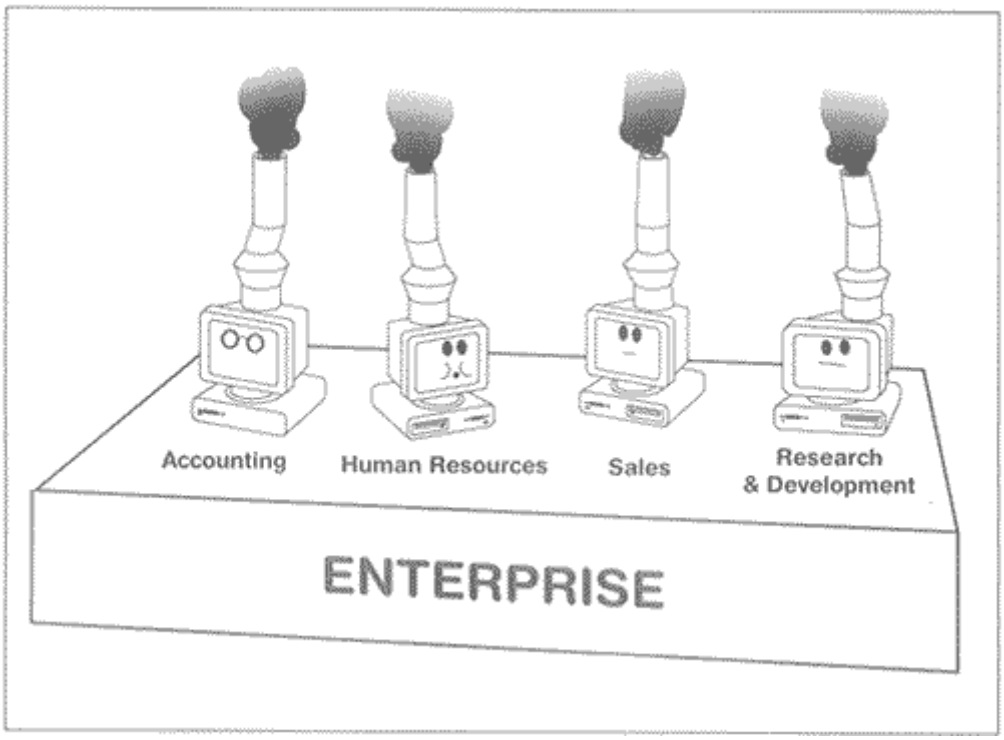


In addition to integrating messages and data, developers are learning to integrate processes as well. A new generation of application servers and distributed object technologies for solving the problem of integrating processes is appearing on the market. These technologies allow developers to build and reuse business processes within an enterprise—even between enterprises.

Evolution of Stovepipes

Within enterprises exist many stovepipe applications that address and solve very narrow problems within departments. (See Figure 1.6.) For example, the new SAP system implemented in accounting is a natural stovepipe application at the module level. Inventory control applications that exist within the sales organization and the resume-tracking system in human resources are also examples of stovepipe applications.

Figure 1.6 Many stovepipe applications exist in an enterprise.



These applications came about when traditional mainframe systems failed to solve departmental problems or, more likely, did not solve them quickly enough. Because of this failure, a "departmentalized" solution ensued, and critical departments implemented their own systems—systems that they owned, maintained, and *protected.*

While departments might have been protective toward their stovepipe systems, that did not mean that they, or the people who maintained them, did not want to share information; it just meant that data, and the audience for the system, typically existed within a single department. Inevitably, this reality demonstrated that the system was built without much thought having been given to the requirements for information sharing. As a result, there were no open application programming interfaces (APIs), open architectures, or other mechanisms that allowed for ready access to the processes and data that existed within these stovepipe systems.

The increased number of enterprise resource-planning applications and other packaged applications that are becoming commonplace in most enterprises represent the "mothers of all stovepipe systems." These systems do not provide effective methods for accessing data and processes within their own environments, and, while interfaces to these packaged applications are improving with time, integrating them within the enterprise represents a significant challenge.

Traditional Systems

Traditional systems (also known as "legacy systems") are stovepipe applications that may exist with many other stovepipe applications in a centralized environment. While mainframes continue to make up the majority of traditional systems, minicomputers and even large UNIX systems may also correctly be called traditional systems.

The characteristics that define the traditional system include centralized processing with terminal-based access. Features of the traditional system include both database and business processing existing together within the same environment. In addition, traditional systems typically support a large user and processing load. It is not unusual for these systems to support thousands of users concurrently accessing the same application.

The significance of this is that the much heralded death of traditional systems has proven to have been somewhat premature. Rather than becoming extinct, these systems not only continue to sell, but older applications leveraging traditional systems have demonstrated significantly more staying power than originally anticipated. The rise of EAI could be attributed, in part, to the need to maintain these older applications and integrate them within the new enterprise application infrastructure.

Microcomputer Systems

Microcomputer systems—personal computers—that exist within the enterprise today represent a significant challenge to those who seek to implement EAI within the corporation. This challenge is made clear by the fact that microcomputers exist on thousands of desktops within an organization, each microcomputer containing valuable information and processes. Complicating the situation is the fact that no two microcomputers are exactly alike. As a result, integrating the data and processes that may exist on these desktops within the enterprise could prove to be a nightmare.

Traditional microcomputer application development, such as those applications built during the rise of the PC, left far too many processes existing on the desktop and thus potentially impossible to access. Accessing both the processes and the data that exist on microcomputers may require rehosting, or moving the processes and data to a centralized server.

Distributed Systems

Simply stated, **distributed systems** are any number of workstation servers and hosts tied together by a network that supports any number of applications. This definition covers a broad spectrum of computing trends—client/server, Internet/intranet, and distributed object computing architectures.

Distributed computing implies that distribution provides clear benefits, including **scalability** and fault-tolerance. While these benefits do exist to a certain degree, distributed systems, though architecturally elegant, are difficult to implement. The majority of failed application development projects over the past decade resulted from overly complex distributed computing architectures. This does not mean that distributed computing is always a bad architecture. It does, however, urge caution and a broad perspective. Distributed computing, like any other powerful computing concept, must exist within context with other types of solutions.

Despite the complexities of distributed computing, there is a real advantage to this architecture when it is considered with EAI. Because the architecture is already distributed—riding on open systems with open, better-defined interfaces—it lends

itself well to integration when compared to traditional, centralized computing architecture.

Packaged Applications

Packaged applications are any type of application that is purchased rather than developed. These applications contain reusable business processes that represent best-of-breed business models and don't require a full-scale development effort. The benefit to business is plain: Why develop a new inventory control system when the best inventory control system already exists?

While enterprise resource-planning applications are the leaders in the packaged applications market, many other types of packaged applications are finding their way into the enterprises. These include call center applications, sales automation applications, and inventory control applications.

While the question, "Why develop a new application when a packaged one exists?" is a valid one, the truth is that the popularity of packaged applications may very well be the driving force for EAI. Packaged applications are natural stovepipes. As we have seen, any stovepipe application is difficult to integrate with the rest of the enterprise. EAI has been brought into play as a mechanism, not only to integrate existing enterprise applications, but also to integrate and free the information from the new generation of packaged applications. This was clear from the earliest days of EAI when most, if not all, integration efforts focused on bundling packaged applications. Packaged applications mean so much within the context of EAI that we have dedicated several chapters of this book to discussing the details of the major enterprise resource-planning applications, including SAP and People-Soft.

Making the Business Case for EAI

We have already noted that the business environment no longer supports using technology for technology's sake. To justify its expense, a technology must demonstrate its usefulness. EAI is no exception.

While the business case for EAI is clear to most people versed in the technical aspects of this discussion, it might not be as clear to those who really need to understand its value. For example: will implementing EAI within an enterprise provide a return worthy of the investment? If so, how long before the return is realized? Is EAI a short-term or long-term proposition? And, perhaps most importantly, what are the methods that best measure success?

Before we make the business case for EAI, a number of things should be understood. First, implementing EAI requires that someone thoroughly understand the business processes in the enterprise. It is only by using this information that the degree of integration necessary to optimize those business processes can be determined. While there are methodologies and procedures that can be applied, most competent managers understand the degree of value when applying EAI without over-analyzing this information.

Not all organizations are equal. For this reason, some organizations will benefit more than others from EAI. While some organizations clearly demand an EAI initiative, others might find almost no value in implementing EAI within their enterprises. If a customer leverages a single, host-based computer system with few applications existing on that platform, only a few thousand users would have only limited benefit from EAI. When applications and data exist within the same environment, they are much easier to integrate. For years programmers have been successfully integrating homogeneous applications.

However, it is when applications and data do not exist within the same environment; when the organization has gone through mergers or acquisitions; when it has experienced uncontrolled architecture and unruly application development projects; or when it has a large, distributed system with a multitude of platforms and protocols that exist to support hundreds, or thousands of users, that the enterprise will realize a tremendous benefit from EAI.

While it may be possible to develop a "common sense" set of metrics to evaluate the success of EAI, the reality is that in most cases they must be significantly adjusted on a case-by-case basis to account for the many factors that exist in any given enterprise. Because of this, there is no easy way to create a broad-based way to define EAI success. It must be measured enterprise by enterprise.

In order to evaluate the value of EAI to your enterprise, you must establish a set of measures that define success for your organization. This can be accomplished by examining and measuring the current state of the enterprise. With this baseline, consider your goals and the amount of effort that will be required for you to realize them.

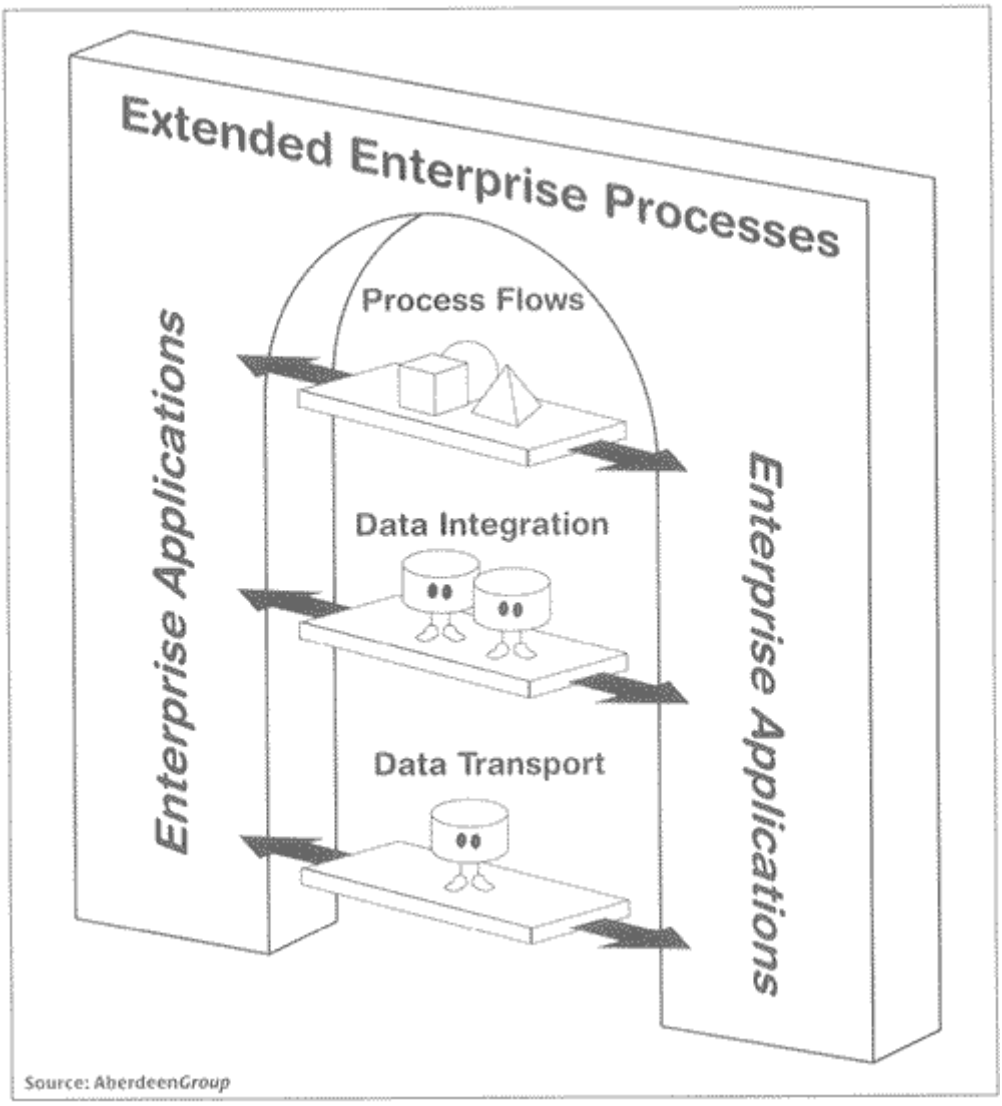
For example, if increasing sales is one of your goals, sharing inventory information with the sales order-processing system may allow sales to operate more effectively and thus help realize that goal. Even so, without a significant gain in user productivity or a reduction in error rate, the integration effort between two systems has minimal value. Therefore, in order to accurately assess EAI, you need to weigh both user productivity and error reduction, giving a measure to both. Only then can you determine the impact of EAI on your enterprise.

The Virtual System

The ultimate EAI scenario is the common virtual system. This allows any and all information required for all transactions to be immediately available no matter where the information is located in the enterprise. Every application, database, transaction table, method, and other element of enterprise computing is accessible anytime, anywhere.

EAI is able to take many diverse systems and bundle them in such a way that they appear—and function—as a monolithic and unified application (see Figure 1.7). While this represents the nirvana of EAI—and as such is not achievable for most organizations in the near future—it represents what ultimately is the perfect integration of systems possible with EAI.

Figure 1.7 Extending the enterprise with EAI



e-Business

e-Business (i.e., using electronic mechanisms to conduct business between your trading partners and your customers) is an ideally efficient method of commerce. e-Business comes in two "flavors": one, business-to-business inclusive of supply chain integration and, two, business-to-consumer, inclusive of Internet commerce or conducting commerce on the Web. While both "flavors" would benefit from EAI, it is a natural fit for business-to-business or supply chain integration in that one is able to serve the needs of the other.

The mechanism techniques and technology that make up EAI are also applicable to most supply chain integration scenarios, or Inter-Enterprise Application Integration. Because EAI is very good at integrating various applications and data stores, it is able to extend its reach outside the enterprise to include both trading partners and customers within the enterprise integration architecture. This enables any application