European Journal of Information Systems (2000) **9**, 25–35

www.stockton-press.co.uk/ejis

The adoption and design methodologies of component-based enterprise systems

M Fan, J Stallaert and AB Whinston

Centre for Research in Electronic Commerce, Department of Management Science and Information Systems, The University of Texas at Austin, Austin, TX 78712, USA

One of the major investments of information technologies in large companies in the past decade has been the enterprise system. Although the enterprise system has the advantages of managing and integrating almost all of the business processes in the whole company, there have been strong criticisms that the enterprise system often imposes its own logic or business process on a company and lacks flexibility and adaptability in today's dynamic business environment. The goal of this paper is to outline a new approach in enterprise system development. We analyse the factors that affect the adoption of enterprise systems. Market and business changes, and advances in information technologies call for a more flexible, open, and scalable enterprise architecture. We describe the process that Dell Computer Corporation took in adopting its component-based enterprise system architecture. The Dell example has demonstrated the importance of fit between business information systems and fundamental organizational dimensions of the company including strategy, business environment, and organizational structure. We also discuss the design methodologies for component-based enterprise system design. We take a coordination perspective, both at the software level and the organizational level, in addressing the design methodologies for component-based enterprise system development.

Introduction

As we end the twentieth century, the changes that are taking place in the business world are accelerating. With increasing competition and shortening product cycle, many large companies are seeking solutions by investing heavily in information technologies to gain competitive advantages. One of the major investments of information technologies in large companies in the past decade has been the enterprise system. These huge software packages can manage almost all of the business processes in the whole company, including financial and accounting, manufacturing, logistics, human resources, and customer service. Information generated in all of the business processes can be kept, integrated, and managed. Although current enterprise systems bring seamless data integration throughout the company, they do so at a heavy price. According to Davenport (1998), one major criticism toward enterprise systems is that they tend to impose their own logic or business process on companies. The system may push a company toward full integration even when a certain degree of business segregation may be in its best interest. With a market of around $10 billion per year, enterprise systems deserve a detailed analysis.

The goal of an enterprise system is to help companies streamline their business processes. In developing or adopting an enterprise system, there are two major factors that must be considered: technology and business itself. First, as information technology (IT) is quickly changing the way companies do their business, technology itself is also moving forward in an amazing pace. The state-of-the-art client/server architecture is being replaced by the more versatile Web-based distributed object computing framework. With the growing importance of electronic commerce, the next generation enterprise systems have to be deployed globally through a universal communication network. To enterprise systems vendors, it is a great challenge to design the architecture for the next generation enterprise systems that are open, secure, scalable, and adaptable. Second, the goal of an enterprise system is to serve the underlying business. Therefore, the system has to fit the overall organizational context. Facing globalisation, increasing competition, and rapid technology changes, companies have to innovate continuously in order to succeed in the market-place. Companies can no longer assume the status quo as the legitimate way of doing business. Rather, they must have high flexibility, ranging from the capability to tailor products and services for customers to the ability to expand production rapidly when market opportunities suddenly arise (Ittner & Kogut, 1995). In a dynamic environment, the enterprise system, which represents the data flow and workflow of the company, should evolve with the business processes. Sticking to today's 'best practice' and relying on a single software company to provide a

standard business process is a recipe for failure in tomorrow's competitive marketplace.

This paper aims to present a new framework for tomorrow's enterprise systems. It focuses on the technological as well as the organizational aspect of a global enterprise system. After briefly analysing the market and technology of today's enterprise systems, the paper discusses the cutting edge technologies and their impacts on the development of the next generation enterprise systems. Next we address the adoption and selection of enterprise systems from an organizational perspective. We describe the implementation process of its enterprise system at Dell Computer Corporation and study the issues of information systems fit and organizational performance. Further, we discuss the methodologies in developing a component-based enterprise system. We focus on the issues of coordination at both the organizational level and the software level. This is followed by the analysis of future trends in component-based enterprise systems development. The last section is the conclusion.

Enterprise systems' market

Even excluding the consulting expenditures, today's enterprise system industry is a $10 billion business (Davenport, 1998). According to Advanced Manufacturing Research, an enterprise application research firm, the enterprise software sector will grow to more than $19 billion by 2001. The biggest players in the market are SAP, Oracle, JD Edwards, PeopleSoft, and Baan. Their total market share is 62% (Figure 1). SAP, a company based in Walldorf, Germany, has 30% of the total market share. We will first look at the architectures for current enterprise systems, particularly SAP and JD Edwards.

SAP has two main products: R/2 and R/3. R/2 is the mainframe product while R/3 is the client/server version. SAP R/3 can have a two-tier or a three-tier client/server architecture (Figure 2). With a two-tier architecture, the business applications can either locate on the database server side or run together with the presentation servers (clients). The latter is suited for applications that are process-intensive on the client side. The R/3 system provides different business modules including asset management, financial accounting, human resources, production planning, sales and distribution, materials management, etc. These business applications are essentially business processes and logic that are imposed by SAP. Therefore, companies that implement SAP have to adopt the standard business processes provided by the software system. Some level of customisation is possible by using the ABAP/4 development workbench that comes with SAP R/3.

SAP R/3 has been a phenomenal success since its release. Companies can use R/3 to integrate data flow and access information in ways they never could before. For example, a company's inventory data can be updated automatically after each sale and the new numbers will be available throughout the organization so that the manufacturing division can make real-time production plans and executives can make better strategic decisions.

However, many companies have found that they have many problems in implementing SAP. First, business processes provided by R/3 are generic solutions for an industry or industries. Although they might be the 'best practice' from an industry point of view, the particular process may not exactly fit the specific business model of the company. Today, customers are becoming increasingly sophisticated. They demand the right product at the right time with the right price. Companies have to respond immediately to the needs of their particular customer segment. Therefore, companies are desperately seeking solutions that will enable them to develop and deploy their information systems rapidly in order to facilitate their business processes and possible changes. But companies have either to invest heavily to develop their customised application or pray that software companies such as SAP will ship out a package that fits their needs.

Second, although the client/server architecture has been an improvement compared to the earlier mainframe

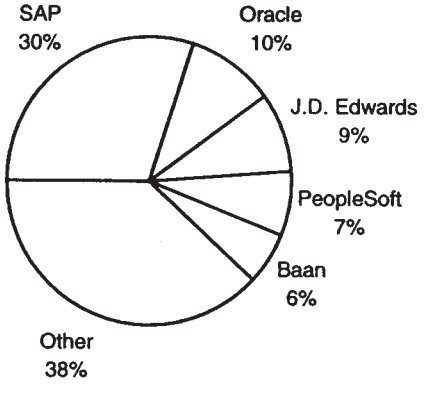


Figure 1 Market share of the major enterprise vendors. (Data source: Red Herring, 1998).

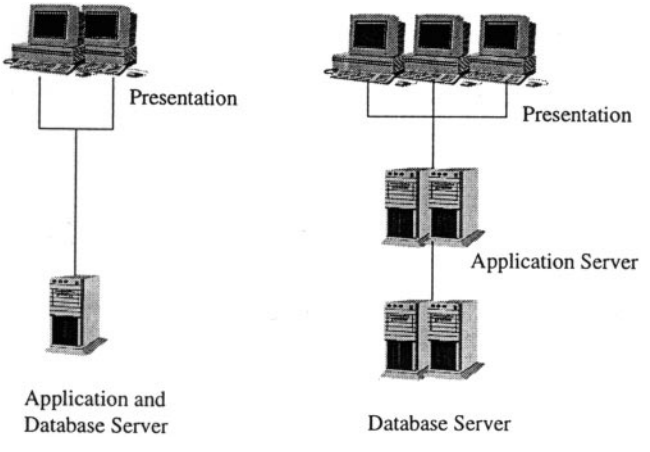


Figure 2 Two-tier and three-tier client/server architecture.

solution in terms of computation speed and graphical user interface, it is very much a centralised and exclusive system. Business logic, being on the application server or on the client side, controls production, distribution, customisation, and other business processes. Different application modules communicate with each other directly without an intermediate layer. The whole software system is a monolithic package. It is extremely difficult to reconfigure the system in order to communicate with applications developed by other vendors. As companies want to change their business processes rapidly, it requires a software architecture that is flexible and has the ability to integrate applications that could be developed by different vendors. Breaking the two-tier or three-tier client/server architecture into a distributed object architecture may be the real solution (Figure 3). We will discuss the new architecture in the next section.

SAP has realised many of the problems and has moved aggressively to improve the flexibility of its software. It plans to allow customers to select and purchase SAP software bundles rather than the whole R/3 system. Meanwhile, SAP is launching its new product, mySAP.com, by integrating its core business applications with the Web and electronic commerce.

Compared to SAP, JD Edwards has been a relatively slower mover in developing a state-of-the-art client/server system. However, JD Edwards has been very aggressive recently in applying new technologies in enterprise systems. The company recognises the limitations of the client/server model and tries to offer a product that can separate the business applications with the technology. OneWorld, the JD Edwards' product, has two modes: client/server mode and browser mode (JD Edwards, 1998). The latter is the company's vision that the future enterprise systems should be deployed through a 'configurable network' with high interoperability. Components can run from different computer platforms and interact with each other through the JDENet, the company's proprietary middleware, which supports ODBC, socket-based communication, EDI, and flat files. In the future, it will support COM, CORBA, and ALE (Application Linking and Embedding) and Idoc (Intermediate Document). JD Edwards has expended a great effort to ensure interoperability and to develop its own middleware. However, the company's core business is to develop business applications. The lack of an industry-standard middleware forces many companies to expend extra effort to solve the problem of interoperability.

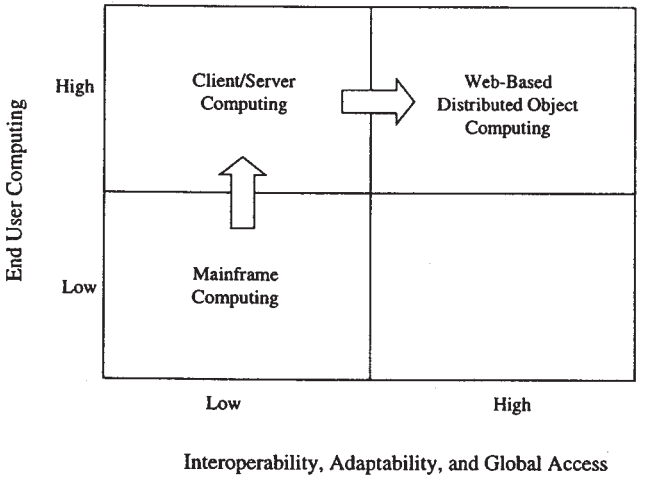


Figure 3 The evolution of computing frameworks.

Technology imperative

Rapid advances in computer and communication technologies are revolutionising the way business systems are developed. The Internet, distributed object technology, and Java are quickly changing the current computing paradigm. The convergence of the Internet-based technologies provides the infrastructure to construct complex software systems over the network.

The Internet and the World Wide Web (WWW)

The Internet and the WWW have changed the landscape of computing in a way no other technologies have done before. Together they provide the universal communication network and standard user interface that unify all different computing platforms and allow people to access information and execute applications virtually from anywhere at any time. The Internet and Web-based computing have crossed the boundaries of different functional teams, companies, and nations and provides a computing infrastructure for companies to develop integrated systems throughout the whole supply chain.

Because of the tremendous anticipated advantages of the Internet, many companies jump onto the Web-based computing bandwagon before they are actually ready. Earlier Web-based computing models predominantly used the Common Gateway Interface (CGI) approach. The focus is on information retrieval and display. The Web browser is simply a display tool. All of the computing is conducted on the server side, and the results are then converted back into HTML pages and sent back to the client's browser. This communication model is not very efficient and wastes valuable network resources. Advances in distributed object technology can overcome many shortcomings in today's Web-based applications, make Internet applications more dynamic, and provide more versatile ways to exchange structured information over the Web.

Distributed object technology

Distributed object technology encompasses not only the original object-oriented model but also component technology. Component-based system development is rooted in the object-oriented model, which represents a major

revolution in software engineering. Objects are straightforward abstractions of real-world entities. Software objects, with properties such as encapsulation, inheritance, and polymorphism, greatly increase software reusability and simplify the software development process. The component model, on the other hand, focuses on building software systems by combining and matching pre-developed software objects. The component concept is an extension of the object concept. However, the focus of component technology is not on inheritance but on the combination and integration of different software components. The benefits of the component approach range from rapid software development to increased customisation and maintainability and enhanced software quality. Components with well-defined interface can be quickly combined and assembled to form a complex application system.

With the component model, some software companies can focus on individual component development and leave system assembly tasks to others. With increasing specialisation, software producers can enhance their productivity. Businesses can purchase components from different vendors or develop more specialised components in-house in order to assemble a highly customised enterprise system. In a dynamic business environment, when companies are changing their business strategies and processes, they can simply replace old components with new ones. The result is easy software maintenance and guaranteed minimum delay between business information systems with business processes. CORBA, Enterprise JavaBeans (EJB), and COM are some of today's most popular component technologies. We will briefly describe the features of these technologies.

CORBA

CORBA is a distributed object-oriented computing standard proposed by the Object Management Group (OMG). It provides an infrastructure allowing objects to communicate independent of the specific platforms and techniques used to implement the addressed objects. The core of CORBA is the Object Request Broker (ORB), which guarantees portability and interoperability of objects over the network, a heterogeneous system. CORBA provides an Interface Definition Language (IDL), which can be used to define object interfaces independent of implementation. Clients and servers in a CORBA environment do not need to know the details of each other's implementation. IDL tells which methods can be invoked on an object. With CORBA, users can create object interfaces and implement objects using different programming languages and achieve real interoperability. However, the complexity in CORBA implementation might be one of the reasons that distributed object technology has not gained much ground in the industry. In addition, client applications implemented in programming languages other than Java cannot take advantages of the WWW.

Java and EJB

Compared to CORBA, Java provides an easier implementation of Web-based distributed object technology. To a large extent, it provides the missing link between Web-based computing and the object technology. As an object-oriented language, Java is easy to understand, easy to learn, and easy to implement. With built-in networking capabilities, Java makes network computing a much easier task. Now network computing becomes one of the most fundamental parts of Java computing. On the client side, Java applets can be seamlessly integrated with Web browsers. As client applications, the applets are full-fledged applications that have rich graphical user interface. Java applets are platform-independent and guarantee that applications are 'developed once and run everywhere'. As client objects, applets can interact with Java objects or objects implemented in different languages in the computer network through Internet Inter-ORB Protocol (IIOP) or Java Remote Method Invocation (RMI). This ensures the interoperability and robustness of the applications. On the server side, Enterprise JavaBeans (EJB) provides the framework to develop component-based, highly scalable, and secure enterprise systems. Under the EJB framework, pre-developed and pre-tested components can be easily combined and assembled on the application server. Client applications can access these components through the Web-server and Java Naming and Directory Services.

COM

COM, which stands for Component Object Model, is Microsoft's competing component development technology. Introduced in 1993, COM is now a mature foundation for component-based development. COM is the technology that one uses to define components and enable software interoperability. It is not a language for implementing components. Instead, developers can use C++, Visual Basic, or even Java to create components. COM acts as a shell that wraps around components and enables communication between clients and server components. Microsoft describes its distributed component architecture as DCOM (Distributed COM). DCOM makes it possible to create networked applications built from components. Compared with EJB, COM is a more mature technology since Microsoft has been shipping and supporting the software for a couple of years. The drawback is that companies have to use COM with Windows NT server, which still lags behind in scalability compared with systems such as UNIX.

With the growing interest in EJB, many CORBA vendors have shifted their focus to developing Java-based component systems. Thus, the true competition in

component development standard may lie between EJB and COM. It is hard to predict who will be the winner in the future. However, at this moment, Microsoft's COM-based model has an advantage for small- and medium-scale applications.

Technology convergence

The convergence of the Internet, the WWW, and distributed object technology defines a new era in computing. Each technology increases the value of the other technology. Taking away any of the above technologies, the impacts and values of other technologies will be significantly less. For example, the distributed object technologies have been around for some time; but with the development of the Internet, the WWW, and Java computing, the technology gets a new life. In our opinion, the transition to Internet-based computing should embrace all of these technologies.

The integration of the WWW and distributed object technology offers a solid technological infrastructure and a set of tools to develop the next generation software systems. This infrastructure has the following synergies:

- Applications can be deployed universally over the Internet and the Intranet. Users can access the system virtually from any place at any time by using all kinds of devices such as laptops, desktops, workstations, phones, or hand-held wireless devices.
- Applications implemented in different languages and running from different operating systems and platforms can communicate and share functions with each other through communication protocols such as IIOP, Java RMI, or COM. Data and information flow in the company can be seamlessly integrated.
- System development cycles will be shortened significantly by using component-based software engineering approach. Companies will able to reuse a lot of software codes.
- The resulting enterprise systems can be highly customised by using the component approach and assembling individual components that fit the business processes best. Systems can evolve with changing business strategies and processes easily.

Organizational perspective

In adopting an enterprise system, technology is only half of the picture. Organizational perspective is equally important. In this section, we begin with the case study of Dell Computer Corporation. Compared to Dell's direct sell business model, less is known about its implementation process of its enterprise system. However, from interviews with executives at Dell's IT department, we learned that Dell's decision to replace SAP with a more flexible approach in developing enterprise applications is critical in executing the company's strategies. Using our insights into the Dell case, we discuss issues in business information systems fit and organizational performance.

Dell Computer

Dell Computer is the world's leading personal computer (PC) producer. Its direct sell and build-to-order model has revolutionised the PC industry. Dell began in 1984 with its direct sell model. By bypassing the dealer channel through which personal computers are traditionally sold, Dell has eliminated the reseller's mark-up and the costs and risks associated with carrying large inventories.

Initially, Dell used different hardware systems and applications programs including Tandem and COBOL. In order to integrate fully all of the systems on an enterprise-wide basis, the company bought SAP R/3 in 1994. Immediately Dell started the customisation of SAP to fit its organizational model and expected the transition and customisation period would take as long as five years. However, after only two years of use the company decided to drop the SAP project and started to use COM as a standard in developing its component-based enterprise systems. Why did Dell make such a drastic decision after spending millions of dollars on SAP? The simple answer is that the company feels SAP is not flexible enough to fit its corporate strategy and meet its rapidly changing business processes.

We portray Dell's decision matrix in Figure 4. Competing in a very dynamic environment with ever changing technology, customer tastes, and supplier relationships, Dell strongly believes that the company has to adopt a more flexible, component-based approach in developing its enterprise systems in order to fit its overall corporate and operation philosophy. Dell Computer's goal is to become the No. 1 leader in the PC industry. The PC market is known for its rapid innovation and short product cycles. The company's strategy is to change rapidly with the market demand, to reduce production cycle, and to hold as little inventory as possible. Its IS is a strategic piece in implementing its corporate

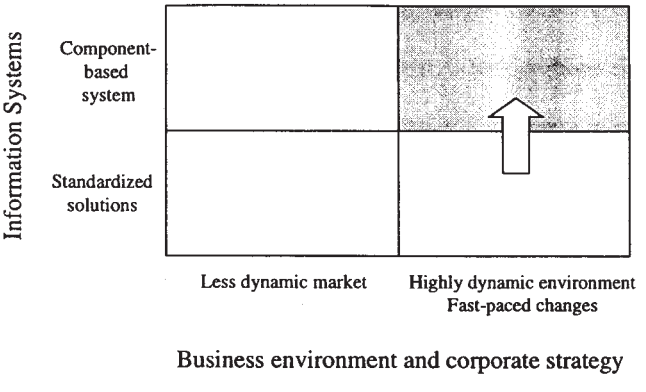


Figure 4 Decision matrix at Dell Computer.

strategies. Dell's website is its virtual market-place. Information captured through close interaction with its customers and suppliers enables the company to revolutionise its business operations by producing exactly what the customers want, virtually eliminating inventories caused by inaccurate forecasting. At Dell, information is a valuable asset, replacing warehouses and inventories. Therefore, business systems cannot be static. They have to keep changing with the changing taste and behaviour of online shoppers. They also have to adjust constantly in order to work smoothly and efficiently with the company's suppliers. In such a dynamic environment, a relatively static enterprise system such as SAP does not fit well. The following are a few specific reasons the company decided to terminate the SAP project:

- The deployment cycle for SAP is too long. The initial plan to convert the whole company's information systems to SAP would take several years, during which the technology and market-place could change a lot. For a high tech company like Dell, which competes in a highly dynamic environment, SAP simply takes too long to be deployed.
- Even though SAP allows some customisation, full customisation is impossible. Subsequently, Dell cannot make the system flexible and versatile enough to change with changing business strategies.
- Should the company fit its business processes to its information system, or the other way around? This is the big debate in deciding the company's IT strategy. Should SAP be implemented, the company would have to change its operations to fit the system requirement. But executives in Dell realised that it is not going to work for Dell. In fact, the IS and business operations become so closely tied together at Dell that many times they have moved forward together simultaneously. A large-scale monolithic enterprise system simply does not fit the organizational culture.

Dell's decision to go with component-based system development paid off. Today managers at Dell can track the company's sales, production, and distribution information in real time and make strategic or tactical decisions. The company continues to be the industry's leader in the direct selling of PC's over the Internet and has achieved a 50% growth rate in the past few years. In 1999 Dell surpassed $18 million a day in Internet sales, or 30% of its total revenue. The company is steadily moving toward its goal of achieving 50% of its sales over the Internet.

Evaluating enterprise systems fit and performance

Deciding what business systems to deploy is an important theoretical and pragmatic question. In Dell's case, even though the company's had a clear vision in its business strategy, it did pay hefty prices in the process of choosing its enterprise system solutions. Eventually the company realised its information systems have to fit the overall corporate strategy. The issue of fit between an organization and its structure, technology, and processes has been an active research area in organization theory (Thompson, 1967; Galbraith & Nathanson, 1979). The concept of fit, which is defined as the congruence between different components of an organization, is central to the theory. Organizations are packages or mosaics in which all pieces must fit together. Organizational dimensions such as structure, reward systems, and resource allocation processes must be consistent not only with the organizational strategy but also with the others.

Some researchers in IS (Robey, 1981; Markus & Robey, 1983; Leifer, 1988) have studied the fit between IS and organizational structures. In general, the fit between an IS and the organizational context is believed to be critical to the success of the system deployment. Although there are a couple studies at the individual user level (Ives *et al*, 1983; Goodhue & Thompson, 1995), research at the large-scale enterprise system level is generally overlooked. With more companies investing in large-scale enterprise systems, study in this area is imminent. The Dell example has demonstrated the importance of congruence between the IS and other components of the company including strategy, business environment, and organizational structure. Empirical studies on understanding and measuring the correlation of the structural variables along different dimensions of organizations and IS are needed in order to provide further evidence.

Dell's tremendous performance also supports the contingency theory that the 'fit' in overall organizational design results in high performance (Galbraith & Nathanson, 1979). This is consistent with the results of economic studies on the technologies and organization of modern manufacturing (Milgrom & Roberts, 1990), which suggest the presence of complementarities among the elements of the firm's strategies. Recently, studies in IS (Barua *et al*, 1996) also showed that organizational payoff of re-engineering is maximised when several factors relating to IT, decision authority, business processes, and business incentives are changed in a coordinated manner. From the Dell case, we can see that the fit between business IS and organizational context is in fact a dynamic process. Simply making IS fit the organization is probably too narrow a view. While a company's IS should be deployed to implement business strategies with minimum delays, organizations may develop new ways to produce and distribute its products because of the innovations in IT. This dynamic process of interactions between IS and organizational context should result in higher performance. Measuring the business value and analysing variations in organizational performance from the interaction of IS and organizational context will make significant contributions to IS theory as well as to practitioners.

Design methodologies

In this section, we discuss the design methodologies for component-based enterprise systems. We will first address the middleware in component-based software engineering. Current component-based software engineering literature has focused on the tools and standard for component development. In enterprise computing, putting together a complex enterprise system based on pre-developed components is not a straightforward task. Thus, we discuss the coordination problems in component-based enterprise system development, at both the organizational level and software level.

The Middleware

Component-based enterprise system development may take several approaches. For example, let us look at an enterprise system that contains the following business modules: order processing, production planning, production operation, and distribution. As mentioned earlier, the traditional client/server approach allows different application modules to communicate with each other directly without an intermediate layer (Figure 5). This might be a reasonable solution if a company is developing the whole application package by itself or if the system is relatively simple. But, for a complex system with components being developed by different vendors, this design will result in many problems, e.g. the loss of flexibility and interoperability since the communication protocols are largely platform and language-dependent.

The solution is to add the middleware as an additional layer between business components, providing generic services such as naming, directory, and communication for different software components (Figure 6). This is exactly the idea of CORBA, COM, and EJB. In the case of EJB, components implemented in different languages, can use the industrial standard IIOP or RMI to communicate with each other. With the availability of naming and directory services, components can communicate with each other in a network-transparent way; i.e. components can be accessed using location-independent identifiers, regardless of their locations.

From a distributed computing point of view, the middleware can be thought of as the distributed operating system in a multi-component environment. The functions that the middleware provides are similar to what a local operating system does for local applications. But, different from the local operating systems, which are monolithic packages, the middleware is a logical unit and can be dispersed over the distributed network. As the system gets more complicated, more services can be added onto the middleware. The basic services the middleware provides should include naming and directory, messaging, and transaction monitoring. These services can be on a single server or on multiple servers.

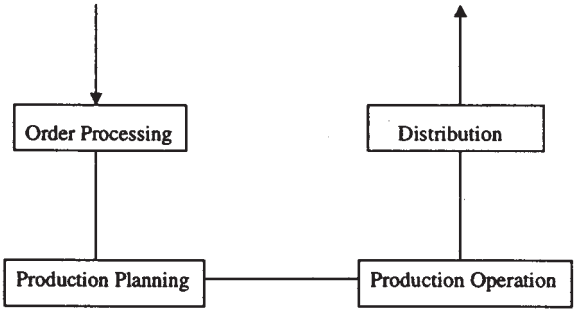
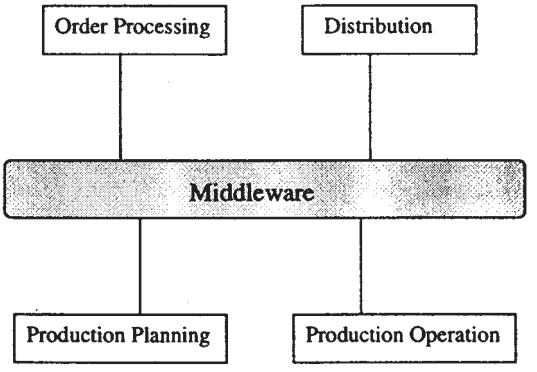


Figure 5 Traditional modular implementation.



or on multiple servers.

Figure 6 Middleware connecting components.

Coordination at the organizational level

Different from scientific computing, enterprise systems contain entity objects that represent real-world business divisions or teams. The goal of an enterprise system is to help companies integrate all business information and streamline their business processes. In order to do that, the system should be able to coordinate activities across the organization and provide decision tools for managers.

According to our recent interviews with executives and developers from the top enterprise system vendors, analytical tools, decision support, and e-commerce applications are the weak link in today's enterprise systems. Today's enterprise systems are able to capture all of the transaction data, but they seriously lack analytical tools to examine the data and help managers to make optimal decisions. It is important to note that the decision-making process in a global company with multiple product lines and several regional headquarters could be extremely complicated. Some decisions are made in corporate headquarters while many are made in a more decentralised way. It is a great challenge to the company as well as to enterprise systems vendors to develop innovative ways to coordinate the decision-making process at different levels of the organization in order to achieve the overall goal of the organization. Meanwhile,

coordination is equally important in an inter-organizational environment. A firm has to find out efficient ways to work with its suppliers and manage the whole of its supply chain processes. For example, an enterprise system for manufacturing companies should be able to solve the following operation problems:

- Decide the current production level of each product.
- Find out the volume of each raw material or intermediate product that should be procured or produced.
- Select the best vendors and negotiate the prices.

Figure 7 depicts the architecture of an enterprise system with components for analytical applications and decision support. Generic middleware provides services such as naming and directory service, transaction processing, and message transmission for functional components. Coordination mechanisms are implemented as domain-specific middleware that provides services for intraorganizational or interorganizational business interactions. Different from generic middleware servers that provide the same services for different organizations, domain-specific services vary from organization to organization, reflecting the differences of organizational structures and decision-making processes.

Coordination at the software level

At the software level, the components that form a complex enterprise system are interdependent, and sometimes they may have conflicts in terms of accessing system resources. We also take a coordination perspective in the system design. Issues such as asynchronous processing and concurrency have to be addressed for distributed component systems.

Our implementation of a distributed electronic market recognised that distributed applications are asynchronous in nature (Fan *et al*, 1999). Asynchronous distributed processing occurs when different components can participate at different times in the application process. For example, an order process system requires the transaction server to handle requests from many different clients. With synchronous processing, the transaction server has to reply to each request before it processes the next one. With asynchronous processing, the server can schedule its operations more efficiently because it does not have to reply to each order immediately. Meanwhile, the client application does not have to wait for the immediate reply in order to conduct the next task.

In an enterprise system, data are shared and can be accessed by many different business applications or components. For example, customer information is shared by order processing, production planning, sales, and distribution. Inventory information is shared by production planning, operation, distribution, and sales. Management of shared information is an important task in component-based system development since the access control mechanism may be distributed in different systems. First of all, we should specify clearly the access rights that are assigned to each application processes. A component is not allowed to access the data that is irrelevant to its process. If different components can access the same data item, it is critical to ensure that an application is free from the interference from other applications so as to avoid the inconsistencies on the same data. The most common way to implement concurrency control is to use exclusive locks. By locking the data, the application is in effect serialising the access to the data. While applying locks, we need to make sure the portion of the data to be serialised should be kept as small as possible. By applying unnecessary locks, the server operation will become less efficient. Meanwhile,

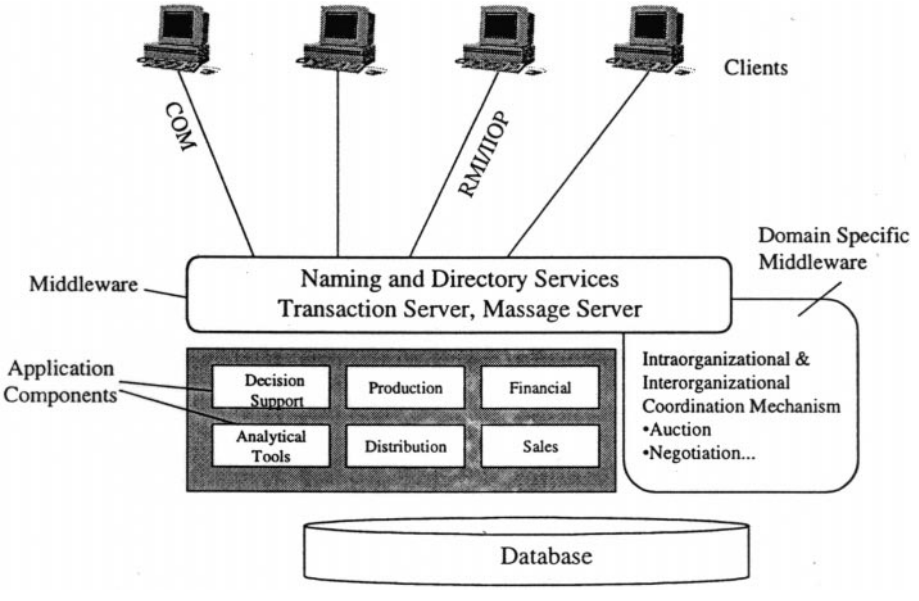


Figure 7 Enterprise system with decision support component and coordination mechanism.

it is necessary to avoid deadlocks. There should be services available to detect and delete deadlocks. There are several other design issues in the component-based system. For example, load balancing, security, and fault tolerance have to be dealt with in a coordinated approach as well.

Future trends

There are several aspects that the next generation enterprise system will improve: (1) accessibility and interoperability, (2) flexibility, (3) adaptability, and (4) real-time processing and intelligence. First, the system should be able to run on any platforms and be deployed through any devices. Applications running on different devices or located on different servers should be able to communicate and share functions with each other easily. In order to achieve that, the systems have to make full use of the synergies of Internet-based distributed object technology.

Second, the future enterprise systems will be more flexible and much easier to customise. The underlying idea is to use the market approach in software development. The monopoly model where a company provides a bundled, monolithic package will not work in the future. Rather, we will see more specialisation and more competition in the software market. There will be companies specialised in middleware and different business applications. Companies will produce components with standardised interfaces. A customer can buy the best components from the components market and customise or have it customised into an application that fits seamlessly with its business processes (Figure 8). This will lead to higher software quality and shorter development cycles.

Third, the systems will have adaptability. One of the greatest advantages of the new computing framework is its capability to deal with changes. Companies today are facing great challenges in keeping up their IS with the changing needs of the organization and reducing costs in converting one system to another. The market-oriented component approach provides a good solution. By using the open architecture, applications developed at different time periods and by different vendors can work together, ensuring that today's software and hardware investment can be protected, and any new changes can be made easily by just plugging new components. Companies can easily replace any components with better ones with great ease. The result is that the system will always support the fundamental business model, rather than the other way around.

Lastly, future systems will be more intelligent and support real-time decision-making. Software agents will become human assistants, monitoring the business operations, collecting information, and helping humans to make better decisions while real-time information analysis and decision-making will replace traditional batch processing.

The component approach is the future direction of software engineering (Figure 8). Assembling pre-developed components represents a huge advance in software engineering. There will be companies specialising in developing business applications as well as specialising in middleware development. The component model of software engineering is essentially introducing the market mechanism to software engineering. In order for the component framework to flourish, a component market is indispensable. A component market will lead to higher competition, better products, and lower costs. Eventually, this might lead to new market structures of the software industry, breaking large monopolies and generating high efficiencies.

The digital market-place for software components cannot prosper without the standardisation of software component products and market mechanisms such as quality assurance. The following are some factors that will affect the future software market.

Standardisation

In order to let software components be like commodities that can be bought and sold over the market-place, they have to be standardised. Naturally, there will be multiple markets for different software components with different granularities. For example, there will be large components such as order processing and production planning components and small components such as a spelling check. Software companies may want to construct large components from smaller components. For example, a production planning component can be constructed by combining sub-components such as production forecast, scheduling, and material requirement planning (MRP).

Customisation

The pre-developed components should allow some level of customisation so that highly customisable software systems can be constructed. This issue will become more important in developing intelligent software systems. Most of today's business processes and IS are rule-based. To a large extent, the rules are functional. However, based on the repetition of learned behaviour from past experiences, rules do not change easily. But the competitive environment requires a new set of capabilities that are not incorporated in the firm's current decision rules. Therefore, the future intelligent components will largely be goal-oriented. Companies should be able to configure the goal of the agents and subsequently induce changes of agent behaviour and the business processes.

Certification

The future software component market will exist over a public communication network such as the Internet.

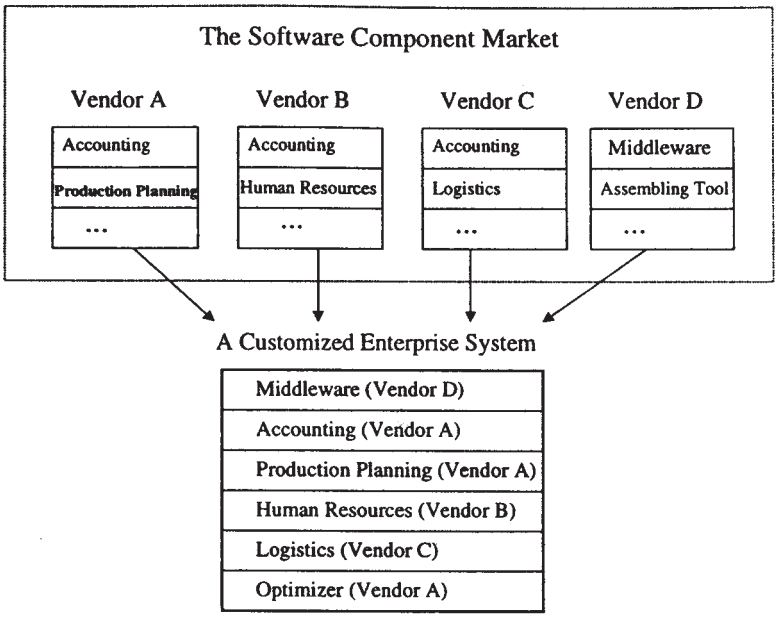


Figure 8 The component-based software engineering.

Components should be traded and delivered easily from the buyers to the sellers. One of the difficulties associated with the online market is to ensure that the software is actually produced by the claimed company. The solution is to let every vendor be certified by a third party certification authority. The vendor should also sign its digital signature on its products.

Pay by use

The software component market should allow innovative sales and payment models. Quality assurance is an important issue in the digital software component market. Distributing the software components freely but collecting revenues based on the use of the software could well be a solution to guarantee the software quality (Cox, 1996). Components with poor quality will have less and less usage. Eventually, only the companies with good qualities will survive, and the 'lemons' will be driven out of the market-place.

Conclusion

This paper has discussed the organizational and technological aspects of component-based enterprise systems. Business challenges and rapid advances in computer information technologies call for the design of a more flexible, open, and scalable enterprise system architecture. The convergence of the Internet, the WWW, and distributed object technology provides the building block for the next generation enterprise system. We described the process that Dell Computer took in adopting its component-based enterprise system architecture. The issue of enterprise system fit is an important research question to both academia and practitioners. The Dell example has demonstrated the importance of congruence between business information systems and other components of the company including strategy, business environment, and organizational structure. Future research should concentrate on empirical studies to understand and measure the fit along different dimensions of organizations and IS, and investigate whether the congruence leads to higher organizational performance.

We also addressed the design methodologies for component-based enterprise system development. We took a coordination perspective, both at the organizational level and at the software level. At the organizational level, domain-specific coordination mechanisms should be in place in order to coordinate various business activities. We believe decision support and coordination mechanisms are the missing links in current enterprise systems. Issues surrounding the design of competing coordination mechanisms for different organizational or inter-organizational processes warrant further research. At the software level, middleware should support interoperability and provide software coordination services for asynchronous processing, concurrency control, and fault tolerance.

References

BARUA A, LEE S and WHINSTON AB (1996) The calculus of reengineering. *Information Systems Research* **7(4)**, 409–428.

COX B (1996) *Superdistribution*. Addison Wesley.

DAVENPORT T (1998) Putting the enterprise into the enterprise system. *Harvard Business Review*, July-August, 121–131.

FAN M, STALLAERT J and WHINSTON AB (1999) A web-based financial trading system. *IEEE Computer* **32(4)**, 64–70.

GALBRAITH JR and NATHANSON DA (1979) The role of organisational structure and process in strategy implementation and related commentaries. In *Strategic Management: A New View of Business Policy and Planning* (SCHENDEL DE and HOFER CW, Eds), pp 249–302, Little Brown, Boston.

GOODHUE DL and THOMPSON RL (1995) Task-technology fit and individual performance. *MIS Quarterly* **19(2)**, 213–236.

ITTNER C and KOGUT B (1995) How control systems can support organisational flexibility. In *Redesigning the Firm* (BOWMAN E and KOGUT B, Eds), pp 155–180, Oxford University Press.

IVES B, OLSON MH and BAROUDI JJ (1983) The measurement of user information satisfaction. *Communications of the ACM* **26(10)**, 785–793.

JD EDWARDS (1998) OneWorld and interoperability. White Paper, JD Edwards.

LEIFER R (1988) Matching computer-based information systems with organisational structures. *MIS Quarterly* **12(1)**, 63–73.

MARKUS ML and ROBEY D (1983) The organisational validity of management information systems. *Human Relations* **36(3)**, 203–226.

MILGROM P and ROBERTS J (1990) The economics of modern manufacturing: technology, strategy, and organisation. *American Economic Review* **80(3)**, 511–528.

RED HERRING (1998) Enterprise software: ERP charges on. *Red Herring*, August, 64–65.

ROBEY D (1981) Computer information system and organisation structure. *Communications of the ACM* **24(7)**, 679–687.

THOMPSON JD (1967) *Organisations in Action*. McGraw Hill, New York.

About the authors

Ming Fan is an Assistant Professor in the College of Business, University of Notre Dame, and a Research Fellow at the Centre for Research in Electronic Commerce at the University of Texas at Austin. His areas of research include supply chain management and large-scale distributed system development based on economic principles.

Jan Stallaert is an Assistant Professor in the Graduate School of Business at the University of Texas at Austin. He received his PhD from the Anderson School of Management at UCLA. His research interests are large-scale system optimisation, financial engineering, and supply chain management.

Andrew B Whinston is the Hugh Cullen Chair Professor in information systems, computer science, and economics; and director of the Centre for Research in Electronic Commerce at the University of Texas at Austin. His current research spans various realms of electronic commerce, its emerging technologies and its impact on business protocols and processes.