

AN O'REILLY FIELD GUIDE TO ENTERPRISE SOFTWARE

# Enterprise Services Architecture



O'REILLY®

DAN WOODS

---

# **Enterprise Services Architecture**

---

# Enterprise Services Architecture

*Dan Woods*

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

# Concepts and Philosophy

The headlines in Silicon Valley in the summer of 2003 are about mergers instead of the latest new thing. Larger companies are grasping for smaller ones. Sectors are consolidating. Pundits are declaring that the entire Information Technology (IT) industry is a commodity, a tool for cost-cutting rather than innovation.

This tornado of hype explaining how companies should react to improve their competitive position offers little of practical value. What is impossible to reduce to a headline, sound bite, or glib analysis are the choices companies should make for their own IT architectures.

At the core of the forces generating the latest news is the widespread recognition that the land-grab phase of the IT industry is over. The industry is growing up. One can look at the growth of IT as the gradual extension of automation, starting with financial applications on the mainframe and then extending out through the personal computer, workstations, client/server, and the Internet as well as through applications for Enterprise Resources Planning, Human Resources, Customer Relationship Management, Supply Chain Management, and so on.

For almost every need there is a product, but each is a world unto itself. They don't talk to each other well or share data easily. Valuable services trapped inside are not reusable.

At the same time, companies are becoming aware that IT should help by automating and optimizing the processes of a business. But understanding of processes has evolved beyond the capability of

existing enterprise applications. Processes cross the boundaries of enterprise applications and extend beyond the boundaries of the company itself to suppliers, distributors, and customers. Automating processes across these boundaries is difficult and expensive. IT has become a bottleneck retarding the next phase of automation.

Portal technology that brings together functions from a variety of applications can help. The emergence of web services, XML-based standards, and other integration technology will make improving the situation easier. But for the next five years and beyond, it is clear that just slapping portals and web services onto existing enterprise applications will not solve the problem of unbounded process automation. The underlying applications are not ready to play that game.

Enterprise Services Architecture describes the solution. It is the blueprint, the set of principles, the values used to form a coherent answer to all of the questions facing companies today. The concept was hatched in 2002 in the mind of Hasso Plattner, the chairman of SAP AG, as he struggled to conceive of the next mountain his customers would have to climb. The concept, which builds on the principles applied during 30 years of company history, has bubbled through SAP and resulted in the creation of SAP xApps and SAP NetWeaver, technologies that enable a company to make progress toward an architecture that embodies Enterprise Services Architecture principles. But Enterprise Services Architecture is not an SAP product or even a product at all. Instead, it is a philosophy that guides the evolution of IT architecture along a path toward increased business value.

The core of Enterprise Services Architecture is a layer of components that coalesces data and application functionality from enterprise applications into useful and reusable modules. The components communicate using enterprise services. These services reduce the complexity of the connections among components to allow for reuse. The transition is similar to imagining a plate of spaghetti and meatballs transforming into a set of tinker toys. The meatballs (the components) become more structured and the spaghetti (the nest of complex connections) disappears, replaced by more clearly defined and structured relationships expressed as services.

Architecture is essentially a set of decisions and choices that one will have to live with for a long time. Enterprise Services Architecture is an analysis and a program for making those choices, based on the state of the IT industry and basic economics, to optimize the business value that is produced from investment in technology.

This book will explain what Enterprise Services Architecture is for the senior management and IT professionals who stand at the threshold of tremendous opportunity and who are, at the same time, being pursued by increasing threats on all sides.

The right architecture for any given company is a matter of crafting the best possible response to its competitive position. In IT strategy the landscape consists of multiple departments or divisions of the company, each with multiple suppliers, all working to create products and services for an increasingly competitive market.

In his book *Management Challenges for the 21st Century* (Harper-Business), Peter Drucker points out that one of the fallacies of management theory over the past 80 years has been the assumption of one right organization for a corporation. This notion applies to IT architecture as well. Enterprise Services Architecture is not a description of a single correct architecture but rather a description of important questions to ask about IT systems and a pattern for successful individualized answers that take into account the existing infrastructure and business conditions of our time.

If this book succeeds in its mission, it will help executives discover what Enterprise Services Architecture means to them, and most importantly what to do about it.

In the following pages, we will take a trip through all of the issues surrounding Enterprise Services architecture including:

- The business drivers putting pressure on companies to be more flexible
- The technology that enables the construction of components and services

- The structure of the Enterprise Services Architecture platform that brings Enterprise Services to life to create business value
- The challenges implicit in designing and implementing Enterprise Services Architecture at a company
- The effect that Enterprise Services Architecture will have on companies, technology vendors, and our economy at large

The IT industry is not the first to grow up in this manner. We will now take a look at the automobile industry, which followed a path of increased abstraction and componentization in the pursuit of efficiency and flexibility.

## Components in Lean Manufacturing

Automobiles used to be thought of as large collections of parts. Each model was a world unto itself, a monolith, assembled from thousands of unique parts. Few parts were shared across cars or car makers. This structure amplified complexity and dramatically increased design, manufacturing, and maintenance costs. In the past 20 years, car makers have replaced the notion of thousands of parts with the idea of a much smaller number of subsystems. These subsystems fit together in standardized ways and are assembled in a particular order. The brakes, the transmission, the engine, and the steering mechanism all have the equivalent of plugs and sockets on them.

This approach has been adopted in every other sort of manufacturing, from watches to cell phones to vacuum cleaners. Componentization has not reduced complexity—cars are more complex than ever—but it has made manufacturing more manageable and less costly. At first, the components were used within the models of one car maker, but now they have spread so different firms use the same standardized components. This approach is called *lean manufacturing*.\*

---

\* For details of this transformation, see *The Machine That Changed The World: The Story of Lean Production* by James Womack, Daniel T. Jones, and Daniel Roos (HarperCollins).

This transformation did not occur overnight, and some firms proved better at it than others. In general, Japanese firms still lead American companies in adopting this approach. They use components extensively and design the car up front. Design is the province of the car maker while implementation details are handled by suppliers.

Enterprise Services Architecture is lean manufacturing for IT. It represents the imperative to componentize according to a standard architecture. Business software is actually much more complex than an automobile. With its myriad data elements, interfaces, and algorithms, a fully functional ERP system is many times more complex than a car. Because software is so pliable, we sometimes think that developing software must be easier than making physical objects. The truth is that the complexity is mind-boggling and should be approached with humility.

Enterprise Services Architecture offers companies a coherent plan for playing the role of car maker. The equivalent of the thousands of parts of the old-style manufacturing mode is the siloed complexity of applications. The equivalent of the standardized components of lean manufacturing is the sort of components that are created in an Enterprise Services Architecture platform. To make this transition a company must understand its processes and its current enterprise applications and then determine where it needs to build components that allow for the flexibility that business conditions increasingly demand.

The IT industry is very early in this cycle. Components exist only at the level of core technology. Relational databases are the best example, while web servers and browsers are the most recent. But even these components are fuzzy compared with components in manufacturing. Relational databases, for example, are standardized only to a degree. Products from different vendors have significant differences in core areas such as how to write procedures that are executed in response to various database events.

IT infrastructure is like aircraft. When a plane gets upgraded, most of its structure stays in place and continues to do its job. Companies

over the past 30 years have built a tremendously complex infrastructure that has gone through several cycles from mainframe, to client/server, to Internet applications, and now to composite applications, which build on the existing infrastructure.

Technology succeeds and stays in place in the long-term because it addresses a business problem. Legacy systems are systems that work. One of the core values of Enterprise Services Architecture is accepting this fact and keeping existing systems in place, encapsulating them in abstractions at the right level, componentizing them when necessary, and then building new systems using them as parts.

## Standards Are Vital

A major implication of these trends is a dramatic rise in the importance of standards. The proliferation of interfaces between companies and between components in companies gives standardization a value it did not have in the past. A tiered architecture relies on standard ways of communicating. In the early stages, the entity with the most power dictates to less powerful partners how they will do business. Wal-Mart did not wait around for a standards body to approve its vendor-managed interfaces. It explained politely to its partners that supporting the interfaces was a requirement of doing business with Wal-Mart. It is now doing the same thing with radio frequency identification (RFID) technology, which is used to automatically identify products via electronic scanning. From the standpoint of mergers and acquisitions, divisions that adhere to standards may fetch a higher price because they offer smooth integration with a new owner.

Standardization can result in increased customer adoption, better customer service, and improved product quality. The GSM wireless standard adopted across Europe enables every wireless vendor to compete in every other country, and the resulting competition has created much more advanced wireless services than in the United States.

Supporting emerging industry standards and participating in their design can be another source of competitive advantage. Enterprise Services Architecture with its emphasis on abstraction and componentization is a perfect mechanism for adapting to and implementing the increasing number of internal and external standards required for doing business.

Standardization took place in the manufacturing industry over the past several decades. In automobile manufacturing, at first only smaller parts like spark plugs and batteries became standard components. Within a manufacturer, chassis, transmissions, and brake assemblies became common across models. Third parties started manufacturing components that were used by multiple manufacturers.

The key to componentization on a mass scale is standardization. Third parties could not make components that would be used across manufacturers without some sort of standard for what the component would do and how it would fit into other components. Over time these standards evolve. Aircraft get updated over their 25-year lifetime with different engines and other subsystems.

Standards have two parts: the design of the individual components and the design of the architecture as a whole that make sense of how all of the components fit together. The challenge of Enterprise Services Architecture is to create an environment for IT where standardized components can work together without a return to monolithic complexity. To create really reusable components, we need an infrastructure that allows components to adapt to the needs of the environment. We also need an environment that is able to adapt to plugged-in components.

Even in manufacturing, standards followed; they did not lead. To gain the benefits of flexibility, lower costs, and the ability to customize to meet market demands, manufacturers componentized their products prior to adoption of standards across the industry. Those in the IT industry who wait for standards to improve their architecture will be waiting quite a while and will lose out on the business benefits of Enterprise Services Architecture in the meantime.

## The Business Drivers for Flexibility

The fragmented world of data and software would really not be much of a problem if business conditions remained constant, as they did in the early 1990s before the arrival of the Internet as a primary force. Back then, most companies were self-contained units, impenetrable to the outside, and IT systems were an internal affair. A painful and clunky user interface was not a large problem. The people using the software were employees, usually with highly specialized functions, and they would make do because they had to.

Now the network has brought suppliers and customers right inside that infrastructure. Customers frequently use computer interfaces to a company's systems, often through the Internet. This computer-mediated interaction with customers applies not just in a few industries and contexts, like customers interacting with a bank through an ATM or with an airline reservations system through a web site, but it has spread to almost every company. Customers become angry if an interface is awkward or if the system is slow. They are insulted if they show up at one touchpoint of the company and it becomes clear that the underlying application is trapped in a silo and does not know about the other relationships with the company that are trapped in other silos.

At the business-to-business level, creating the most efficient supply chain involves opening up the core systems of the company and integrating them with vendors' systems at key points. Under the vendor-managed inventory paradigm that Wal-Mart has made famous, orders automatically flow from stores to suppliers based on how fast products are selling, creating a core competence that is hard to replicate. Enterprise Services Architecture enables IT flexibility so that such innovative strategies are practical and affordable.

We will now take a closer look to try to understand the forces putting pressure on business to expose their internal systems to the outside world.

# Trends Increasing the Demand for Enterprise Services Architecture

While it is possible to see how expanding automation, increasing flexibility, and so on are beneficial in general, how do we know that they will result in a clear ROI? Why will these factors be an important part of successful companies? In previous eras, expanding to new lines of business, gaining market share, or improving financial stability were the most important factors in determining success. Why are the benefits of Enterprise Services Architecture crucial for the next battles that will be fought in the marketplace?

These questions deserve a book of their own. They cannot be answered in general, but only for a specific company with regard to a specific set of circumstances. The goal of this book is to provide an understanding of Enterprise Services Architecture—its meaning, structure, and implications—so that executives can complete the job of evaluating how Enterprise Services Architecture might help by filling in details from their own context. But, that said, several major trends support the argument that the benefits of Enterprise Services Architecture and composite applications will be key factors for success.

Our argument is that the increase in customer power, direct access to systems, increased competition, and the rise of services will all require increased flexibility in IT systems.

The rise in the power of customers is the first major trend that will drive companies to become more flexible. In many industries, the Internet has shifted the balance of power in favor of customers. In automobile sales, the manufacturer used to control all the information and held all the cards. Consumers who walk into an auto dealer today are likely to be armed with precise cost information that enables them to bargain harder than ever before. With web interfaces for most stores, it is now possible to comparison shop with little effort. The cost of switching suppliers is lower than ever. This consumer power translates into a demand for better service, lower prices, and products more precisely tailored to a much more granular segmentation of the market. This trend places direct demands on

the IT infrastructure to support the most customized products and interaction with the consumer at the lowest possible cost.

The second major trend is the rise in computer-mediated interaction with consumers, suppliers, regulators, financial institutions, and every outside party involved in a company's operations. Ricketty first-generation web sites have given way to streamlined interfaces that are a pleasure to work with. Companies now differentiate themselves and compete on the basis of user experience. How many of us still use Amazon.com even though we know of discounters that might be able to save us a dollar or two per book at the cost of having to suffer through a more difficult user experience and a less reliable fulfillment process? The rise in interfaces means a rise in the visibility of internal systems. Outsiders can now peer into the glass house of the data center and see if it is a mess. How many companies miss opportunities for meeting customers' needs because they cannot clean up their internal systems well enough to present accurate information to the outside world? This remains a major problem for financial companies that, through internal growth and from mergers, may have the same customer for consumer banking, credit cards, investments, and insurance and not know it. The components of Enterprise Services Architecture can unify this information without requiring extensive retooling of the underlying enterprise applications. The resulting repository of all customer information can provide any particular system access to a complete view of the customer.

Barriers to entry to many businesses are dropping. Small telecommunications providers such as GTE have grown to take on the Baby Bells. Moreover, AOL grew from a small Internet access provider to a giant that gobbled up Time Warner. In the content industry, fierce competition from Internet content providers is leading to rapid consolidation in the publishing industry. Biotech firms are redefining the traditional pharmaceutical industries. As the barriers to entry fall, entrepreneurs and venture capitalists stand ready to take advantage. Peter Drucker points out that new entrants to a business frequently have about a 30 percent cost and efficiency advantage over existing

firms because their newly built systems are optimized for the modern supply chain. Companies that are inefficient and inflexible become targets in this environment.

Corporations are seen increasingly by both consumers and other businesses as a set of services. We see interfaces to Amazon.com or Google being exposed as web services. More advanced examples include hosting and support relationships with complex service level agreements. CenterBeam, for example, provides off-site management of a local IT infrastructure. This paradigm exchanges employees for centrally provided services that provide equivalent functionality—maintaining and backing up servers, monitoring important applications, and managing a network. Companies are differentiating themselves from competitors by the kind of services they provide and how they deliver them. The ability of an IT infrastructure to support the delivery of precisely defined services maps naturally to the component structure of Enterprise Services Architecture.

All these trends are leading toward the same transformation in many businesses that occurred in manufacturing. In the automobile industry, for example, the major car manufacturers stopped making many of the components used in their cars. The car makers became the branding and design centers of their industries. Tier 1 suppliers manufacture the components designed to be assembled into completed automobiles. Tier 2 firms make parts that are used to assemble tier 1 components. Tier 3 companies provide raw materials or commodity parts. This manufacturing-style tiering is spreading to a wide variety of industries. Amazon.com, for example, offers used books from thousands of mom-and-pop distributors through the same interface it uses to sell new books. Amazon is happy to do this because it provides better service for its customers and the company makes about the same amount selling a used book from a third party distributor as selling a new book from its warehouse. Amazon.com is the car maker, the assembler or orchestrator, and the smaller bookstores are the tier 1 suppliers. Starbucks, an orchestrator, is becoming a major provider of Wi-Fi services from T-Mobile, a tier 1 company in this scenario.

A fundamental argument of this book is that this sort of tiering will become increasingly important and that Enterprise Services Architecture is the way to be an effective participant in this structure. As we will see in Chapter 6, which addresses where companies should seek to increase their flexibility, the tiering will increasingly force companies to examine what is core to their business and what is secondary. The optimal structure is one in which a company invests its resources in core areas and outsources the rest to other companies whose core competencies lie in those areas. For all this, IT flexibility will be required.

With all of these forces putting pressure on companies to increase flexibility, the question becomes how can the architecture keep up. We will now take a look at the practical next steps.

## Transforming the Architecture

Leading innovation in IT can be a risky business. At many companies, the leader of the reengineering effort of the early- to mid-1990s is nowhere to be found. The person who led e-business initiatives was generally fired for not meeting wildly excessive expectations. The leader of the Y2K project is probably not a hero.

“Who is going to believe in anything in this environment?” says Adolf Allesch, a seasoned consultant from Cap Gemini Ernst & Young.

The question is what do you believe and why? The faith that Enterprise Services Architecture requires is as follows:

- The next stage of IT evolution must build on the systems currently in place.
- Architecture must be designed based on a thorough understanding of existing systems and business strategy.
- An architecture based on loosely coupled components and services provides the most flexibility at the lowest cost.

- As the cost of change drops in an IT infrastructure, more tactics become affordable, the cost of supporting new relationships is reduced, and new strategies can be implemented faster.
- The ability to change the IT infrastructure faster than competitors to support evolving strategy and to adapt to business conditions will lead to a significant competitive advantage.

The implication of these beliefs is that IT strategy can no longer be decentralized. Even if systems are distributed for operational purposes, they must all speak the same language. Affordable implementation of these articles of faith demands that the IT infrastructure become an ecosystem in which applications present themselves to each other for reuse, where new user interfaces can be constructed out of components and services, and where hard-coded business processes can be replaced with more flexible means of scripting the coordination of components.

Components and services are not a panacea. They can be used to create a monolith (i.e., an inflexible system). But when implementation is guided by a vision for where flexibility is most required and a competent design, Enterprise Services Architecture can be a powerful enabler of business value. Unlike many other technology solutions that provide part of the solution, Enterprise Services Architecture completes the loop and provides an end-to-end vision of cleaning up existing operations and then incorporating third parties. Enterprise Services Architecture provides the essential blueprint for creating systems that allow companies to gain real-time understanding of their operations and take informed actions that produce business advantages. The technical developments that make this possible are the next topic in our analysis.

## The Evolution of IT Architecture

While the motivations for bothering with Enterprise Services Architecture at all lay primarily in the realm of business, the reasons that Enterprise Services Architecture has become a short-term imperative lie in the realm of technology.

The design principles of Enterprise Services Architecture spring from mature concepts and techniques such as service-oriented architecture, modeling, and object-oriented programming. Up to this point, these ideas have mostly been relevant to advanced system architects who employed them in creating elegant implementations within applications.

But the expansion of automation throughout the enterprise, the arrival of technologies such as the Internet, XML, web services, and business process management standards, and the maturation of a full set of platform systems for content management, data warehousing, and portals all create the necessary conditions for important architectural ideas to break out of the realm of theory and to start informing the creation of business technology.

Enterprise Services Architecture provides the opportunity to reorganize three generations of business technology around proven architectural principles from computer science. The mainframe era, the client/server era, and the Internet era all filled out the toolbox, creating standards, enterprise applications, and platform component systems to meet the fundamental application and functional needs of the business world. Enterprise Services Architecture now seeks to make sense of it and apply some structure to increase the value businesses can extract from technology.

The reason that Enterprise Services Architecture has gone from a good idea that might happen someday to an exciting opportunity right now is that a variety of elements that have evolved over time are now able to interact in a new way. This is similar to biological evolution in which parts of an organism gradually morph until they all work together differently and enable a major leap forward. Enabling factors in Enterprise Services Architecture's evolutionary process include the availability of advanced development tools, networking standards, and standardized data structures.

The first factor setting the stage for an evolutionary leap is the full suite of enterprise applications developed over the past 30 years. The core financial and control functions were the initial focus of the

mainframe era. Applications grew out from this core gradually. During the client/server era, their footprint expanded as applications for Human Resources and Sales Force Automation took root. In the Internet era, the pervasive network allowed applications like Customer Relationship Management, Supply Chain Management and Supplier Relationship Management to automate processes. As a result of these phases, most corporations have collected a heterogeneous mix of applications from many different vendors.

The second major factor in the evolution of infrastructure is the parallel development of platforms and toolkits for a broad scope of application functionality. At the foundation are the operating systems themselves, which provided the base on which computer languages, interactive development environments, and related developer tools were constructed. Relational database systems gradually matured into a stable repository for persistent data from most applications. Platform component systems for content management, data warehousing, and application integration all provided powerful toolkits to build custom applications and extend the functionality of existing legacy and enterprise applications.

A third development that makes Enterprise Services Architecture ready for prime time is the way the Internet broke through the armor that surrounded the enterprise. With TCP/IP as the standard way of communicating across networks, web browsers based on HTML as the user interface, and HTTP as the communication protocol, enterprise applications could now reach out directly to consumers and suppliers. Applications could also more easily communicate with each other. The investment boom that grew from these technological developments resulted in the acceleration of the growth in enterprise applications and platform component systems. Network-oriented enterprise applications and platform components such as application servers and enterprise application integration (EAI) systems were created in this period.

Data is being prepared for Enterprise Services Architecture through XML. Inspired by the simplicity and popularity of HTML, the ornate and complicated language of SGML was distilled into XML, a flexi-

ble way to describe the structure of data. XML also describes interfaces and protocols in a way that is interoperable across different applications and platforms. The rise of XML spurred a wave of standard setting for data and protocols in almost every industry that continues to this day. Efforts like RosettaNet and OASIS are extending standards in every direction including the semantics of the data. Almost every standards effort is based on XML in one way or another. Enterprise applications use XML as a standard way of describing data, and EAI technology moves XML messages from one application to another. XML has also spurred the creation of a variety of toolkits for managing and transforming data stored in XML format. The result of the spread of XML is that conversations about standardization and integration are much more focused on what to do rather than how to do it.

While all these other developments are profound in their effect, web services is perhaps the most significant. It is the catalyst that allows Enterprise Services Architecture to spring into being. Web services are like a universal plug and socket system that allows applications to describe and implement different ways to communicate with each other. Unlike CORBA or DCOM and other previous attempts at this sort of connector technology, web services are based on XML and truly platform-independent protocols such as SOAP. Like HTML and XML, web services are also simple and flexible. Web services technology has become the default interface to application components and services. Web services are simple and universal enough that the vast majority of the participants in the IT universe are convinced that all applications will eventually be able to communicate with each other through web services. This simplicity has become a problem as its popularity has outstripped its maturity. Some basic definitions for security, guaranteed delivery of messages, transactions, and other portions of the standard are still being defined. These gaps do not obscure the fact that, like XML, web services have changed the conversation about application integration to focus more on the goals than the methods.

One development that is still more a hope than a reality is the emergence of a strong consensus around the creation of languages to describe business processes. IBM and Microsoft combined their separate long-standing efforts toward business process modeling to create a common language called BPEL4WS, which merges the best ideas from both companies. SAP has signed on as an author of the standard and many other leading technology vendors also endorse BPEL4WS. Similar efforts called Business Process Modeling Language and Web Service Choreography Interface are also under development. These languages could become a standard way to describe the orchestration of components and services and the foundation for portable descriptions of business processes. The largest benefit accrues from these efforts if one of these languages becomes the standardized way to control the behavior of enterprise applications, which would allow tremendous configurability and interoperability.

With all of this infrastructure in place, including a complete set of functionality from enterprise applications and platform component systems, standard ways of describing data and interfaces between applications, and even possibly a method of orchestrating the behavior of multiple components, Enterprise Services Architecture is now a practical possibility. The amount of work required to achieve a better architecture has become manageable.

## **Inside Enterprise Services Architecture**

So far we have explained the context in which Enterprise Services Architecture exists and the business benefits that may accrue. We have hinted at the mechanisms, such as loosely coupled components, services, and modeling, that represent crucial building blocks. Now we will delve into the moving parts and demonstrate how they work together.

It is fashionable to complain about silos of data trapped in distinct enterprise applications, as if this problem could have been foreseen

and avoided. Perhaps it could have, but during the boom times from 1995 to 2000 when most of these applications were implemented, few companies kept their eye on creating a coherent architecture. Technology vendors provided little assistance during this period as they too focused on expanding the footprint of application functionality rather than on the larger issues of integration. Implementing applications to attack new markets and pave the way for growth was the order of the day. Most companies preferred gaining the automation from new applications to reconciling and synchronizing all the data and functionality. The result was the silos: each new application arriving with new data and new functionality, separate from those that came before.

While all of this may be obvious, the best way to escape the silos is far from clear. As the IT industry matures, it will be beneficial for products and systems to follow the same road as manufacturing and become a series of subsystems or components. How can a company or a vendor figure out the right collection of components? How will they fit together? What benefits would accrue from such a structure? What problems would arise? How will they be purchased or constructed? And at what cost?

Economic reality sets limits on any solution. Investment in the current generation of systems and the value they provide is so large that we cannot start from scratch. How can we build on top of the current generation of applications in an incremental fashion? How can the company rise above the silos to get flexibility at a reasonable cost?

To explain in concrete terms the solution Enterprise Services Architecture proposes, we will take a brief tour through the development of enterprise applications from the mainframe era to the present day.

The first computer applications that gained wide acceptance were built on mainframe computers that had only the minimal support for development in the operating system. The resulting applications had to do everything for themselves in one big monolithic blob of functionality as shown in Figure 1-1.

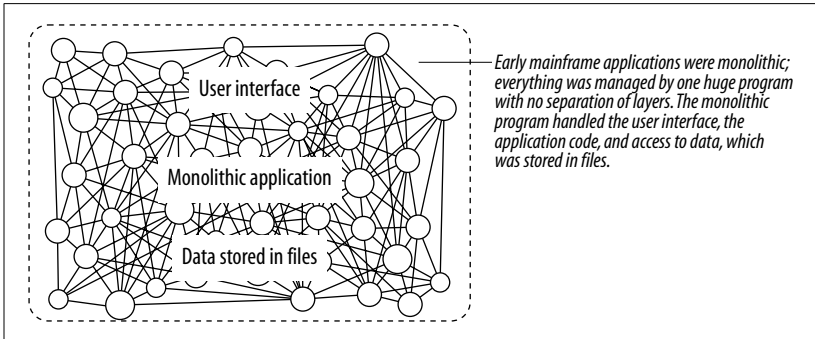


Figure 1-1. Monolithic application structure

Monolithic applications fulfill the purpose for which they were originally constructed but they are hard to adapt and improve. The layers, to the extent they exist at all, are tightly connected to each other. A small change in one area can break the entire application.

The next step for application architecture involved separating the database from the monolithic application, creating a two-tier application. The user interface and the application code remained entwined in the monolith, as shown in Figure 1-2. This structure took advantage of database programs that started to arrive as separate products.

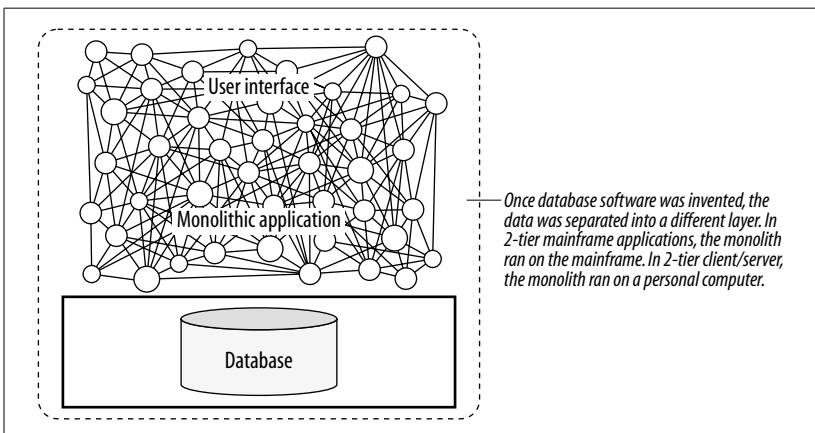


Figure 1-2. Two-tier application structure

In the late 1980s and early 1990s, with the help of the personal computer and the network, the three-tier application arrived. In this

model, the user interface was separated from the monolith, creating three layers as shown in Figure 1-3. Both the user interface and the monolithic application would sometimes run on a personal computer, and sometimes only the user interface would run there. After the arrival of the Internet, everything ran on the server and the browser became the interface. The monolith still existed but shrank as the user interface layer was carved out.

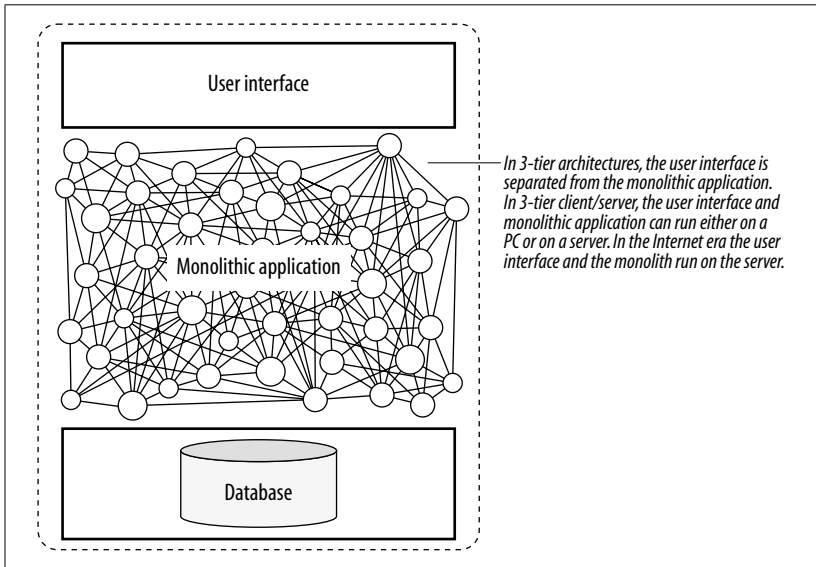


Figure 1-3. Three-tier application structure

The current architecture at most companies is now composed of collections of three-tier applications that either use personal computer applications or web browsers as the primary user interface, as shown in Figure 1-4. These applications are the silos and the fundamental problem is that the user interface of these applications is still bound to the monolith. Companies have made progress in reusing portions of the monolith through portals and through using APIs to integrate one monolith to another. But this approach has significant limitations. Portals are able to access only a fraction of the monolith, and integration using APIs is difficult and expensive.

The primary goal of Enterprise Services Architecture is to overcome these limitations so that creating new applications is much easier

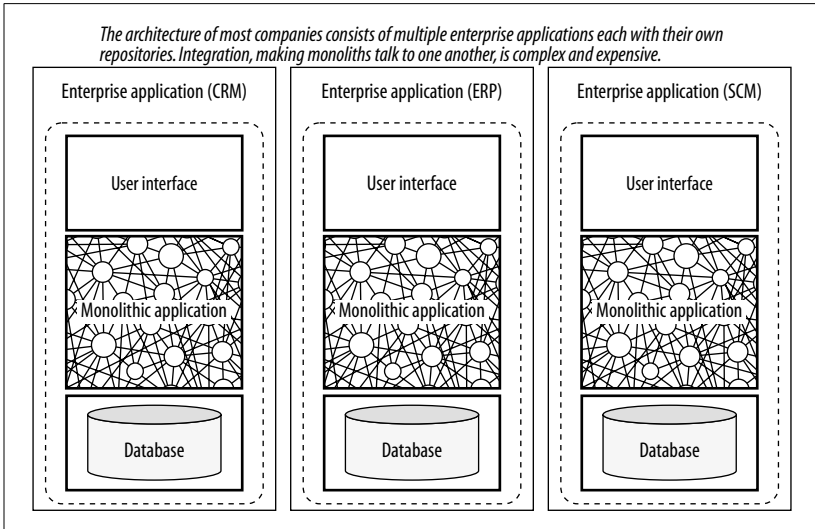


Figure 1-4. Current enterprise architecture

and much more of the monolithic functionality may be reused. This is achieved by breaking the monolith into components as shown in Figure 1-5.

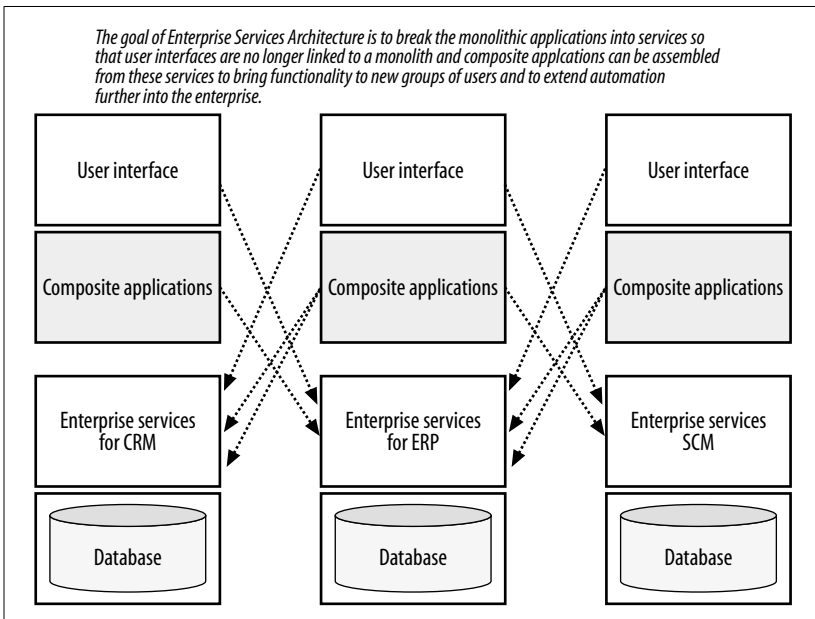


Figure 1-5. Enterprise Services Architecture

Many important things are shown in this diagram. First of all, the monolithic portion of the application has been transformed into a set of components that are connected through services that hide the complexity of components from one another. This unlocks value by making much more of the application reusable.

User interfaces have been freed from the monolith and are not limited by what the original designer of the monolithic application thought was required years ago. They are bound instead to the tasks users need to perform their work and participate in processes. Components can be used to support as many user interfaces as make sense and each user interface can draw on components from several different enterprise applications.

This structure also allows composite applications to be constructed. New components can be created and combined with components from existing enterprise applications to solve new problems and automate processes without regard to the limitations of silos. Integration between applications or with outside parties also becomes component-based and can be accomplished much quicker and cheaper.

All of these benefits accrue when the monoliths have been transformed into components.

But before we get too joyful, let's come down to reality. The world described in Figure 1-5 is not possible with current versions of enterprise applications, which still present their monolithic functionality through APIs. It is impossible for us to build architectures assuming that every enterprise application has been reconstituted into a set of components. They are not and won't look like that for five years or so, by some reasonable estimates.

Fortunately the completely componentized world described in Figure 1-5 is not required for extraction of significant business value. We don't need flexibility in absolutely every part of the infrastructure in order to meet the challenges of the marketplace.

Because of this, we also do not need to wait five years to get the benefit of a componentized architecture. The Enterprise Services Architecture platform, a layer of software that allows components to be constructed from the current generation of enterprise applications, allows us to introduce the right level of flexibility to create maximum value, as shown in Figure 1-6.

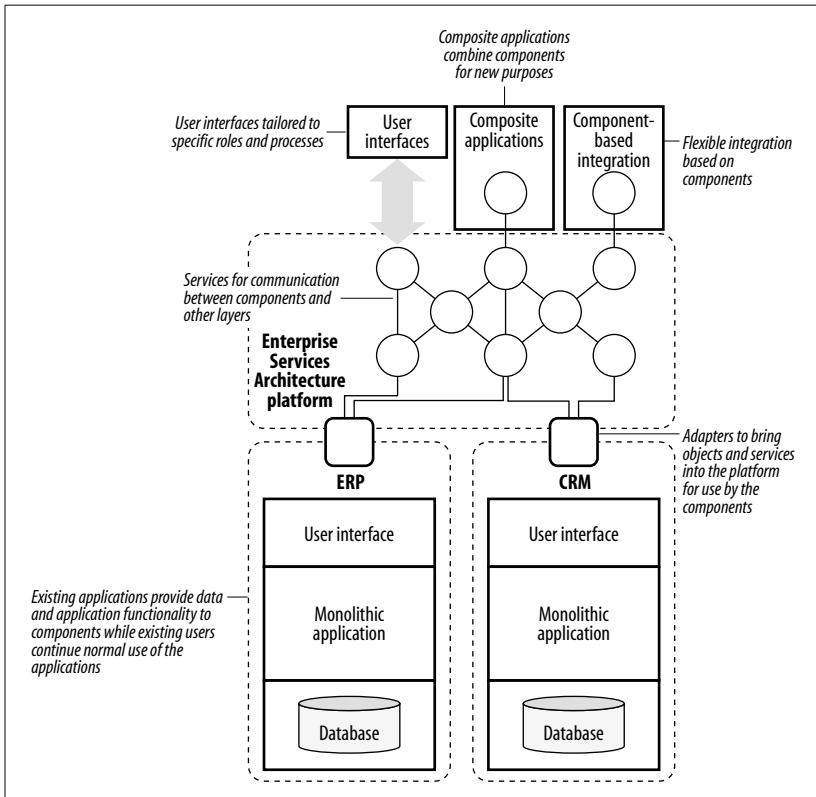


Figure 1-6. Enterprise Services Architecture platform structure

In this graphic, adapters bring the functionality of enterprise applications into a component architecture that can be used to help create new user interfaces, composite applications, or integrations. Some applications such as payroll are stable. Some applications are flexible or dynamic and it is in these areas where change and optimization will occur. This is where components are needed. In Chapter 6, we examine in detail how to determine where flexibility is required in an architecture to provide maximum business value.

This structure brings up a large design problem. What is the right size and shape for the components we have been referring to? How must they be constructed to produce the value we claim? We will now examine the fundamental principles by which these questions are properly answered.

## The Core Values of Enterprise Services Architecture

The core values of Enterprise Services Architecture can be expressed in terms from the realm of computer science. The ideas behind these words, however, are not impenetrable or difficult to understand. They are around us every day.

All of the fundamental ideas of Enterprise Services Architecture are simple, but we have mathematical-sounding names for them: abstraction, models, components, objects, services, processes, loose coupling. Enterprise Services Architecture is at its core the systematic application of principles of successful design applied to IT architecture to produce the maximum benefits for a business. Like a manufacturing company, we want to have flexibility in what products we build with the lowest possible cost for design, manufacturing, and maintenance. A simple example will help to illustrate.

Let's start with the old joke that goes like this: if you ask him what time it is, he will tell you how to build a watch. When we ask our hypothetical friend for the time, let's freeze the action before he answers and think about what we are asking. Time itself is a standard abstraction. Seconds, minutes, and hours do not exist anywhere except in our minds. Time passes; that's reality. But the way we keep track is a convention. There are 60 seconds in a minute and 60 minutes in a hour. Would time be more useful if there were 100 seconds in a minute and 100 minutes in an hour and 10 hours each day, with 5 being the equivalent of noon? Who cares? Our current division of time happened so long ago that it isn't really practical to change it. Time is time, and we are so used to it that we think of it as

a reality, not an abstraction. It doesn't really make sense to change the way we mark time because it would be so impractical.

Now let's start the action back up again, and let our friend answer the question. He telling us how to build a watch instead of what time it is. He tells us about the ticking mechanism that records that small increments of time have passed and how the counters add those small increments together and register seconds, minutes, and hours. He is telling us more than we need to know and is not telling us what we want to know. We want our friend to hide the complexity of the workings of his watch from us and just provide the useful information that is important to us. Hiding the details is what happens in an abstraction. The complex workings of the watch proceed inside the abstraction but the useful information, the time, pops out.

Hiding complexity, making a two-sided conversation simpler, reducing the conversation to the important elements is the essence of abstraction. The first core value of Enterprise Services Architecture is to increase abstraction where it makes business sense.

The second core value of Enterprise Services Architecture is modularity. The strategy is to manage the complexity of a large system by breaking it into a number of smaller parts. Each part should contain enough functionality to do some useful portion of the work of the system.

The third core value of Enterprise Services Architecture is to connect the modular parts (the components) using services. A service is a description of a specific interface to a component that performs some function. A service for customer lookup might take last name and first name as input parameters and return a list of customers who match. The description of what can be asked of a component is a service interface. The code inside the component that does the work is a service implementation.

The fourth core value of Enterprise Services Architecture is loose coupling. Loose coupling means that the service interface exposes as little as possible of the service implementation. The complex work done by the service implementation should not be visible to the out-

side world. Hiding such complexity makes reusing the component much easier and reduces to a minimum the unnecessary dependencies that make a monolithic architecture hard to change.

The final core value of Enterprise Services Architecture is an emphasis on design that is incrementally improved. At the beginning of the creation of a set of components, the architect must sort out all of the functions of an application and then decide which services each component should have and how they will all work together to get the work done. In a typical design, some components orchestrate the work of other components. Some components represent important data such as the customer object. Other components may manage a resource such as an RFID scanner that reads information about products. No matter how much time is spent on a design, generally new requirements come up after implementation. It is the philosophy of Enterprise Services Architecture that the design process should continue based on what is learned in implementation so that the system is gradually improved.

Let's return to our watch example. Making the watch into a component would be the equivalent of hiding the inner workings and only allowing a predefined set of questions to be asked. What time is it? What date is it? What day of the week is it? These three questions represent services of the component. The inner workings of the watch are no longer important. The watch has been modeled as a set of three services. The inner workings should be able to be replaced as long as those services are provided. The complexity of the watch is still there, but there is now something in between, an abstraction that reduces the watch into three services.

The goal of these core values is to create a complex of components that have the minimum dependencies among them so that they can be recombined for different purposes with minimal effort. The other end of the spectrum is a monolithic system, which is not broken into modules and has many dependencies. We have spent a significant amount of energy trashing monoliths. Now we will explain why we do not like them.

## Shortcomings of Monolithic Architecture

In the context of Enterprise Services Architecture, monolithic means one time, one purpose, one way.

Most companies do not have architectures that adhere to Enterprise Services Architecture principles and they suffer for it in a variety of ways. A nest of enterprise applications that evolved without a plan tends toward unmanageable complexity. It is possible to increase automation in such an environment, but the cost is high and every additional integration increases maintenance costs and reduces flexibility.

How do monoliths make companies suffer? Tight coupling makes change expensive or impossible. Products cannot be upgraded to take advantage of the next version. The IT staff is fearful to change anything because they don't know what will break. Integrations are expensive and always start from scratch. Functionality is not reused. The answer to enabling a new business strategy is no, not yes.

How are monoliths built? First, build a solution as fast as possible without regard to flexibility or future needs. Add functionality a bit at a time. Voilà—in no time you will have trouble changing the system or adding anything to it. Vinod Khosla, a venture capitalist with Kleiner, Perkins, Caufield, & Byers, likes to illustrate monolithic architecture with the image of the Cat in the Hat balancing 20 different objects while standing on top of a ball.

The problem with current integrations between applications can be described perfectly using the watch example. Tightly coupled integrations involve looking at the complex workings of the watch on both sides, understanding them, and then building a bridge between them. Such connections are fragile and expensive to build and maintain. If the inner workings of one of the watches change, it may have a dramatic effect on many integrations. Complexity on one side mapped directly to complexity on the other side is the essence of tight coupling. The more dependencies between the two sides, the tighter the coupling. The more that one side needs to keep track of

what is going on in the other side, the tighter the coupling. In tight coupling, the impact of a change on one side is hard to predict. One has to know a lot before one can make even the simplest change.

The classic mistake that is repeated over and over in unplanned architectures is tight coupling to the data model. When a CRM system tries to read data from the HR system, the tightly coupled method reads the database directly. If there is a change in the database structure that affects what is being read, the program reading the data must be changed. However, the person upgrading the HR system may not know that the CRM system now depends on the HR system having a certain database structure. The new schema in HR will have consequences for CRM, a problem that may be uncovered during testing but probably won't be found until the new version is in production.

As the number of such dependencies increases, it may become impractical and costly to upgrade an enterprise application from one version to another. Taking advantage of new opportunities to integrate with suppliers or offering direct access to customers can be difficult. Reusing portions of an application is difficult when the functionality of the enterprise is trapped in large monolithic chunks. Composite applications, one of the main advantages of Enterprise Services Architecture that we will explain in detail below, are expensive to build in this context and only add to the complexity and inflexibility.

The worst effect of a rigid, inflexible, and expensive to maintain architecture may be the way that it shuts down innovation and creativity. When people are fearful of changing anything because the unintended consequences may be so devastating, they are unlikely to propose new ways of doing business that require system changes.

## **Components and Services**

Components and services are the cure for monolithic architecture. Given the previous discussion, Enterprise Services Architecture can be described simply as a path away from a tightly coupled, mono-

lithic architecture toward one that is loosely coupled and based on components and services. The key challenge is deciding what components to create, what services they should offer, and how they should be connected. To explore these issues, we will take a look at the structure of enterprise applications.

In a company of almost any size, enterprise applications are the fundamental building blocks. All have a similar structure that is generally described through the following layers of the application stack:

*User interface layer*

Manages communication with the user

*Process layer*

Keeps track of where the user is in the process, and orchestrates the work of the services and objects to get the work done

*Services layer*

Performs the work of the application by extracting data from objects, performing some useful function, and then returning the result

*Object layer*

Contains the objects that are collections of data and services for maintaining and performing basic transformations on the data

*Persistence layer*

Stores the data from the objects

These layers are illustrated in Figure 1-7.

Components, which are constructed out of these elements, consist of objects, services, and processes that are related to each other. Generally, a component aggregates many of these elements and communicates with the outside world through a simplified set of services that constitute the abstraction of the component. The scope of a component can be very large or quite small. Several applications might work together to provide one component. A single application might be broken into many different components. One component might create a unified view of what's going on in many components.

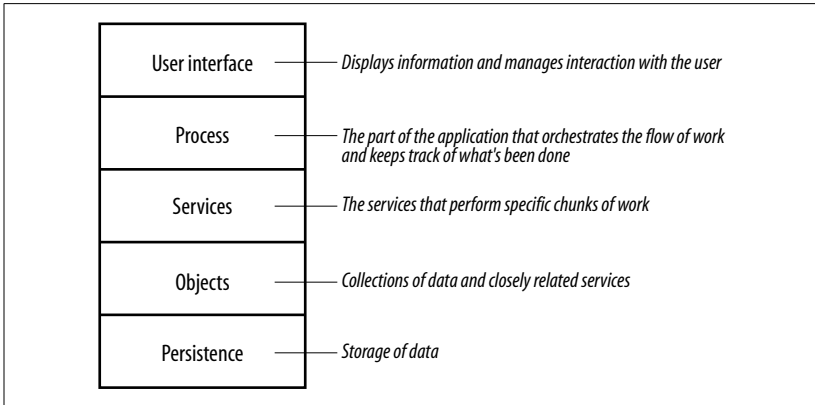


Figure 1-7. Application Stack

Services connect one component to another and exist at many levels. The methods on objects can be thought of as services. A web service that allows a name lookup can be a service. An enterprise service may be an aggregation of a set of web services that performs a common function such as controlling the process of creating a new customer record.

One of the key questions that this book attempts to answer is how to sort out the elements of a firm's infrastructure and decide where and at what level of granularity components should be defined.

So what are the core values of Enterprise Services Architecture? Enterprise Services Architecture is in favor of increasing abstraction, componentization, and loose coupling using services, all under the umbrella of a comprehensive and incrementally evolving design. Enterprise Services Architecture is against monolithic aggregation of functionality, tight coupling, and unmanageable complexity. Enterprise Services Architecture seeks to create practical business advantages rather than uniformity or consistency for its own sake.

Seasoned technologists and businesspeople will have had enough of this discussion by now, even if they are sold on the core values of Enterprise Services Architecture. As the old pros know well, “better” in the world of IT does not mean technology that has a more prestigious computer science pedigree or a more perfectly designed abstraction. The best technology from an IT perspective produces

the most business value and helps a company win in the marketplace. We now turn to the issue of whether the core values of Enterprise Services Architecture can get that job done.

## The Business Value of Enterprise Services Architecture

The starting point for Enterprise Services Architecture at most companies is a relatively large base of application functionality. In the past 10 years, enterprise applications have extended automation to parts of the organization that previously did not have support for their operations. The mainframe era focused on centralized applications for financial information. Customer relationship management, sales force automation, and supply chain management barely existed. Now that enterprise applications have extended their reach throughout the enterprise, the new direction for automation is across the existing applications.

The new functionality added as a company implements Enterprise Services Architecture will likely be composed of parts from existing systems and will focus on the following goals:

- Automation of cross-functional processes that span the boundaries of the organization and of existing systems
- Support for strategic processes that require flexible workflows and integration of collaboration and unstructured information—such as documents and spreadsheets—with transactional systems for finance and operations
- Expansion of the existing user base for enterprise applications within the organization by extending access and improving support for specific roles through focused user interfaces
- Tighter integration with systems of suppliers and key partners
- Direct access for customers, suppliers, and key partners
- Increased support for targeting specific market niches

- Improvement of change management processes such as mergers and acquisitions or program management

This new frontier for automation will provide important benefits. Companies will gather more data, think faster, and integrate decisions with actions. The flexibility of the IT infrastructure will allow companies to reinvent their businesses with less cost and to extend their value network more cheaply.

While all of these goals could theoretically be met with existing technology, as a practical matter, the integrations turn out to be brittle and expensive and they are tightly coupled in a way that we will explain later in this chapter. Tightly coupled means that developers who build integrations between enterprise applications of the kind shown in Figure 1-8 must understand all of the complexity of the monolithic enterprise applications being connected. This prerequisite increases the expense and reduces the flexibility of the integration.

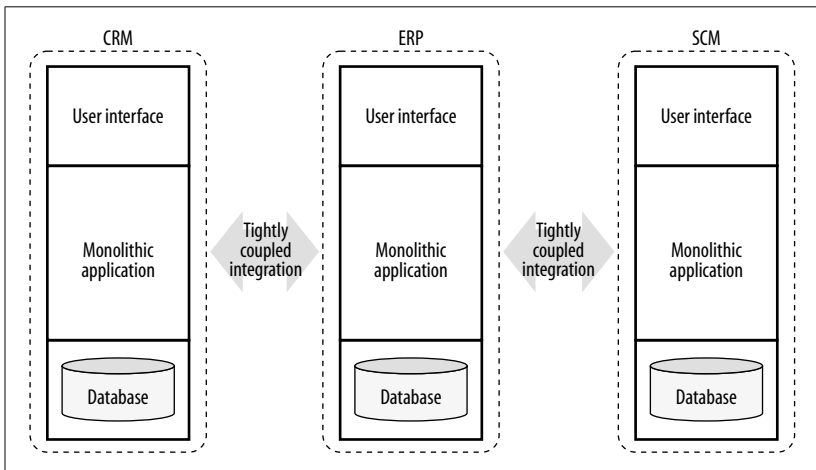


Figure 1-8. Typical enterprise architecture

## Composite Applications

One new type of system built on this architecture is called composite applications. Packaged Composite Applications sit on top of an Enterprise Services Architecture platform layer, a software product

that creates components out of existing enterprise applications. *Packaged Composite Applications* (O'Reilly), the first book in this series, explored the nature and implications of the composite applications paradigm.

One of the most important aspects of Enterprise Services Architecture is the new kind of automation that composite applications enable and make affordable. Because a composite application sits on top of all of the existing applications that have been made into components, it now can ignore the boundaries between the underlying applications. Indeed, they may not even be visible to the composite applications. This cross-functional automation of processes easily spans application boundaries. Components from transactional systems that keep track of records in databases can be combined with components that manage unstructured information found in documents, email, and discussion groups. Collaborative activity of a wide group of people such as email, discussion forums, or file sharing can be centralized and coordinated. New relationships between information from a diverse set of underlying applications can be managed and stored.

The component-based applications structures we explain below provide examples of how components in Enterprise Services Architecture help create each type of application.

For example, companies will think faster because, by nature, composite applications rapidly assemble critical information for strategic decisions. Composite applications can reach through components provided by an Enterprise Services Architecture platform into the distinct enterprise applications, pulling out the information needed to describe the current state of a company's business. Instead of having to wait days or weeks for manual rollups, composite applications can make this happen instantly, and more effort can be spent analyzing decisions than gathering information. Composite applications also facilitate distributing and collaborating on strategic information (see Figure 1-9).

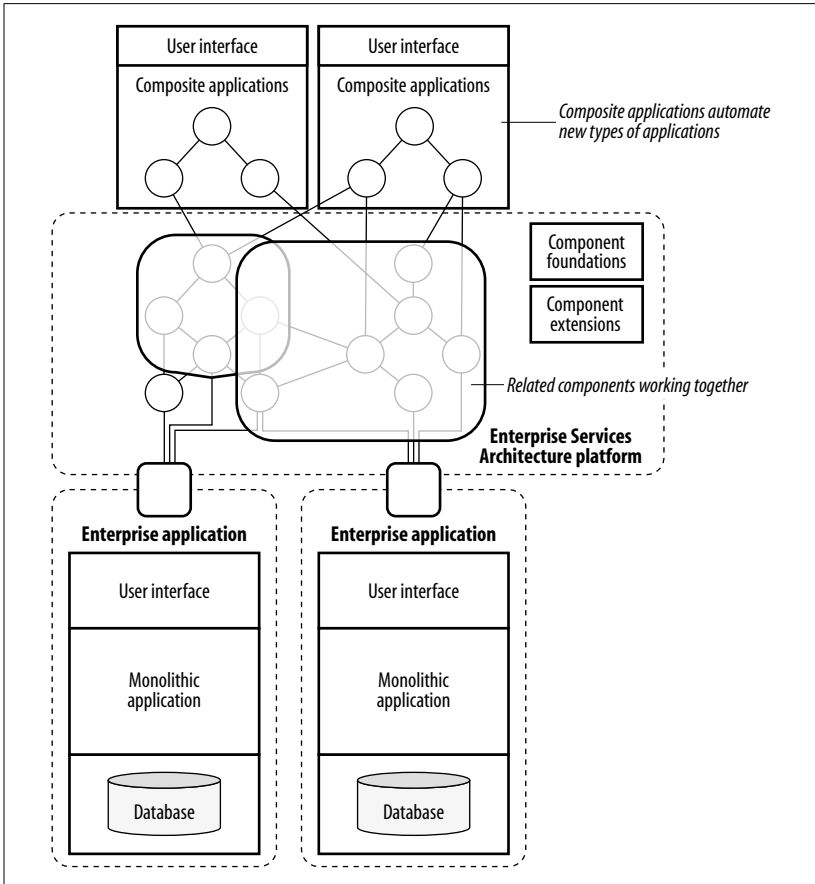


Figure 1-9. Composite applications

When decisions have been reached and action needs to be taken, composite applications built on an Enterprise Services Architecture platform go beyond read-only views of information in a variety of systems. The components built on top of existing systems can allow invocation of functionality in the underlying systems. If information in a sales order is found to be incorrect, a user doesn't have to then go to the system of record and find the data field in question. Corrections can be made immediately from the composite application at the time of discovery. A variety of underlying enterprise applications can be invoked and coordinated from a composite application.

Enterprise Services Architecture also increases the instrumentation of a business. The number of available metrics increases, as does the ease of getting them. Components organized to implement an order fulfillment process can provide reporting at each step of the process. When such a process is implemented in a tightly coupled set of objects within a monolithic legacy or enterprise application, information about the process is limited. Legacy systems provide coarse-grained information, usually just the time the order was started and the time it was completed. In a component-based composite application, the metrics are more granular and include details such as the time required to take the order, the time it took to prepare the order for production, the time it took to assemble all the parts needed to fill the order, the time it took to prepare the order for shipping, and so on. Better information highlights inefficiency and drives out politically motivated finger-pointing.

When it comes to supporting new supplier relationships, the components of the composite application provide a clean level of abstraction for the automation of the relationship as shown in Figure 1-10. Suppliers in essence become defined as a component. Supporting new relationships requires designing the component and determining what objects will be managed within it and what services will be used to represent the supplier relationship. The component may be implemented within the boundaries of a company's IT infrastructure or remotely. (Web services, which we discuss later, make implementing remote components easier than ever before.) Supplier relationships are commonly automated using EDI technology in which an agreed upon document format is transferred between two companies and carries the information needed to support the relationship. Although transferring documents back and forth is a powerful technique for coordinating the behavior of systems at both companies, it is a paradigm that smacks of the world of batch processing. In a world in which applications are facing customers or highly paid staff instead of data entry clerks, users need real-time links to corporate business partners, a crucial benefit of the component-based approach. Imagine if Amazon.com could not tell for sure if a book was available but had to get back to you later after the EDI

messages had a chance to flow back and forth between companies. Supplier relationships are increasingly trending in this direction.

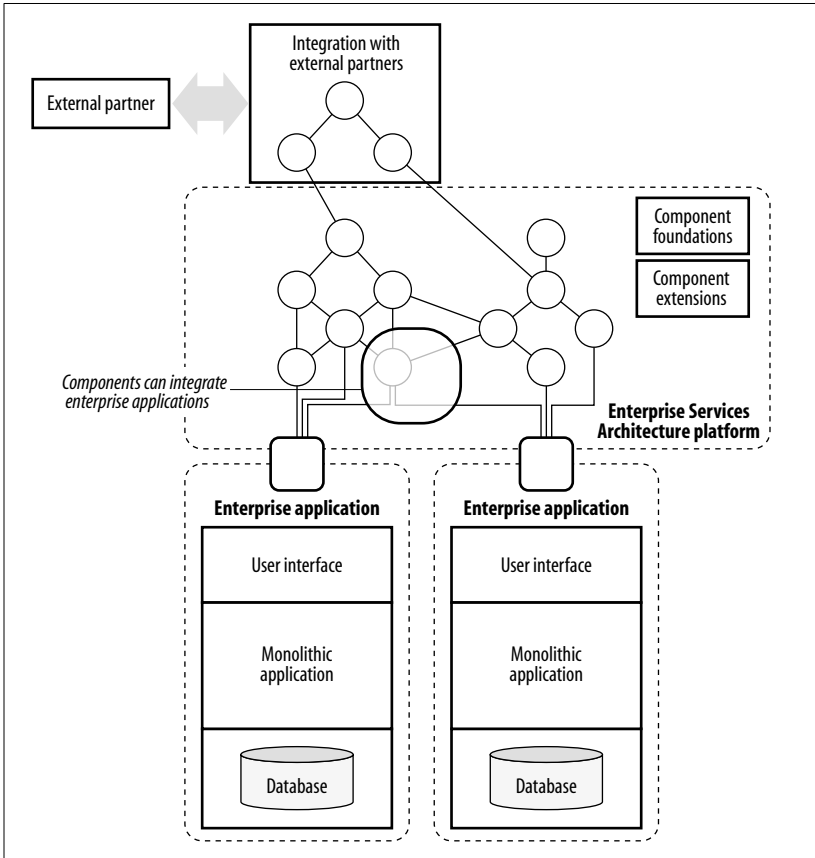


Figure 1-10. Component-based integration

Composite applications expand the reach of the underlying enterprise applications to new populations of users, as illustrated in Figure 1-11. The typical enterprise application is primarily transactional, meaning that records in databases that represent specific financial or organizational transformations are the focus of the application. An order for a product or hiring a new employee are classic examples. The users that interact with these applications are primarily administrative staff with specialized roles. In a composite application world enabled by an Enterprise Services Architecture

platform, components from all different sorts of applications can be reassembled to meet the needs of a broader cross-section of a company. As a result of expanding the reach of the application, more people in a company get the benefit of the information and the functionality of the underlying enterprise application. This increased reach is enabled because the components span the available information and functionality, but also because development is affordable. The cost of assembling a new composite application is much lower than creating a traditional application. Assembling services from a set of components is much more productive and less expensive to maintain than creating a traditional application with languages like Java or C#. Using a process-modeling approach to configure the behavior of an application or to orchestrate many components offers the possibility of even more flexibility. Process modeling is a method of controlling the behavior of an application in a simplified manner that is much less difficult and expensive than programming.

The lower cost of assembling applications is not merely a matter of development efficiency. When the cost of automation is lower, so is the cost of creating customized products or services to address a market niche, which can be a significant factor in increasing the number of niches that can be profitably addressed.

Composite applications also improve on existing enterprise applications through support for processes with a fuzzy definition. Automation of a process in traditional enterprise applications tends to be rigid. The automation of cross-functional and strategic processes tends to focus on processes that are less precisely defined than the process of filling an order. Making a decision about an M&A transaction, for example, involves identifying potential target companies, evaluating their suitability for acquisition, narrowing down the list to a smaller set of targets, negotiating the transaction, performing due diligence, and then executing the merger. Each of these steps involves information gathering and analysis by a large group of people inside and outside the company. Parts of the process may be rigid while other parts may involve tracking the collaboration in documents, email, discussion lists, and video conferences. Part of the

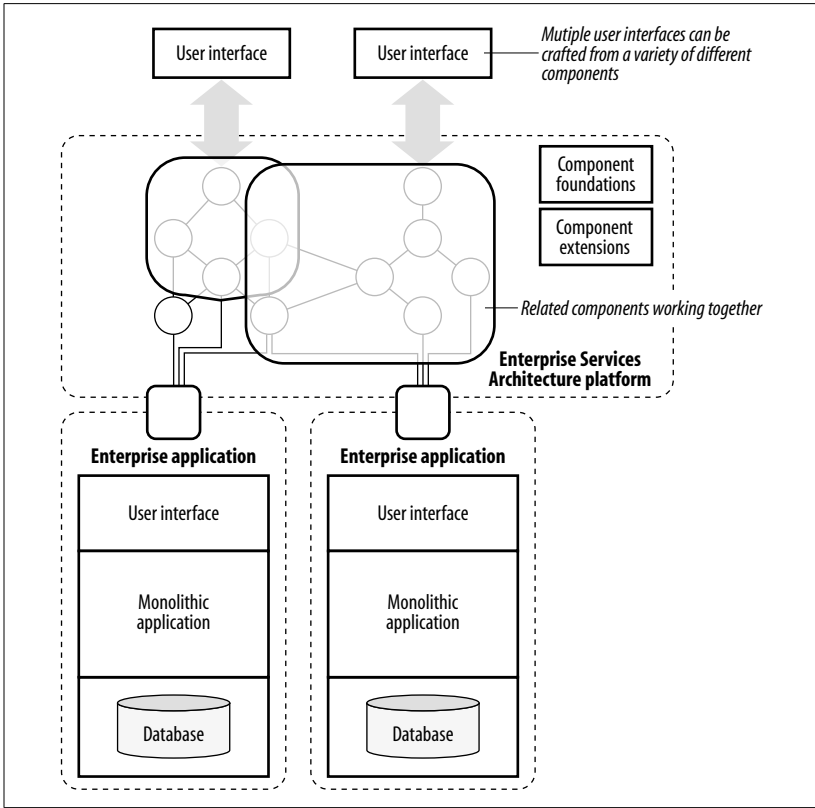


Figure 1-11. Targeted user interfaces

information for a merger resides in transactional applications while other parts are found in unstructured documents and spreadsheets. Composite applications' ability to automate the backbone of a process and fill in the gaps with collaborative functionality greatly expands the potential footprint of automation.

These beneficial factors are the mechanisms of change that result in greater flexibility, an increased scope of tactics, reduced cost of change, and the expansion of automation in an era of composite applications based on Enterprise Services Architecture platform components.

An example from the beverage industry puts many of Enterprise Services Architecture's benefits into perspective. A global beverage maker started noticing that dance clubs accounted for a significant

proportion of its sales. It turned out that one of their specialty spirit drinks was being mixed with a soft drink to create soda pop with a kick. After studying the market, the company determined that there was a niche opportunity to sell a premixed version of the drink to a small but significant population. The problem was that it would never work out financially if the company addressed the market in traditional ways because their traditional marketing and distribution cost structure was based on launching a nationwide product with a potentially massive audience, not a niche market.

To address the market, the beverage company created a streamlined process that involved tighter and lower-cost coordination with their advertising agencies and other marketing support firms and an altered relationship with their distributors that allowed for smaller quantities to affordably move through the supply chain. To automate this process from beginning to end, all parties had to adopt a more efficient Internet-based automation of the advertising and marketing programs and move smaller quantities through the distribution system. As a result, a market niche was served with a targeted product. Further, the company put processes in place that would open up other niche markets. This transformation is similar to the way that composite applications allow companies to serve smaller niches of users because such applications are cheaper to create and maintain.

## **Process and Compliance in Implementation**

It has always made sense to have a long-term vision for corporate IT architecture. In the past, this vision has taken the form of presentations that explain “where we are going.” Frequently these slides are so vague that it is impossible to disagree with them (“We are going someplace good”). In other cases, the plan stretches out for 10 years or more, making the vision so distant that it is ignored (“It will take a long time to get where we are going”). Enterprise Services Architecture transforms this nebulous cloud of architectural intention into a much more specific set of steps that allow a company to determine

how far away each of its systems is from the desired level of componentization. Enterprise Services Architecture makes it quite clear exactly how we will get “where we are going” and provides milestones for progress along the way.

## The Enterprise Services Architecture Process

For most of the history of IT, the primary goal has been internal efficiency. The current generation of enterprise applications are precisely aimed at that goal. The next wave of change for IT will be driven by an increasing demand that companies work better with the marketplace, as Andy Mulholland points out in the Afterword to this book.

Whether a company believes that Enterprise Services Architecture is a good idea or not, its IT house will have to be in order to take advantage of advanced interaction with suppliers. However a company believes that it will thrive and grow, its IT strategy will need to be tightly synchronized with its business goals. It is at this juncture, between the synchronization of business and IT strategy, where most architectural plans fail.

The mistakes are varied but numerous and deserve a book of their own. But to get the plan right, the company must have complete and justified answers to a variety of questions:

- Where should we be on the technology curve? Should we be using the most advanced, bleeding-edge technology or should we be using the most stable, proven technology?
- What is our core competency? How and where are we differentiating ourselves from the competition?
- What changes are taking place in our marketplace and how are we planning to adapt our systems to meet new challenges?
- What are the most stable and the most volatile parts of our business and how will our IT systems support the needs of each part?

These basic questions are just a start and Enterprise Services Architecture is silent on all of them. Enterprise Services Architecture has nothing to say about what a company's core competency is or where it should be on the technology curve. Enterprise Services Architecture comes into play after you answer the last two questions about the marketplace and the stability of various parts of the company. Once a company decides where it is likely to need flexibility, integration with external partners, and optimized processes, Enterprise Services Architecture provides a way to get there.

Chapter 6 provides detailed discussion of these issues, but here's a high-level view of the game plan. Imagine that each system in your architecture is a component and define what services it provides, what data it owns, and how it interacts with other systems. When analyzed in this way, some systems will be just fine as they are. A payroll system, perhaps one that is outsourced, is probably just perfect as a component in its current state. Information describing what everyone should be paid is fed in, and checks and tax calculations emerge. Most companies will not find a strategic advantage in changing this.

A thorough analysis of existing systems reveals the important data elements that exist in multiple locations as well as the systems that will support the company in an evolving marketplace or in meeting changing customer needs. Where flexibility and optimization can be anticipated, Enterprise Services Architecture's component approach can lower the cost of change. Where customization of enterprise applications or the construction of new customized applications is required, using the Enterprise Services Architecture component approach can dramatically increase the value and usefulness of this investment. Some components will be large, and Enterprise Services Architecture provides an ecosystem so that the services they provide can be reused and recombined to create new composite applications to extend automation to strategic and cross-functional processes. Enterprise Services Architecture ensures that every step in "where we are going" provides more power to the enterprise and increased business value. Every dollar spent on Enterprise Services Architecture

should put a company further ahead of its competition and in a position to more completely leverage existing technology.

The key to proper application of Enterprise Services Architecture is finding the few processes that are likely to be in need of componentization. Key data objects that show up over and over number in the tens, not the hundreds. Companies rarely try to differentiate themselves in every aspect of their operations. Enterprise Services Architecture provides the structure so that investment in flexibility can be focused on a certain aspect of a company's operations in a way that allows the finer-grained components to work in conjunction with macro-level components. The Enterprise Services Architecture blueprint provides a way for this focus to shift over time with minimum disruption and maximum reuse.

If a company's judgment about itself is correct and it invests in the right sort of componentization, it should result in a good position from which successful combinations will result.

## Compliance

The notion of Enterprise Services Architecture compliance makes objective analysis possible. Enterprise Services Architecture compliance is a hierarchy of abstraction that can be applied to any individual system, to a group of systems, or to a company's architecture as a whole. Each level of Enterprise Services Architecture compliance is defined by a combination of how completely a company understands what it wants from a system and how well that system is able to deliver it. Much of the time when we say *system*, we will actually be talking about one application, although a system could be comprised of many applications working together or could be a part of an application. By *objects*, we mean data and closely related services.

The levels of Enterprise Services Architecture compliance are:

### *Level 1: The Big Think*

Understanding and documenting the important data, objects, services, and processes that a system provides.

*Level 2: Data Services*

Read/write access to the objects of the system in a way that maintains the consistency of the application.

*Level 3: User Interface Abstraction*

The ability to separate the user interface from the rest of the system so that the services can be used as components in composite user interfaces or by other systems.

*Level 4: Loosely Coupled Components and Services*

The services and objects are grouped into components that are designed so that they can be loosely coupled.

*Level 5: Process Control*

Processes are exposed and configurable to allow the behavior of a component to be easily changed.

The purpose of this scale is to assist in categorizing what sort of componentization is needed for different parts of a company's architecture. Applying Enterprise Services Architecture is the process of understanding where your business is likely to need the agility, flexibility, and reduced costs of integration and change that result from increasing abstraction through a componentized architecture.

In Chapters 4 and 5, we closely examine what each of these levels means and how the levels relate to the current state of enterprise application systems. In Chapter 6, we analyze different ways of deciding on the right Enterprise Services Architecture compliance level for each of your systems, and in Chapter 7 we provide a road map for execution.

It is reasonable, and perhaps wise, to be skeptical of the value of this sort of componentization. Flexibility for its own sake is a losing proposition. In the next section, we will review the process of how a company should analyze where the componentization of Enterprise Services Architecture might make sense, but before we do, an example from the history of SAP might help illustrate the value of the clean separation of layers that Enterprise Services Architecture is all about.

In the early 1990s, SAP was completing a four-year project in which its R/2 ERP system was being rewritten to take advantage of the more advanced partitioning capabilities of IMS and to further extend the functionality of the suite. SAP applications are separated into the application code, which is written in the ABAP fourth generation language, and the Basis layer, which is a componentization of the services provided by the operating system and database. The development of the R/3 Basis layer had been taking place on Unix workstations and was going to be ported over to the IBM systems. The ABAP development of the R/3 applications was taking place on the IBM mainframes. The plan was to move the R/3 Basis layer to the IBM systems to complete development of R/3. When the debugging began, it turned out that the development tools on the IBM systems had some severe limitations that prevented any sort of progress. The practical power of abstraction revealed itself.

Peter Zencke, a senior SAP executive who is now a member of the executive board, suggested that instead of moving the Basis code from Unix to IBM, why not move the ABAP code from IBM to Unix? The fact that debugging could proceed then on Unix would allow the company to show R/3 at the upcoming Hanover Fair which was only six weeks away. Many long nights later, this approach proved successful. The availability of R/3 on Unix transformed SAP and—along with many other features of the software and the breadth of the range of its applications—set the stage for the company’s dramatic growth in the rest of the 1990s.

The lesson related to Enterprise Services Architecture is the flexibility that a cleanly defined abstraction gave to SAP to pursue a completely new business strategy. If the layers had been intertwined or tightly coupled to IBM software, then SAP would be much a smaller company than it is today.

## **An Industry Grows Up**

The next step for the IT industry is the componentization of applications. Componentization will occur gradually as vendors figure out

the right way to use web services and all of the other standards and tools at their disposal to create components out of their products that can meet the needs of their customers.

The IT industry currently faces the end of a youthful period in which each era opened virgin frontiers for enterprise applications or functional platform component systems. Dynamic change and innovation, of course, will still occur, but the existing footprint of IT infrastructure is so large that the first question that any new technology will have to answer is how well it works with existing systems. We look at how the maturation of the industry will affect technology vendors, the enterprise value chain, and the world at large in Chapters 8 and 9.

The forces driving standardization will result in a new battle for standards followed by a continuing wave of commoditization. For example, the initial conflict will relate to which vendor will define the basic components that present the functionality of an application such as corporate accounting. Once one or two vendors have proven dominant, all other related applications will write their software to address the winning component and service models. Other vendors will then imitate the standard components, which could lead to commoditization of the applications.

Vendors will expand their functionality to combat commoditization, but for mature applications, this will prove difficult. Once corporate accounting has been defined in a well-designed set of components, there is no longer significant functionality to add as a differentiator.

Vendors will instead differentiate themselves with either comprehensive component offerings that meet a wide variety of generic business needs or with offerings that effectively target a vertical market. Companies that provide component breadth or depth will become the equivalent of manufacturers. More than that, they will become service integrators. They will outsource more of the standardized processes, aggregate component services from multiple vendors, and package them in the form of innovative customer solutions. They will have a sophisticated understanding of the components they need

to support their core processes, will purchase or outsource those that are commodities, and will create a flexible system of components to allow strategic flexibility, optimization of the value chain, and customization of products and services to serve specific customer needs. In other words, the companies that succeed will understand how to map their business needs from the IT infrastructure onto a set of components.

The goal of this book is to provide a thorough explanation of Enterprise Services Architecture, to convey a useful way to think about the deeper problems of enterprise computing, and to teach companies how to chart the course from where they are to a place that opens up a powerful sequence of moves that translate into success. Chapter 2 illustrates this architectural paradigm with some examples of Enterprise Services Architecture in action.