System Integration and Architecture

Chapter 1

Assigning the Technology Landscape

In the last decade, major changes have occurred in information technology. Microcomputer technology drove the transition from large, centralized computing systems to client-server and personal computing. The Internet and Web technology eliminated communication barriers and gave individuals immediate access to information from anywhere in the world. New businesses sprang up to exploit new business opportunities created by the global electronic marketplace. Old businesses were shaken by new competitive pressures and the need to reexamine basic operating assumptions.

Information technology is driving a revolution in business. The core drivers are microcomputer and Internet technologies, but these have spawned a wide range of new, related technologies and capabilities that must be integrated in a consistent enterprise architecture for the enterprise to remain competitive and exploit the business opportunities of this new era.

Legacy Systems

We cannot understand the impact of new technology without considering it with respect to legacy systems. Systems have been developed to solve specific high-priority problems or achieve specific productivity improvements. The evolution of technology also has contributed to the fragmentation of systems. When new systems are developed, the latest technology is employed. As time goes on, the old systems become more out of date and difficult to operate or integrate because of both changes in the technology and changes in business operations that are reflected in the newer system. In large corporations, local units develop many of the business solutions. The isolation of similar organizations, difficulties reaching consensus, and competition between business units discourage the sharing of ideas and the development of common solutions. The solutions that are developed typically are inconsistent from a business standpoint as well as technically. Business data are captured in different forms, given different names, and computed in different ways. Sources and uses of the data will vary. When corporate management seeks information on the associated business operations, the data available from different units will not support summaries and comparisons. Top managers of major corporations are frustrated by their information systems. They have difficulty getting information about how the business is running, as well as difficulty getting information to analyze the causes of major problems in order to develop solutions. When needs arise to change the business, the information systems are a major barrier to the implementation of change. In addition, the Internet and electronic commerce are changing the way business is conducted and have increased the need to improve the operation of the enterprise as a whole.

The Internet has created a global marketplace. Even small businesses must prepare for global competition. Large enterprises must be able to integrate globally distributed operating units. There is new demand for access to information from across the enterprise to respond more quickly to problems and opportunities. Customer expectations for Internet-speed responsiveness are driving the need for more responsive internal systems. Batch systems are being replaced by transactional systems, where inputs are processed as they occur.

Businesses need to exploit commercial-off-the-shelf (COTS) applications where custom solutions no longer provide a competitive advantage. COTS applications can reduce maintenance costs and put the burden of incorporating technological advances on the COTS vendor. At the same time, business processes must be streamlined and improved continuously. It must be possible to incorporate new technology for competitive advantage without undertaking long, expensive, and risky projects. Change must be rapid and incremental, supported by a flexible integration framework.

Data Warehousing

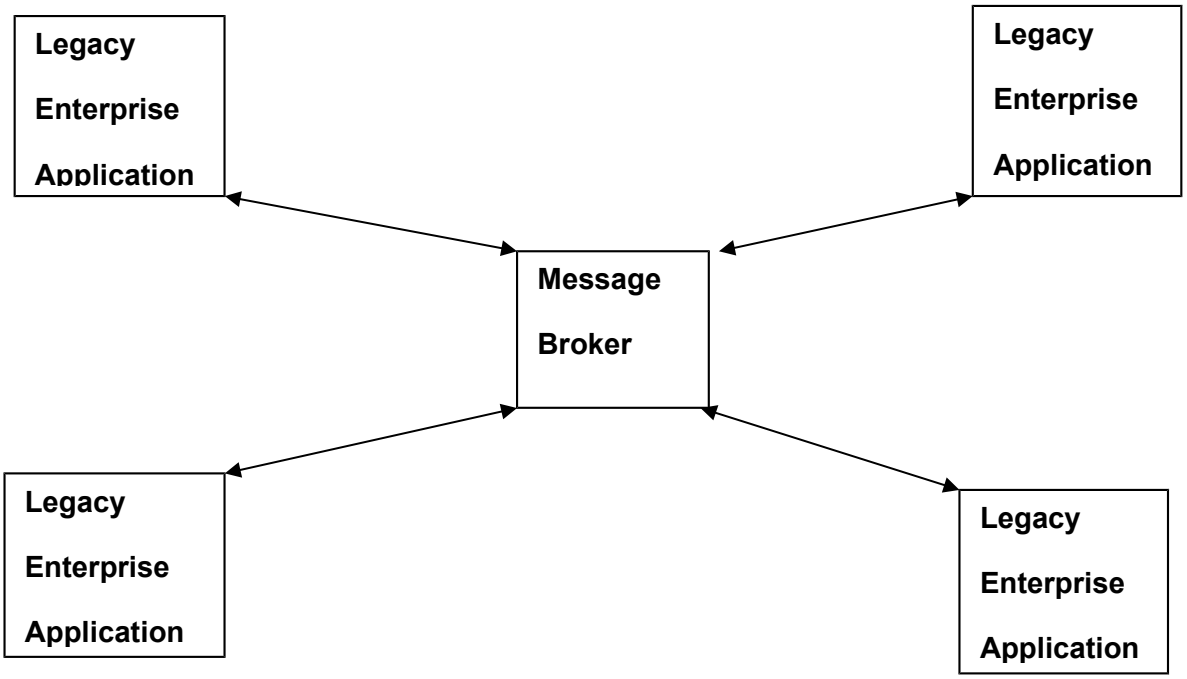
Data warehousing emerged some years ago to address needs for decision support. Mainstream business systems contain the day-to-day operating information, but it is often difficult to access such information in a form that is appropriate for analysis of business problems. Where the scope of problems spans multiple business functions, access to consistent information is more difficult. In addition, most systems for current operations would not have the tools available to obtain such information on an ad hoc basis. Online analytical processing (OLAP) tools provide much more sophisticated analyses of relationships and trends. Data warehouses typically establish an operational data store (ODS) to capture and integrate operational data where mainstream systems do not support operational queries and reporting. The need for data warehouses and data marts will continue. Mainstream systems will not retain historical data for years, nor support specialized analytical tools. On the other hand, operational data stores duplicate the data found in mainstream systems and lag in timeliness. As mainstream systems are redeveloped, it is reasonable to expect that they will meet the need for operational decision support, and the ODS will be phased out.

Enterprise Application Integration

Enterprise application integration (EAI) also involves the capture and transformation of data, but for a different purpose. EAI was driven by the need to integrate COTS applications. In the mid-1990s, a number of commercial applications emerged to support specific business functions, particularly manufacturing, personnel, and accounting functions. The installation of these products required integration with related legacy systems. EAI is the practice of linking COTS and legacy systems and a number of products have emerged to support this practice.

The basic approach is the store-and-forward transfer of data between systems. Although this may be accomplished with batch files, the need for more timely data transfers led to the use of message-queue technology typified by IBM's MQ Series. In its basic form, a sender places a message (a record) in a queue for transmission, a queue manager forwards the message to a destination queue, and the destination queue holds the message until the recipient is ready to process it. This provides a loose coupling connection because the recipient need not be ready to process the message when the sender originates it. Further independence of the sender and recipient is achieved by providing a transformation facility that converts the sender's data format to the recipient's format requirement.

Where data are exchanged between many sources and many destinations, it becomes inefficient for every source to have a connection to every destination. This is addressed by a message-broker facility that receives messages from many sources and selectively redirects them to many destinations (see Figure). This improves the flexibility of communications and reduces the overall number of links.



Electronic Commerce

The Internet and the World Wide Web have opened the door for enterprises to communicate directly with their end customers, exchange data in a more timely fashion with business partners, and establish new business relationships more quickly. The Internet, as depicted in Figure is the new marketplace, and it is global. Buyer-seller relationships can be established where they were not even considered in the past. New relationships can be established quickly, and the cost of establishing new relationships is going down.

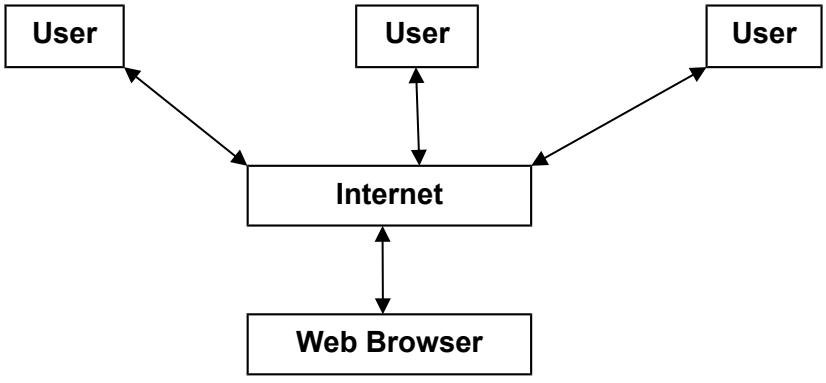
Enterprises want to make information available to customers to promote products and services. They want to make it faster and easier for customers to obtain those products and services. Finally, they want to establish a continuing relationship with customers to promote sales and customer loyalty. This business-to-customer (B2C) electronic commerce requires the exchange of data between customers and enterprise systems in new more effective and personalized ways.

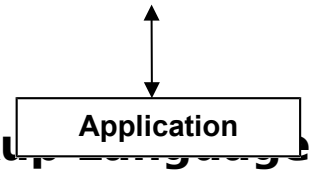
The Internet also has opened the door to closer communication between business partners-business-to-business (B2B) electronic commerce. Business operations can be streamlined by replacing Electronic Data Interchange (EDI) batch-file transfers with the exchange of information at the level of individual business transactions as they occur. Through the Internet, an enterprise can integrate with its suppliers to streamline the purchase, shipment, and payment for component products. It can integrate with its distribution channels to streamline the sales and distribution operations. Streamlining these processes can contribute to reductions in inventories and the capability to respond more quickly to customer orders and changes in market demand.

Web-Enabled Applications

Early efforts to provide Web access to applications were driven by the need to provide access to customers. However, Web browsers provide a universally available form of interface to any user with a workstation or personal computer. This means that a system that is Web-enabled can be accessed by just about anybody without any special preparation of the client system. This is a valuable capability within the enterprise as well as outside.

Making applications Web-enabled, as depicted in Figure resolves a significant portion of the difficulties involved in accessing information across the enterprise. Although the data may not be in a compatible form, at least an authorized user can access them. In addition, users of a system are not confined to accessing data from particular devices located in a particular office, but they can potentially access many systems from anywhere in the world over the Internet.



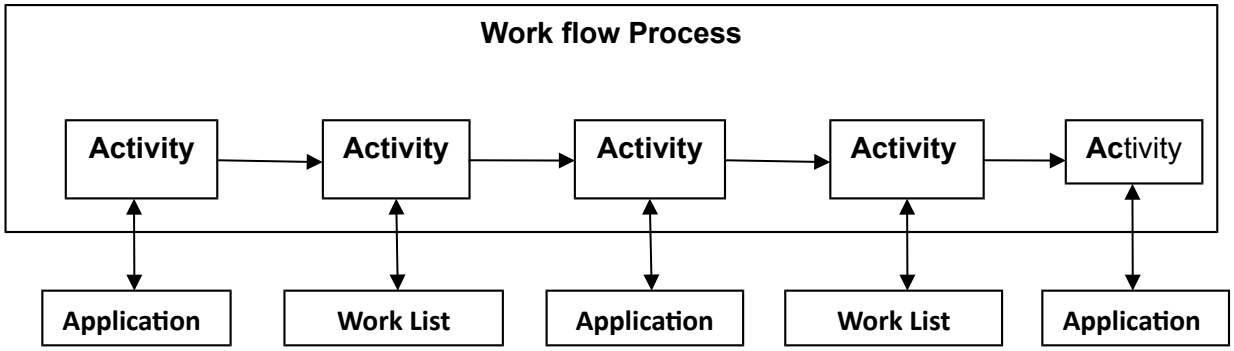


Extensible Markup Language (XML)

The Extensible Markup Language (XML) is a language for free-form expression of data structures in a universal character set. It is a a refinement of the basic syntax of HyperText Markup Language (HTML), the language of data exchange for Web browsers. It is a specification established by the World Wide Web Consortium (W3C). For a number of reasons, it has become the preferred medium for the exchange of data both between enterprises and between systems within an enterprise.

Workflow Management

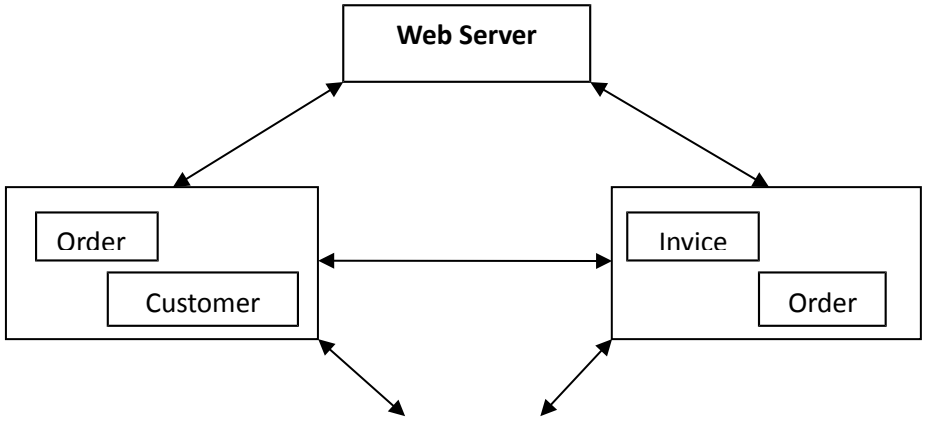
Workflow management systems (sometimes called business process management systems) have been around for many years. For the most part, they have been applied in small organizations for managing the flow of documents between activities performed by humans. They are now gaining attention for the automation of business processes on a larger scale and for coordinating the activities of humans and computers, as depicted in Figure.



Distributed Objects

Distributed objects technology supports the development of systems with distributed components that interact as objects exchanging messages over a network. The objects may be shared services or objects of a business application.

Distributed objects systems often take the form depicted in Figure 1.6. The Web server manages the user interface. The objects on application servers (Order, Invoice, and so on) implement application functionality. A database server manages the persistent state of the objects.



Components

Object technology for many years has held the promise of enabling the development of reusable components. Within individual applications or development teams, considerable reuse has been achieved. However, the vision of constructing applications from purchased application components has not been realized. There are a number of reasons; for example, sharable components require a standard environment in which to operate, they require consistent protocols by which they interoperate with other components, and their behavior either must meet specific business needs or must be adapted easily.

In addition, very large components in the form of enterprise applications have been developed and marketed successfully. Nevertheless, the development of finer-grained components for the assembly of applications has not yet been successful.

Java

The Java language has had a major impact on the industry. Its primary strength is portability achieved through execution in the Java Virtual Machine (JVM). Sun Microsystems has complemented the JVM with the Java Development Kit (JDK), which provides a number of user-interface and computing environment objects to support Java applications and encapsulate platform differences. Applications can be developed to execute without modification on multiple platforms. Java applets can be downloaded to Web browsers to provide active content to Web pages. Web browsers provide a JVM that restricts the execution of Java to prevent corruption of the client computer by ill behaved applets.

In addition to portability, Java has some other important features. The JVM provides memory management so that developers do not need to explicitly destroy no-longer-used objects. It provides a degree of reflective capability, enabling a program to modify itself at run time. New objects can be introduced to an active environment so that a program can by extended dynamically.

Unified Modeling Language

Unified Modeling Language (UML) is a specification language adopted by the OMG. It was based on a specification developed by Rational Software, but experts of the OMG specification represent the combined efforts of 21 tool vendors and industry experts. It has gained wide acceptance in the industry, and efforts continue to expand its scope.

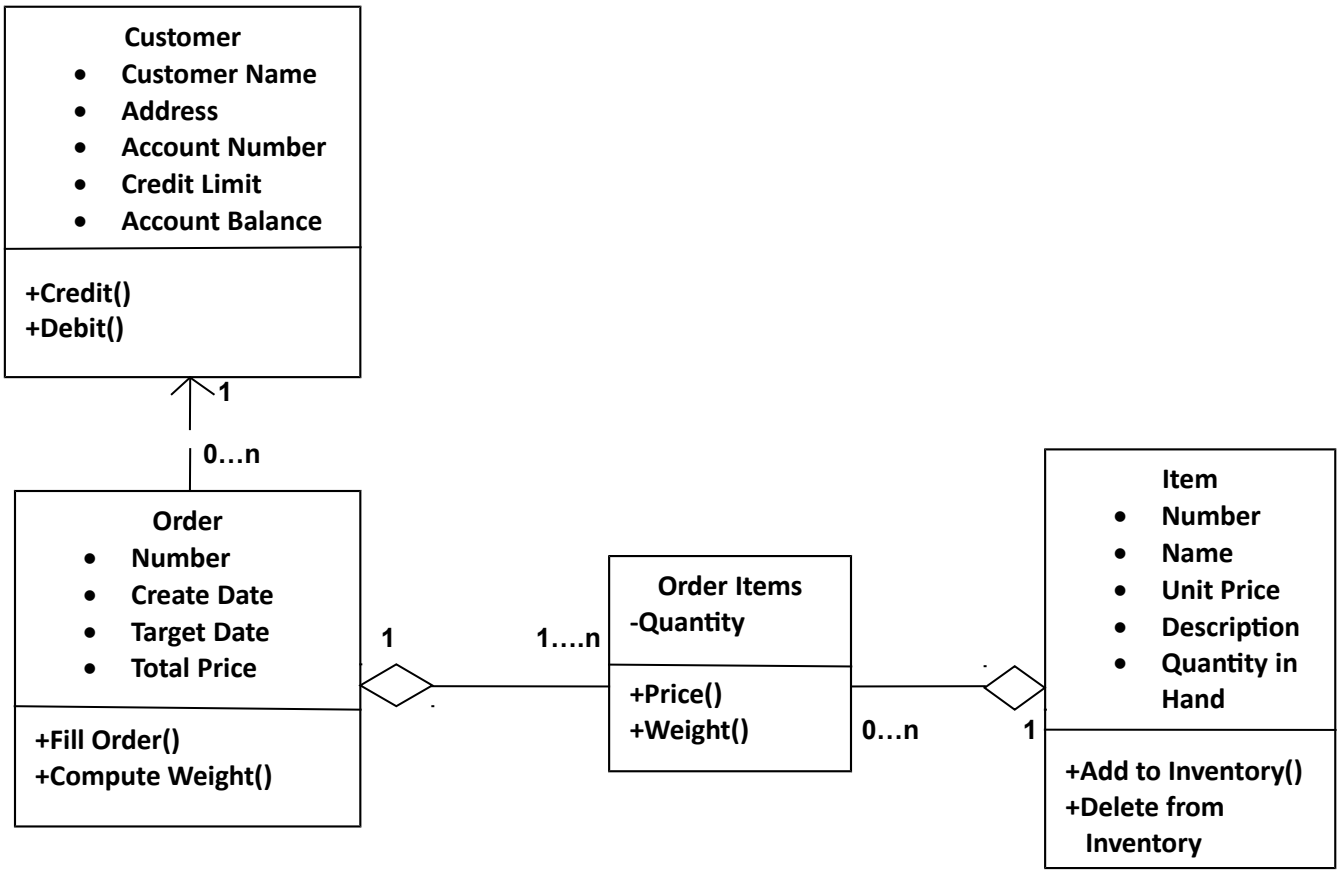
At this writing, UML has been extended to define a Common Warehouse Meta model (CWM) for specifications related to data warehousing systems. Specifications are under development to address specific requirements for modeling Enterprise Distributed Object Computing (EDOC), EAI, and action semantics, that is, application logic. Specifications are anticipated for workflow process definition.

UML is complemented by XML Model Interchange (XMI) specifications that provide for the exchange of UML models in XML syntax. This enables models to be stored in shared repositories and imported by different tools.

Public Key Infrastructure (PKI)

Security has become an increasingly critical element of enterprise systems. First of all, enterprise systems are becoming increasingly accessible. Second, systems are interacting with customers and business partners who may be personally unknown to the enterprise and who may be accessing the systems from insecure environments. Traditionally, users are given an identifier and password for each system they must access. A typical user is burdened with many identifiers and passwords. Schemes that provide a single sign-on with passwords increase the risk by

storing passwords in more places or by using one identifier and password for access to everything. A more effective security approach is provided by a public key infrastructure (PKI).



Public key technology involves the use of two complementary keys: one public and the other private. The private key (the password) is never shared, so only the owner can expose it. If a message is encrypted with the private key, it can only be decrypted with the associated public key, and if a message is encrypted with the public key, it can only be decrypted with the associated private key. One use of these key pairs is with digital certificates for user identification. Digital certificates are issued by a trusted certification authority and are encrypted with the authority's private key. The authority's public key is generally known, so a certificate is determined to be valid if it can be decrypted with the certification authority's public key. The systems and services for issuing and using certificates are called the PKI. The PKI provides a mechanism by which users can obtain certified identification from a trusted authority for use on the Internet, and they can then be identified to systems without previously being identified explicitly to each system.

Digital Signatures

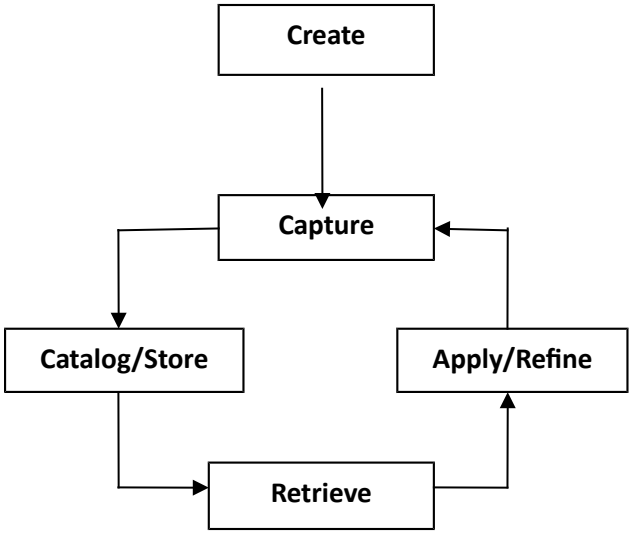
A digital signature, when attached to an electronic document, functions essentially the same as a handwritten signature on a paper document. Digital signatures employ public key technology. The signature authenticates the document by validating the signer's identity, and it prevents the signer from repudiating the document because only the signer's public key can decrypt the signature.

Wireless Interface Devices

Cellular phones have become commonplace. Some cellular phone services already provide Internet access. Personal digital assistants also can have wireless Internet access. This enables a user to have Internet access from virtually anywhere. It enables new forms of Internet purchases, and it frees employees to conduct business anywhere and at any time. It also enables communication in countries where the communications infrastructure is of poor quality or limited in availability.

Knowledge Management

Knowledge management involves the capture, cataloging, retrieval, and application of enterprise knowledge, as depicted in Figure 1.9. In product design in particular, lessons are learned over the years that contribute to the quality of the product, including its ease of use, maintainability, reliability, efficiency, and so on. When employees leave or transfer to other activities, this knowledge may be lost. In addition, the circumstances under which such knowledge is needed may not be recognized.



Agent Technology

A software agent is an autonomous program that senses and reacts to its environment according to its own internal rules of operation. Its environment m a y include information about the actions of other agents. Unlike a conventional application, the agent itself will determine if, when, and how it will perform its function. A simple agent might be employed to monitor some activity and raise an alarm when indicators suggest that the process is out of control. More sophisticated agents may be employed to direct the flow of work through the cells in an automated manufacturing facility. Such agents are effective when desired results can be achieved without overall control of the activities.

Interactive Voice

Few applications make use of voice input or output. This is because most systems are accessed with devices that have keyboards and displays. Voice input and output require additional functionality and increase the risk of errors. The widespread use of hand-held devices, particularly cell phones, is likely to change this. Users do not want to carry around keyboards and large displays. Voice input and output also enable hands- and eyes-free operation in activities and environments that are not conducive to conventional terminal interaction.

Model Driven Architecture

The Model Driven Architecture strategy of the OMG provides the ability to specify applications and standards as Platform Independent Models (PIM) that can be mapped to evolving technical platforms. Tools provide the ability to transform a PIM specification to executable code. Standard mappings will enable independently developed applications and components to be interoperable. The UML Profile for Enterprise Distributed Object Computing (EDOC) provides the modeling elements for the platform-independent specification of component-based, large-scale systems. The technology to implement MDA exists, the standards and tools for composing large-scale applications are under development.

Chapter 2
Setting Enterprise Integration Design Objectives

Enterprise integration is a rather sweeping concept that could mean different things to different people. The strategic objectives of most business enterprises are to reduce costs, improve quality, and respond quickly to business problems and opportunities. Enterprise integration contributes to each of these objectives in a variety of ways. In this chapter we will consider the following design objectives to put enterprise integration into perspective, guide the efforts, and link the elements of enterprise integration to business objectives.

Adaptable Systems and Processes

Today's business environment involves constant change. Changes are being driven by technology, by globalization of business activities, by rapidly changing markets, and by intense competition, as well as by reorganizations such as consolidations, divestitures, and mergers. Enterprise systems and processes must support these changes.

System changes are further complicated by the fact that the businesspeople responsible for the processes do not understand what is implemented in the business applications and the technical people who once understood the details of the design are no longer available.

The result is that changes in business processes require a long and tedious implementation that involves many people, considerable time, and usually trial-and-error solutions.

Coupling between business functions will increase the complexity and often the scope of change. Coupling may take the form of the need to share information or resources, or the need to coordinate activities. In addition to creating risks of delays or rework, coupling causes changes in one business function to have ripple effects in related business functions. Often these ripple effects are not well understood, and inefficiencies persist long into the future.

Management Information

Enterprise integration should enable managers to monitor the operation of the business, identify problems and opportunities, define desired changes and improvements, and measure results. Although the automation of business systems and processes has improved business operations, it often has left managers with inadequate information to manage the business effectively. Operating data are difficult to find, retrieve, and understand and may be inconsistent when measures involve data from multiple systems.

Five key elements are required for effective management information:

- **Data consistency**
- **Data accessibility**
- **Process consistency**
- **Exception reporting**
- **Historical data analysis**

Support for Electronic Commerce

The Internet and the World Wide Web have made electronic commerce a critical aspect of integration for many enterprises. Electronic commerce is essentially enterprise integration that extends to customers and business partners.

Electronic commerce requires compatibility with the systems and applications of customers and business partners, as well as the mechanisms for timely and reliable communication of information. It also creates new security and accountability risks. Many unknown people in unknown environments will have access to enterprise data and resources. Personnel of business partners will have authorized access, and the Web facilities are exposed to many potential intruders on the public Internet.

The enterprise must adapt both technically and organizationally while participating in electronic commerce exchanges around the clock, establishing relationships with new customers and business partners, streamlining internal processes to respond more quickly to customers and partners, and providing current status information on orders for products and services.

Integrated Security

Electronic exchanges raise new concerns about security. Web servers are connected to the Internet to be accessible by anyone. They open up the enterprise to new risks not present in the old systems with a defined set of users accessing terminals in a controlled office space. Certain people on the Internet aspire to break into enterprise Web servers to make their mark on the enterprise sometimes for the challenge and sometimes for personal gain.
Security must be an integral part of the integrated enterprise. The following key elements must be addressed:

- **Firewalls**
- **Authentication**
- **Authorization**
- **Integrity**
- **Confidentiality**
- **Non-repudiation**

The enterprise security strategy must recognize not only the risks of access from the public Internet, but also the risks from within. A disgruntled employee may not need to worry about getting past a firewall to corrupt a system; the employee is inside the enterprise. The only difference between a disgruntled employee and an outside intruder may be the extent to which the enterprise can make employees accountable for their actions. Unfortunately, accountability may come into play only after the damage is done.

Replaceable Components

During the 1990s, enterprise applications were developed as commercial products: commercial-off-the-shelf (COTS) software. These applications provided highly integrated and comprehensive functionality. The cost and risk associated with the implementation of these applications were much higher than most customers anticipated. As the industry evolved, the need to adapt these applications to the needs of individual enterprises became a critical success factor. This adaptation is not acceptable if it undermines vendor support, so adaptation necessarily became part of the application design. Much of this was accommodated through configuration options and parameters.
The enterprise architecture should build on this trend toward the loose coupling of finer-grained components. It should support the composition of systems from components that provide limited units of business functionality even though most of the components may be custom developed in the short term. Vendor-developed components should be integrated where they provide stable, appropriate, and limited-scope functionality for noncompetitive business functions. Applications should be designed with a component architecture that reflects a logical partitioning of functionality associated with business process activities. These functional units should be loosely coupled. This architecture will enhance system flexibility in the short term and position the systems for the integration of replacement components when they become available in the future.

Reliable System Operation

Enterprise integration will increase the enterprise dependence on its computer systems because people will come to rely on the automated processes, the streamlined business functions, the coordination of activities, and the accessibility of information. At the same time that integration is streamlining enterprise operations, it may be making it impossible for humans to function effectively if systems fail. Consequently, the need for system reliability increases.
Systems exposed to the Internet, as well as the systems on which they depend, must be much more reliable than in the past. There are three basic techniques to improve system reliability:

- **Minimize the risk that the system will fail.**
- **Detect malfunctions early.**
- **Limit the impact of failure.**

Economies of Scale

The enterprise integration strategy must include a consideration of economies of scale. It is not enough simply to connect all the systems together. The strategy should move the enterprise toward greater efficiency, leveraging solutions, eliminating duplicated effort, reducing complexity, and making the best use of skilled personnel. In general, a well-integrated enterprise should cost less to operate and to adapt to changing business needs.
Several important opportunities must be considered to achieve economies of scale:

- **Standards**
- **Software reuse**
- **Common infrastructure**
- **Consolidated systems operations**

Common Infrastructure

A common infrastructure is the complex of computers, networks, software, and associated services that supports the operation and interconnection of many systems. A common infrastructure provides connectivity, an operating environment, and shared resources to reduce the burden and promote synergy between systems. A common infrastructure is not necessarily monolithic or managed centrally. It simply provides consistent application-independent services and support. Such an infrastructure can yield substantial economies of scale in several ways:

- **EAI support**
- **Web services**
- **Personal computing services**
- **System management facilities**

Consolidated System Operations

The consolidation of system operations can achieve significant economies of scale. Traditionally, enterprise applications were executed in centralized data centers. The data center enabled the sharing of expensive computer equipment to achieve high utilization. Microprocessor technology provided a less expensive alternative, and systems developers were able to implement new, m ore user-friendly systems more quickly on desktop computers and local servers. The focus of much enterprise computing shifted from the data processing center to a client-server architecture using local servers and desktop computers. Unfortunately, while the costs were reduced, so were supporting services that were essential to the secure and reliable operation of systems.
Although consolidation offers economies of scale, the degree of consolidation must be balanced against the risks and diseconomies of scale. Consolidation puts all the eggs in one basket. A failure or breach of security could endanger the entire enterprise. Some of the economies of scale will be achieved at the expense of flexibility. Adherence to accepted standards and systems' software product selections may become more important than the selection of an appropriate COTS application; changes or extensions to computing services may be difficult to obtain due to the priority given to operational stability and the time required to evaluate the potential impact on the rest of the systems.

Chapter 3
Defining The Enterprise Architecture

In the preceding chapters we examined the technical landscape that sets the stage for enterprise integration, and we considered the design objectives for enterprise integration. In this chapter we will define the enterprise integration architecture. This architecture is based on current technology, existing or emerging standards, and industry trends. Using industry standards minimizes the architecture's dependence on a particular vendor and enhances its flexibility. Considering industry trends allows the architecture to adapt to new products and solutions as they become available.

General Characteristics of the Enterprise Architecture

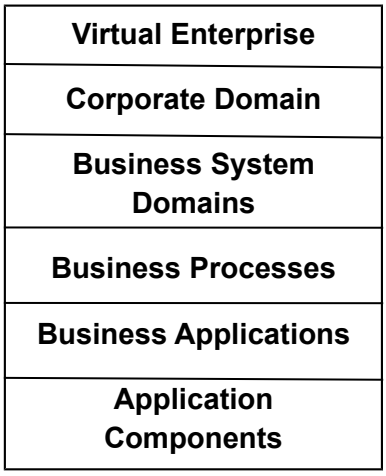
This section describes the following general characteristics of the enterprise integration architecture:

- **Distributed computing**
- **Component-based applications**
- **Event-driven processes**
- **Loose coupling of business functions**
- **Decision support information**
- **Workflow management**
- **Internet access**
- **Personalization of interfaces**

This section thus will provide a high-level view of the nature of the architecture as a foundation for more detailed discussions in subsequent sections.

Business Systems Hierarchy

The systems of an enterprise can be viewed at a number of different levels of detail . As we move up the hierarchy, the scope of business functionality increases, and the level of detail exposed decreases. These levels have implications for both the structure of the systems and the design of interfaces. We will examine each of these levels in the sections that follow, starting from the bottom of the hierarchy.



Application Components

At the bottom of the hierarchy are application components. Components are units of application software that can be combined to produce larger units of functionality. Basic components are application objects that are implemented with programming languages. These will be grouped into

object structures that provide functionality representing a service or business entity (a customer or a production schedule). In a distributed computing environment, these or larger compositions of components are accessible over the network as distributed components. Such components might be implemented as Enterprise JavaBeans (EJB). A component at run time may support a number of instances of the concept it implements; for example, a customer component may support object structures representing many customers, each accessible over the local network. Distributed components representing business entities have been called business objects.

Business Applications

The concept of business applications is changing. A business application traditionally is an integrated system that executes relatively independently, provides the functionality required by a group of users, and has an associated database and user interface. A business application in the future may manage some of the same information, but the business-specific functionality will be incorporated into business processes that are separated from the subject matter objects. The application will manage the model of that particular segment of the business but not the business processes. Multiple applications may share a database, common services, and a Web server. Consequently, an application will be defined in this context as a unit of software that provides a set of closely related business functions.

Business Processes

As noted earlier, most automated business processes currently are embedded in application code. These processes may determine the order in which activities occur, the participation of users, and the actions to be taken to resolve exceptions. When there is a need to change these business processes, users and programmers must work together to determine the changes required, and programmers then must modify program code to implement the changes. Often the business processes are not easily observed before or after the changes.

Business System Domains

Business processes, applications, and their components must be managed to meet the needs of a particular organization. The organization will coordinate and control changes to the systems it uses. It will have primary responsibility for the integrity and security of the information, including the ability to recover the system from failures.

In the target architecture, a business system domain (BSD) is the set of business processes and applications that share components and maintain consistency among the state of those components, including the capability to roll back a computational transaction that cannot complete successfully and to recover to a consistent state in the event of a system failure.

Corporate Domain

A corporation can incorporate a number of BSDs, just as the corporation may incorporate a number of organizational units. The corporation integrates these BSDs through infrastructure services and protocols that support communication and coordination between the business functions.

Coordination is accomplished primarily by the communication of business process requests and events. Processes in one BSD may invoke processes in other BSDs. Events in one BSD may cause processes to be performed or altered in other BSDs. In some cases, BSD might be outside the corporation outsourced.

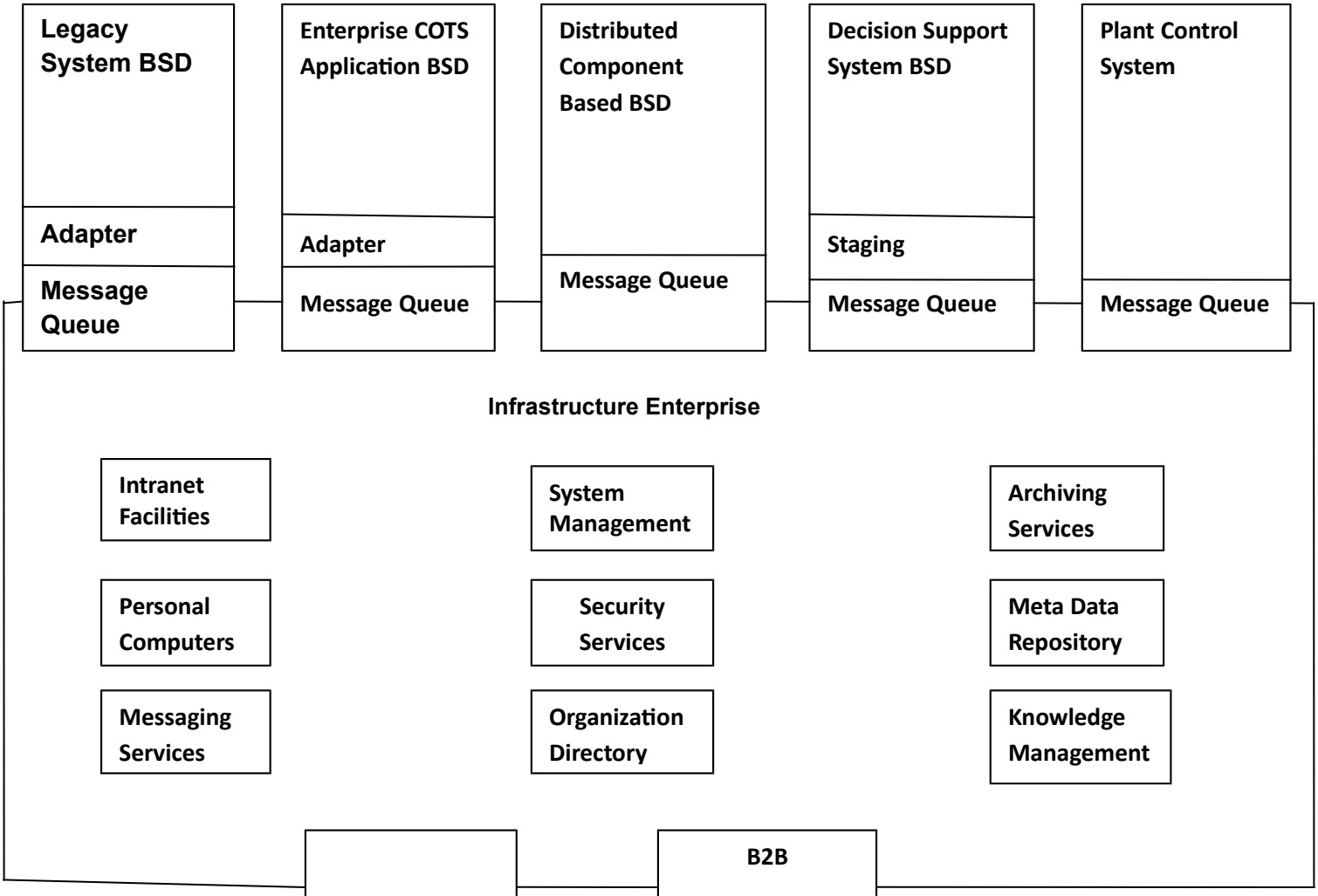
Virtual Enterprise

Up to this point in our discussion, the systems and their users are contained within the single managing entity of a corporation. The virtual enterprise level integrates customers and business partners. The difference from intra- enterprise level integration is the level of isolation between the

corporation and its customers and partners and the use of the Internet to provide communications connectivity. The corporation must be protected from adverse actions of customers, business partners, or persons posing as customers or business partners and at the same time exploit the economy and flexibility of the Internet and close working relationships. This integration includes electronic commerce and outsourcing.

Integration Infrastructure Model

The integration infrastructure is the collection of shared facilities that allow the various business systems to work together. The enterprise will incorporate many systems. These systems will come in various shapes and sizes. Some will be legacy applications, some will be purchased commercial-off-the-shelf (COTS) applications, and some will be custom applications developed to meet specific business needs. Although these systems may be developed independently, there is an opportunity to provide common facilities to support their integration and achieve economies of scale. These common facilities will provide flexibility in the overall operation of the business, reduce the cost of development and implementation of new applications, and enable the use of diverse technologies for the solution of specific business problems.

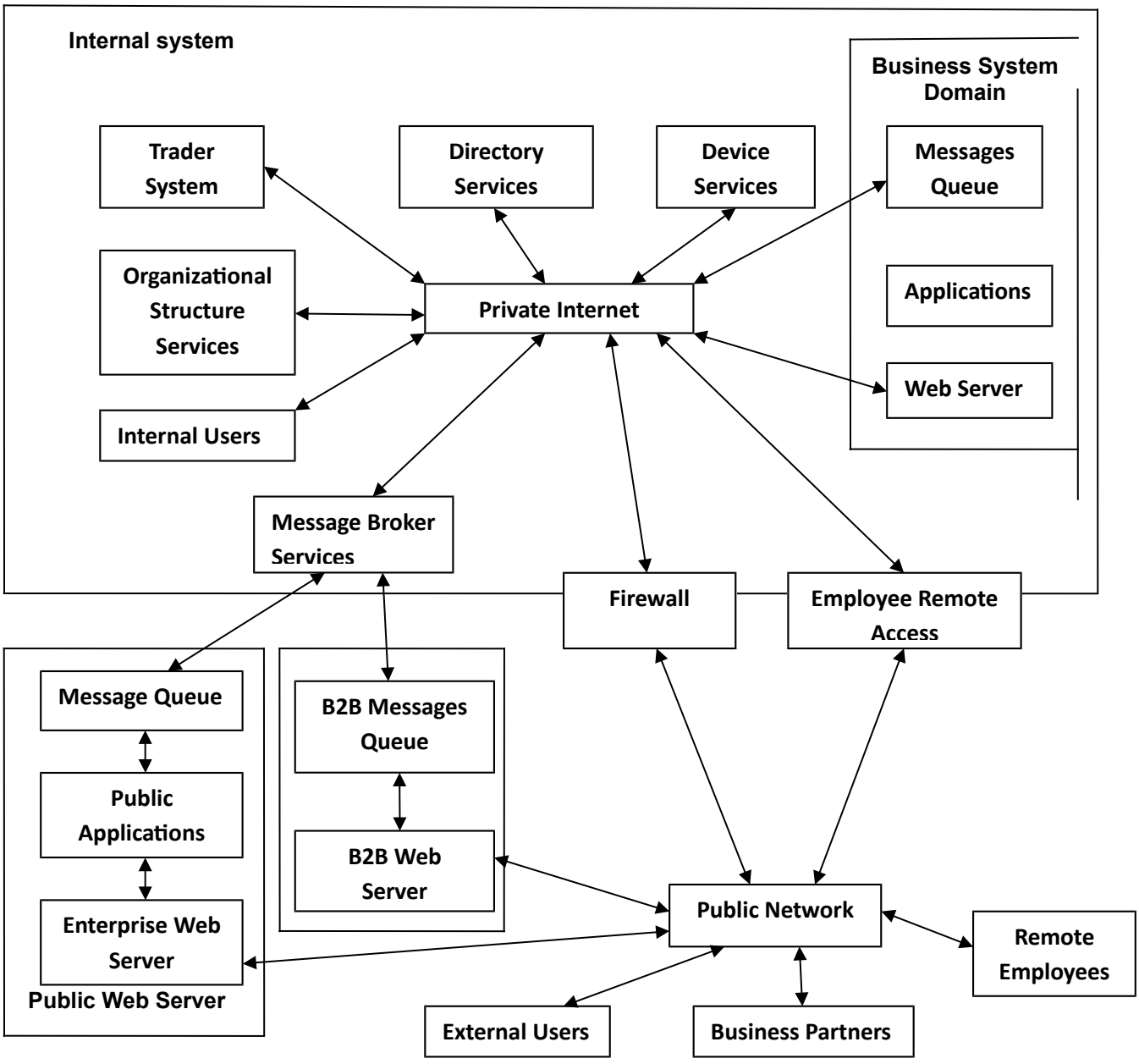


Network M

The elements of enterprise systems must be linked through networks. The networking model is illustrated. There are three primary domains:

- **Internal systems**

- Public Web applications
- The public Internet

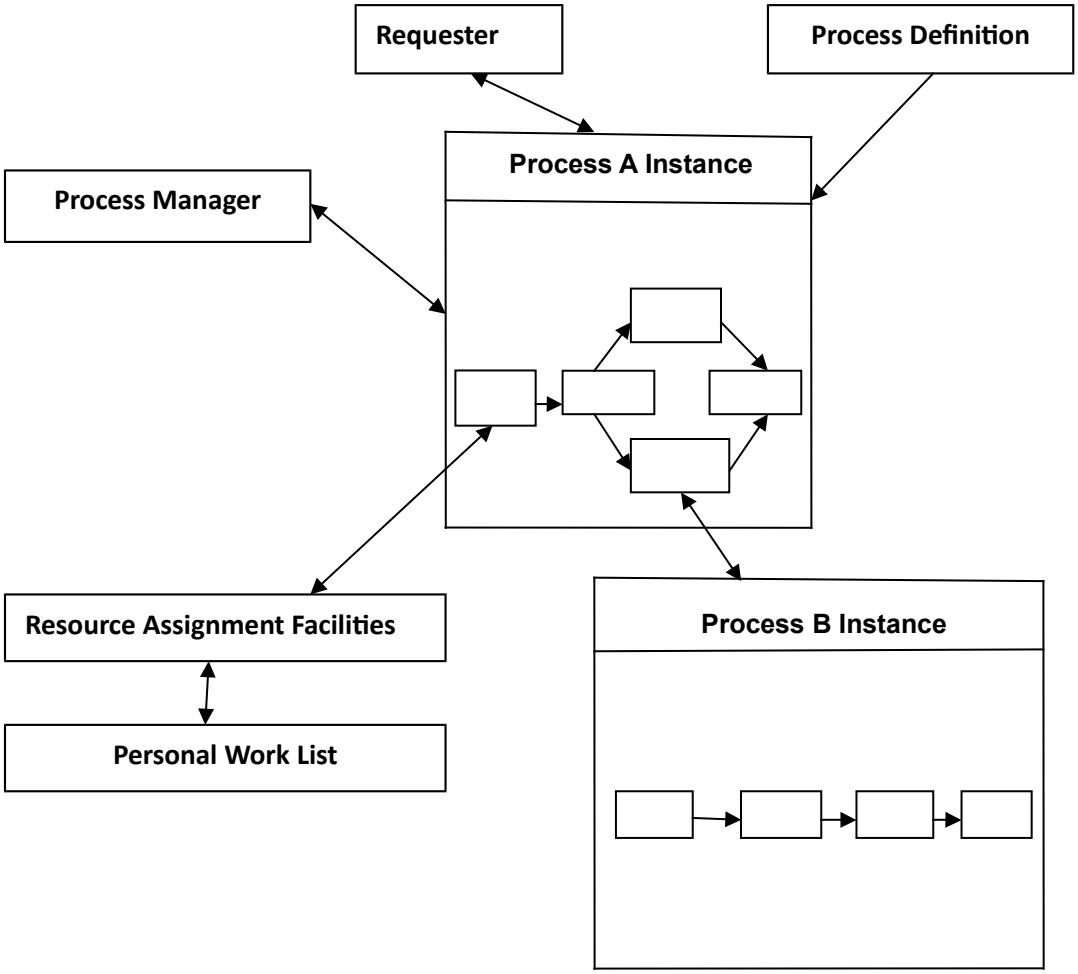


Work flow Process Model

Business processes drive the operation of the business. Some business processes perform enterprise-level actions, such as filling a customer order, and some processes automate relatively trivial operations, such as obtaining approval of a package pass. In all cases the automation of

business processes should be performed by workflow management systems. These systems have a common model. In the following subsections we will describe each of the components of this model, their relationships, and, in general terms, their implementation.

- **Process**
- **Process instance**
- **Activity**

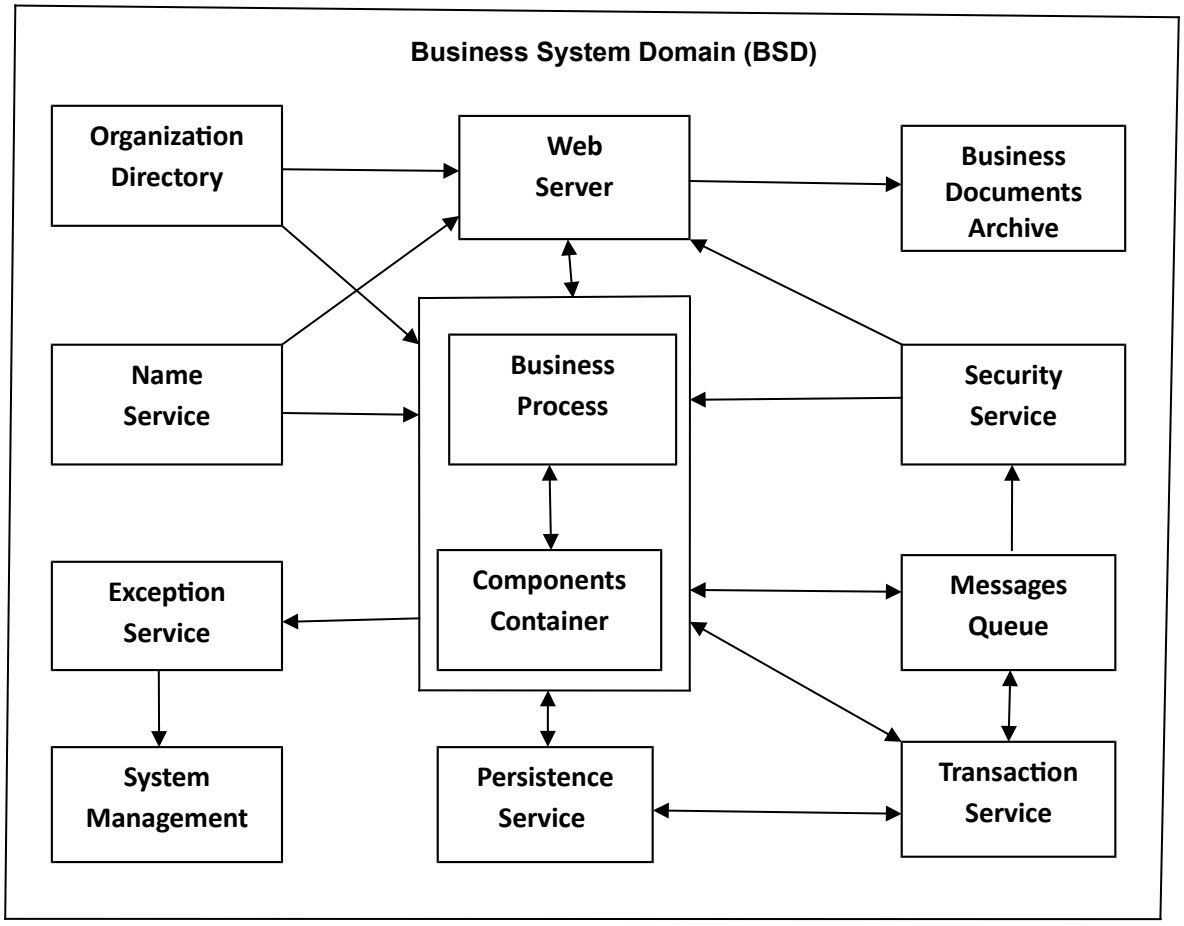


- **Requester**
- **Personal work list**
- **Resource assignment facility**
- **Process manager**
- **Process interoperability**

BSD Model

Although the enterprise architecture will accommodate a number of application models, the component-based application model will be preferred for new applications. This model is consistent with the Java 2 Enterprise Edition (J2EE) specification, which incorporates EJB

components. J2EE and EJB are considered preferred technologies in this book because of the widespread popularity of the technology and the commitment of a number of vendors to providing products compliant with these specifications. CORBA components technology provides similar capabilities and allows for implementation in other languages. J2EE can interoperate with CORBA components.



Enterprise Data Storage Model

The enterprise data storage model, depicted in Figure 3.6, is the basis for observing the current state of the enterprise as well as historical activity. It is key to providing management information to support monitoring, planning, and decision making.

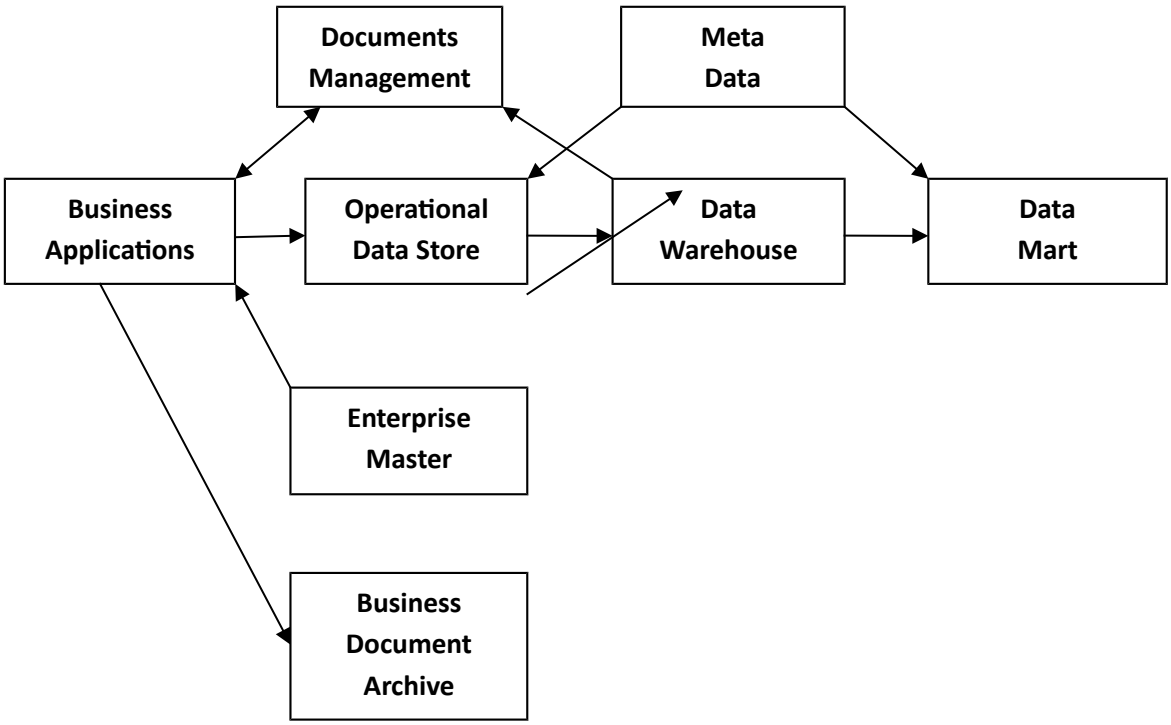
The integrity of this model and the information derived from it depend on the consistent definition of data and the consistent coordination of data changes in the different components of the data storage model. The components are discussed in the subsections that follow:

Business operation applications

Document management

- **Operational data store**
- **Enterprise master databases**
- **Business document archives**
- **Meta data**
- **Data warehouses**
- **Data marts**

Enterprise Data Storage Model

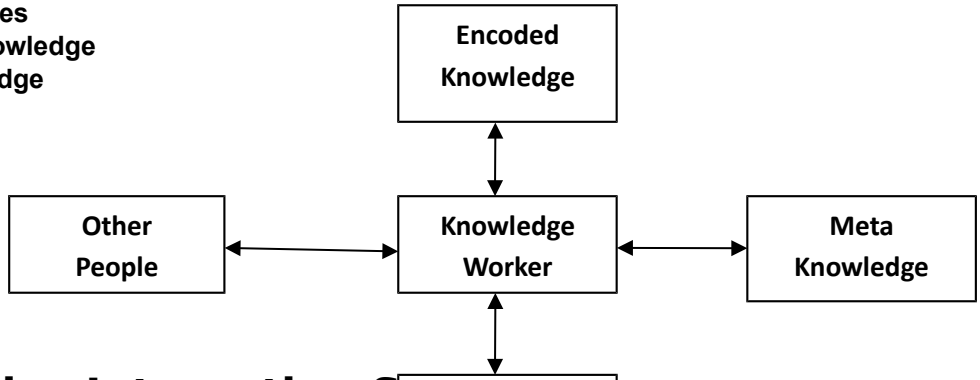


Knowledge Access Model

Knowledge is information about the business and its technologies. Typically, i t is not stored in operational databases or data warehouses, but often it is stored in less structured ways or retained in the memories of knowledge workers. Knowledge management involves capturing and preserving enterprise knowledge so that it is accessible and is not lost when memories fade or people leave the enterprise.

Figure depicts the forms of knowledge to be accessed by the knowledge worker:

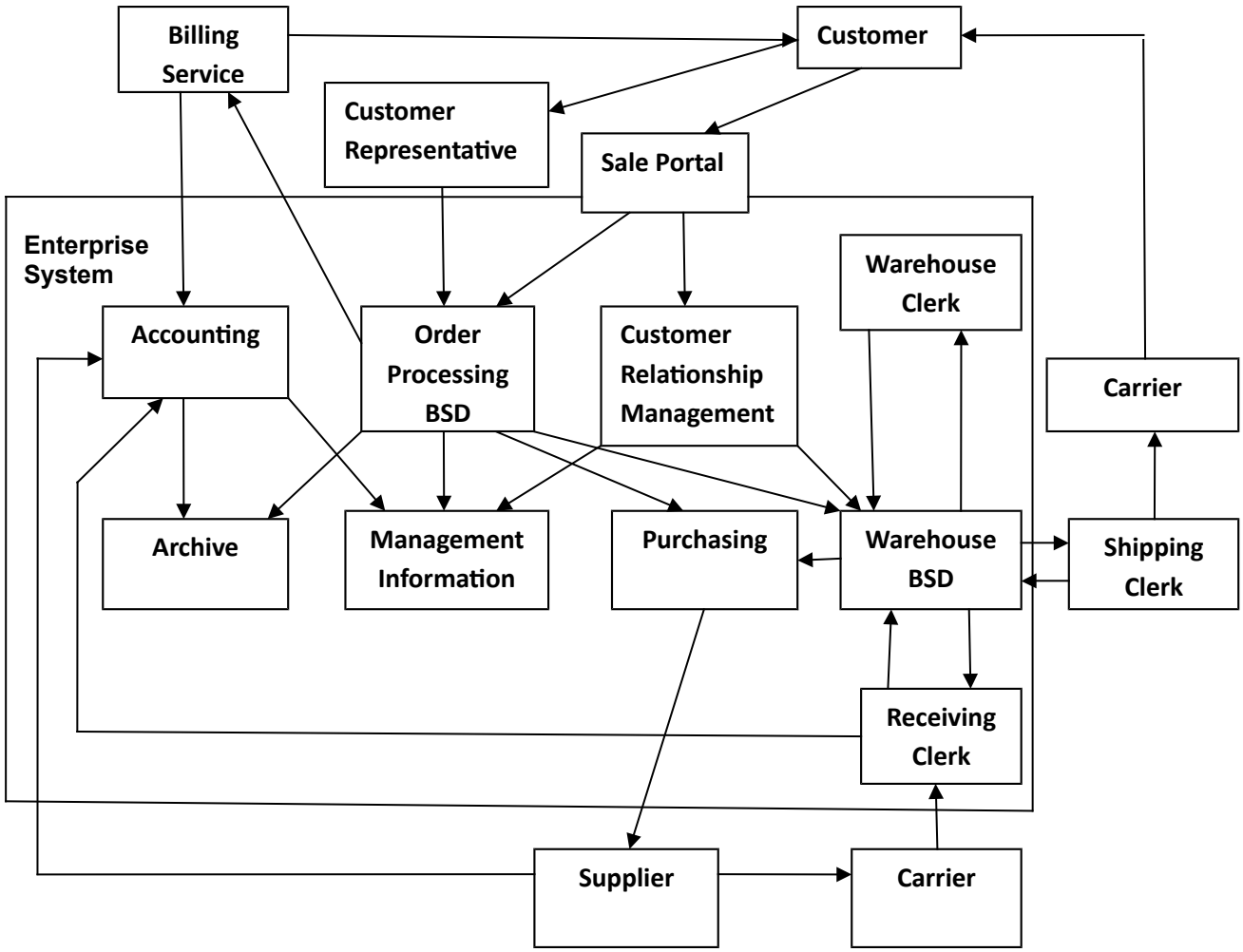
- **Other people**
- **Web resources**
- **Encoded knowledge**
- **Meta knowledge**



An Enterprise Integration S

Up to this point, the discussion of enterprise integr bstract. In order to link these concepts to the real world and provide a basis for t issues throughout this book, we have created the hypothetical business scenario depicted in Figure. We will discuss the

operation of this example enterprise and then make some observations about the integration aspects.



Hypothetical Operation

Figure as above depicts an enterprise engaged in retail sales. A customer accesses the enterprise sales portal and selects a product, such as a modem, for purchase. The sales portal is a publicly accessible application that has information on products and prices and accepts customer orders. The portal is not linked directly to other enterprise applications because of the risk that intruders might compromise the security of enterprise systems through the portal.

The customer's completed order is communicated in store-and-forward mode to the order-processing BSD. This BSD is internal to the enterprise. The communication is controlled to ensure that only customer orders, changes, and status requests are accepted from the sales portal. The order processing BSD validates the order and sends a request to the warehouse BSD to ship it.

Observations

Implicit in this example are many requirements for enterprise integration as well as potential requirements for an integration infrastructure.

- Messages between BSDs

- **Events and requests**
- **Reliable communications**
- **Coordinated activity**
- **Potential outsourcing**
- **Supply-chain integration**
- **Customer relationship management**
- **Management information**
- **Legal records**

Chapter 4
Establishing the Enterprise Infrastructure

Integration of the enterprise relies on an enterprise-wide infrastructure to support communications and shared services. Infrastructure facilities should achieve substantial

economies of scale, improve sharing and access to information, and provide the backbone management and communications that link the business system domains (BSDs) to each other and to the outside world.

This chapter will establish the general design of the enterprise integration infrastructure. This infrastructure unifies the various systems while allowing diversity in their functionality and implementation technology. It reflects a balance between the scope of consistency achieved by the enterprise infrastructure and the flexibility and diversity achieved through the autonomy of BSDs.

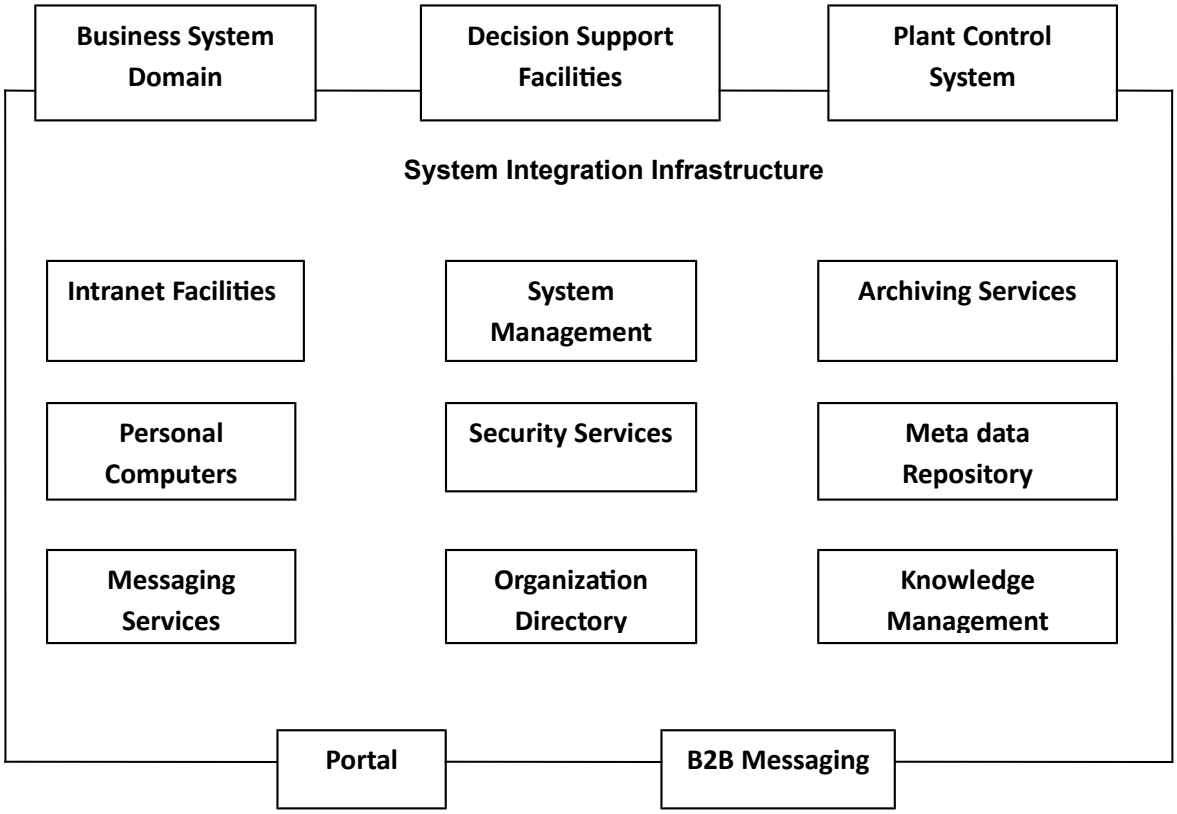
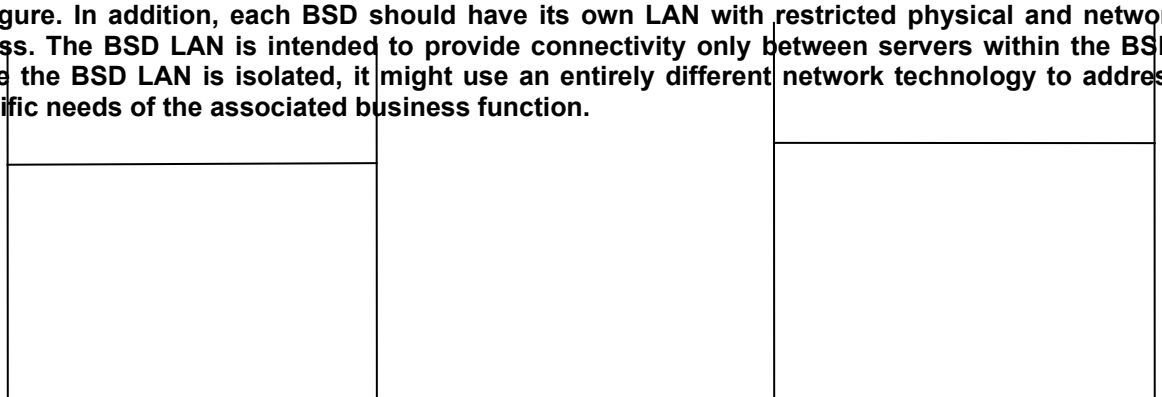


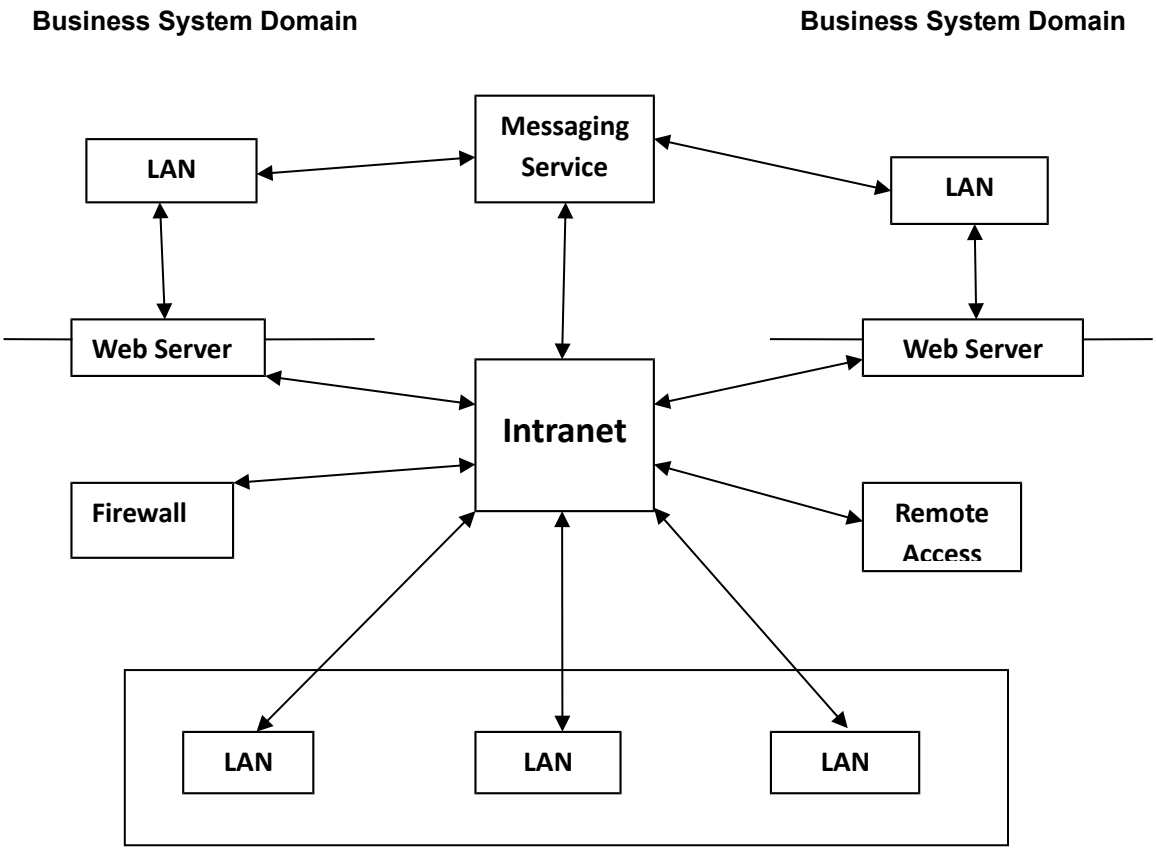
Figure as above depicts the enterprise integration infrastructure as a collection of facilities and services that link the BSDs, management information services, and portals for customers, partners, and employees.

The integration infrastructure supports a number of other systems designed for particular business functions. In Figure as above we have characterized them as BSDs, decision support facilities, and plant control systems.

Intranet Facilities

The intranet is the primary vehicle for providing internal user access to applications and for the exchange of messages and files between applications. The enterprise landscape consists of a number of local-area networks (LANs) for work groups linked by a wide-area network, as depicted in Figure. In addition, each BSD should have its own LAN with restricted physical and network access. The BSD LAN is intended to provide connectivity only between servers within the BSD. Since the BSD LAN is isolated, it might use an entirely different network technology to address specific needs of the associated business function.





Personal Computers

Personal computers are part of the infrastructure because business applications depend on them for user access, and many personal computer applications should be common across the enterprise. In addition, they represent a substantial investment and potential source of risk to the enterprise.

Personal computers execute the Web browsers that are the primary user interface facility for the enterprise architecture. In addition, personal computers handle e-mail and provide a number of personal computing applications for documents, spreadsheets, presentations, distributed conference displays, and plug-ins for the Web browser. There is certainly an opportunity for economy of scale in the purchasing and support of these common applications as well as the computers and operating systems on which they run. In addition, it is desirable that many of the applications are common throughout the enterprise to reduce incompatibilities in the sharing of information.

Messaging Services

Messaging services, sometimes called message-oriented middleware (MOM), provide communication of messages between BSDs and their applications in a store-and-forward, that is, asynchronous, mode of communication. An application posts messages to a queue for later delivery. The messages are forwarded to one or more destination queues. A receiving application removes a message from its queue when it is prepared to process the message. The use of queues allows messages to accumulate between the sender and receiver. The mechanism of updating the queues and the communication of messages provides guaranteed once-and-only-once delivery.

Message transformation is another service associated with asynchronous messaging. Transformation specifications are used to transform the structure and format of message data in

order to resolve incompatibilities between the source and destination. These transformations must be coordinated with changes to the source and destination applications.

System Management

Infrastructure system management services support operation, configuration, and problem resolution for the infrastructure. Exceptions, hardware failures, and performance problems must be recognized. Action must be taken to resolve infrastructure problems quickly and reliably. Upgrades to components and changes in workload must be anticipated, coordinated, and managed to avoid disruption of service. Network facilities must be configured and updated to accommodate new systems and users.

Security Services

Security is primarily the responsibility of each BSD. The BSD should execute i n a physically secure environment. Users must be authenticated and then authorized to access particular BSD data and functionality. For communication between BSDs, participating servers should authenticate each other, and the communications between them should be encrypted.

Organization Directory

Information about the employees and the corporate organization structure must be accessible anywhere in the corporation. While a digital certificate provides authentication of a user's identity, the organization directory defines who that person is in the organizational context. Valid, enterprise-level roles of employees should be defined in the organization directory as distinct from roles within individual organizations, which would be defined within the specific BSDs with respect to the local application data and functionality.

Archiving Service

An enterprise produces many types of documents. Some of them define agreements with customers and other enterprises. Others represent internal transactions. With the movement to electronic commerce and the elimination of paper, many documents that used to be on paper will now be digitally signed electronic documents. For performance and accessibility, it may be appropriate for individual BSDs to archive documents locally. Documents accumulated in a BSD might be batched periodically to the enterprise archiving service for long term storage. In addition, some e-mail messages will take the form of signed documents. In the long term, documents will be generated in many contexts. Legally binding documents are relevant to the enterprise as a whole. Consequently, it is appropriate for the infrastructure to provide a shared service to save legal documents in order to achieve economies of scale and ensure reliable long-term preservation.

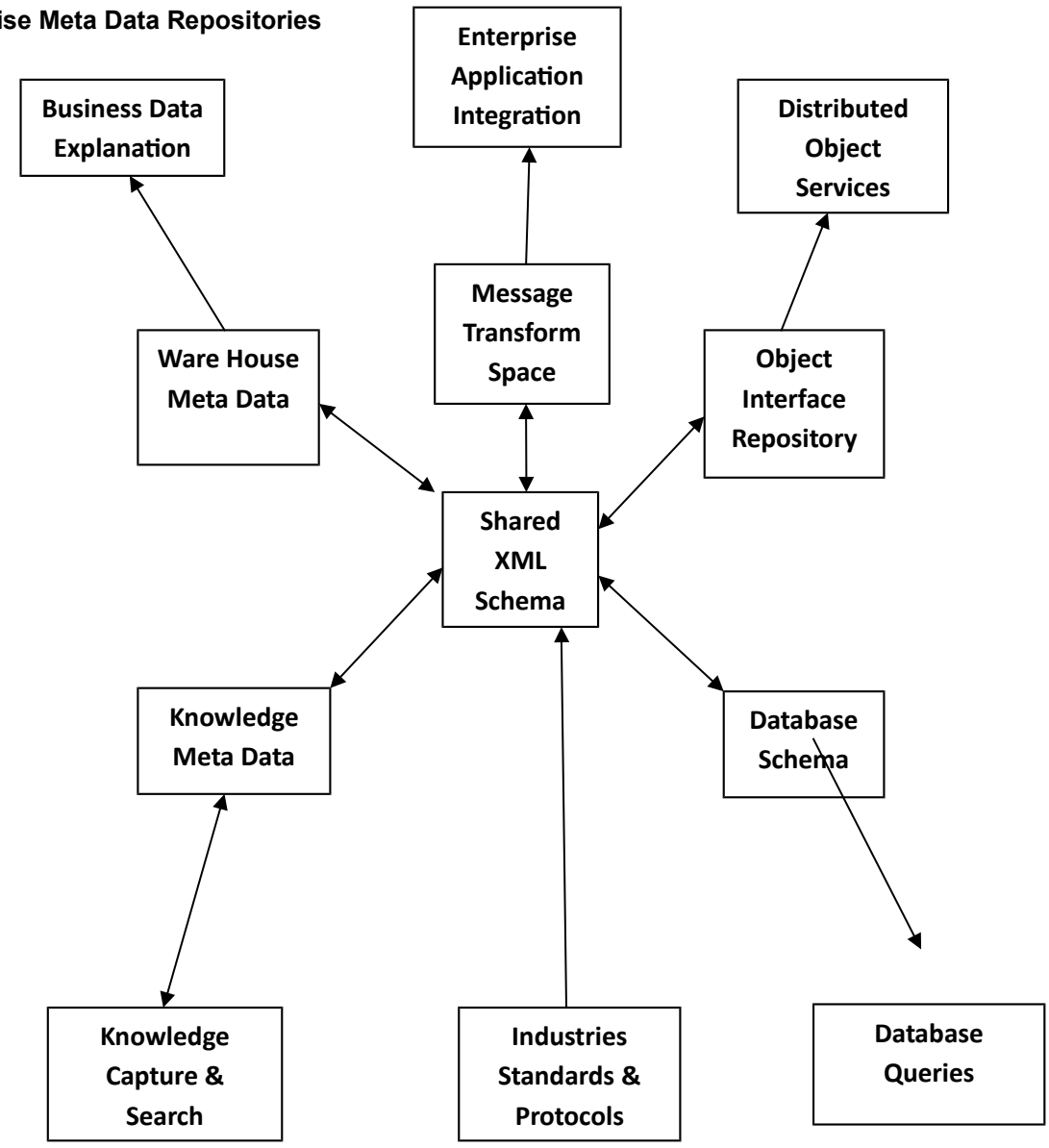
Meta Data Repositories

A meta data repository is similar to what used to be called a data dictionary. It contains data about data but also data about systems and interfaces. The traditional data dictionary was used to provide common data models for programmers. These meta data repositories provide data for runtime operations. This should be distinguished from model repositories used to support application development. A runtime repository must represent the current state of system specifications. The development repository may represent one or more future states for future systems or future versions of existing systems.

A broader concept of repositories has been developed to retain Unified Modeling Language (UML) models. The Object Management Group (OMG) has defined specifications for this repository as the meta object facility (MOF).

Six key categories of meta data are to be addressed. These likely will each have separate repositories, as depicted in Figure.

Enterprise Meta Data Repositories



Knowledge Management

We discussed knowledge management briefly in Chapter 3 and will explore it in greater depth in Chapter 12. Knowledge is information about relationships, consequences, and priorities. Knowledge, along with relevant facts, provides the basis for decisions. Knowledge will exist throughout the enterprise, sometimes in documents or rules but more often in people's heads. Knowledge is an enterprise asset and should be shared, where applicable, across the enterprise to achieve optimal performance. The key role of an infrastructure service is not to store knowledge but to help find it. The capture and classification of knowledge must be, for the most part, a responsibility of specific business functions. The infrastructure should provide for the storage and search of knowledge meta data to enable knowledge workers to find the knowledge they need.

Portals

Portals are windows on the enterprise from the public Internet. They provide points of contact and control for access to information and services provided by the enterprise. They are intended to let the outsider see only what the enterprise determines is appropriate so that risks to the enterprise

are minimized while the benefits of Internet visibility are exploited. In addition, by routing accesses through a portal, the computing workload can be managed and the user's linking patterns can be studied. User linking patterns provide a basis for improving the effectiveness of the Web service and potentially identifying customers to be targeted for particular marketing efforts.

There are four primary categories of portals:

- **Enterprise portal**
- **Employee portal**
- **Retail portal**
- **Customer service portal**

Business-to-Business (B2B) Messaging

B2B exchanges typically will be conducted between computers of business partners rather than between a human and a computer. Business partners could be suppliers, financial institutions, outsourced business functions, or commercial customers. Each participant must authenticate the other, and each must agree on the protocol for the business it wishes to conduct. Electronic business XML (ebXML), a set of standards sponsored by the United Nations, provides for the specification of protocols and documents for such exchanges along with a protocol for a reliable messaging service to ensure once and only once delivery of messages.

Chapter 5
Creating Business System Domain

The business system domain (BSD) is a central level of the business systems hierarchy. It participates in enterprise integration and electronic commerce, and it incorporates business

processes, applications, and components. The enterprise infrastructure of Chapter 4 provides a common environment for the integration and support of BSDs. In this chapter we will establish the detailed requirements of a BSD and design considerations for creating a BSD.

Characteristics of BSDs

This section will examine the following characteristics of a BSD:

- **Scope**
- **Interfaces**
- **Supporting enterprise infrastructure**
- **Alternative types of BSDs**
- **Alternative technologies**

These characteristics should apply to a variety of implementations and thus provide a basis for the relationships of BSDs to the enterprise infrastructure, other BSDs, and the outside world.

BSD Scope

BSDs are where the actual computational work of the enterprise gets done. Data are captured, computations are performed, records are updated, and results are produced. Many current enterprise applications may be characterized as BSDs. Enterprise applications typically perform primary business functions such as accounting, human resources management, or manufacturing planning, scheduling, and control. These systems typically are highly integrated and loosely coupled with other enterprise applications.

BSD Interfaces

In the preceding subsection we identified two primary interfaces through which the business functionality of a BSD is expressed: the user interface and the messaging interface. The BSD is integrated with the enterprise infrastructure through several interfaces. We will examine these interfaces in three categories:

- **User interface**
- **Messaging interface**
- **Infrastructure interfaces**

Messaging Interface

Input and output queues support asynchronous messaging with other BSDs and external systems. An asynchronous message will be posted to an output queue to be removed later and delivered to a destination queue. Messages received by an input queue are removed and processed at a time appropriate for operations being performed by the receiving BSD. The enterprise infrastructure guarantees once-and-only-once delivery of messages. The message queues are recoverable so that messages waiting in a queue will not be lost or duplicated as a result of system failure.

- **Point-to-point messages**
- **Publish-and-subscribe messages**

Infrastructure Interfaces

The user and messaging interfaces provide the linkage for business activity. At the same time, there are interfaces to the infrastructure to support the integrated operation of the BSD. We will briefly examine the following categories:

- **Directories**
- **System management**
- **Archiving**

Supporting Enterprise Infrastructure

In Chapter 4 we developed the requirements of the enterprise infrastructure and defined a number of components. The following infrastructure components as they relate to the BSD:

- **Intranet**
- **Personal Computers**
- **Messaging**
- **System Management**
- **Security**
- **Organization**
- **Archiving**
- **Meta Data Repository**
- **Knowledge Management**
- **Portals**
- **Business to Business Messaging**

Alternative Types of BSD

In Chapter 3 we identified the following alternative forms of BSDs:

- **Distributed, component-based**
- **Legacy**
- **COTS**
- **Decision support**
- **Plant**

Alternative Technologies

Distributed computing can be implemented with several alternative technologies such as the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA), Microsoft's Component Object Model (COM+), or Sun Microsystems' J2EE and EJB. While the enterprise architecture will comprehend a variety of other technologies, the preferred technology for BSDs should support distributed objects for flexibility and scalability.

CORBA Component Model (CCM) and EJB provide containers for components that enable deployment to a variety of platforms. The J2EE specification defines an application architecture for EJB components that provides a more robust model for a BSD implementation.

The J2EE specification includes interfaces to a messaging service that complies with the JMS specification. This will provide a direct link between the application and the messaging service. The J2EE specification also includes a Web server for the user interface.

However, J2EE and EJB do not provide the complete answer. For example, t h ey do not include workflow management for automation of business processes, nor do they define the infrastructure support for enterprise integration.

BSD Components

In Chapter 3 we introduced the distributed, component-based BSD model. In this section we will examine the components of this model, including:

- **Computing environment**
- **Web server**
- **Business document archive**
- **Business processes**
- **Component containers**
- **Persistence service**
- **Organization directory**
- **Name service**
- **Exception service**
- **Security service**
- **Transaction service**

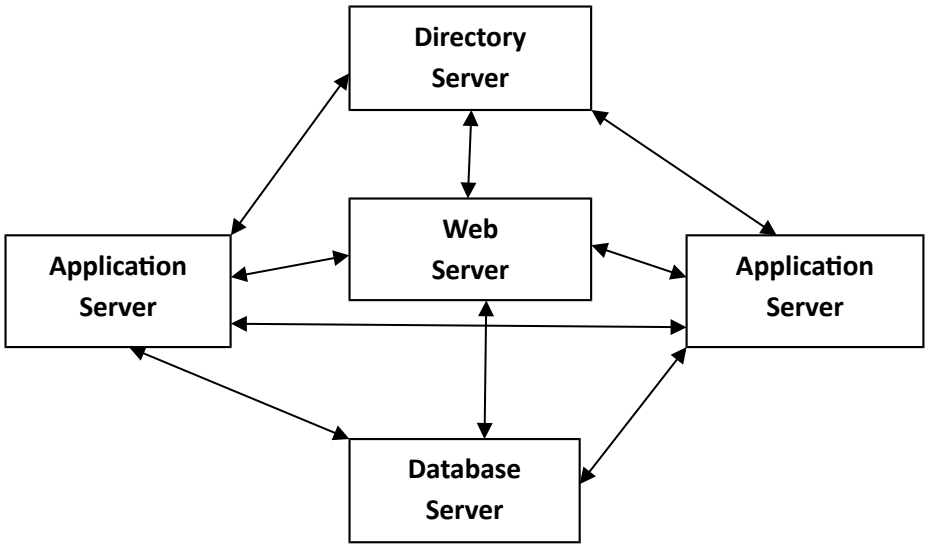
- **Message queues**

Computing Environment

The computing environment is not explicit in the diagram, but it is an essential component of a BSD. We will discuss these four basic aspects of the computing environment in order to characterize the environment in which the remaining components are integrated:

- **Servers**
- **Local networking**
- **Request broker**
- **System management**

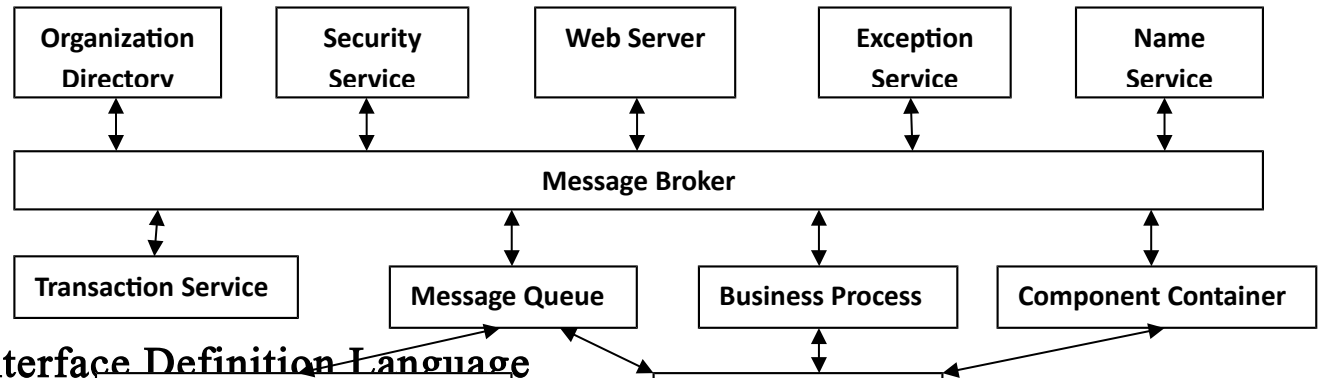
Local servers networking as follow:



Local BSD Networking

Request Broker

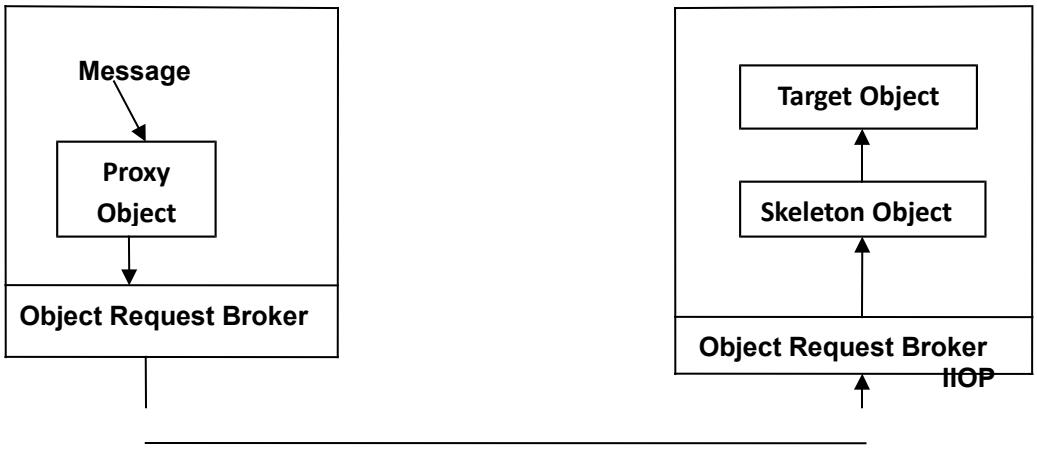
A request broker provides the mechanism for invoking methods on remote objects. This remote, synchronous messaging is the primary mode of communication between components in the BSD environment, as depicted in Figure. For J2EE- and CORBA-based systems, these messages use the Internet Inter-ORB Protocol (IIOP) defined by the OMG.



Interface Definition Language

The interfa ts are defin ition Language (IDL), a CORBA and International Standards Organization (IS rd. The IDL specifies

the object method signatures and attributes that are accessible over the network. Method signatures include parameter and exception specifications.
The IDL defines interfaces, not implementations. Different classes of objects can implement the same interface. An interface specification defines a type, which has a unique name. Types can inherit specifications from other types subtypes inherit from super types. Generally, if an object supports a subtype, it also is referenced in terms of its super type interface.



A message to remote object

System Management Facilities

System management facilities must be provided to monitor the operation of applications, manage the workload distribution across servers, manage the startup and shutdown of the BSD or specific servers, monitor network activity for potential performance problems, and resolve server or network failures.
BSD system management facilities may be dedicated to the BSD, or they may be integrated with infrastructure system management facilities. Integration may provide economies of scale, particularly if a number of BSDs are collocated. The primary consolidation would occur in the business processes and personnel for operations, configuration management, change management, problem resolution, and performance monitoring. In addition, some facilities may be shared by multiple BSDs, such as storage management and Web services, such that these facilities become part of the local, shared infrastructure.

Web Server

The Web server manages the interaction with the user. Exchanges use HTTP, the protocol of the Web. The user uses a standard Web browser. The functionality used by the Web server depends on the nature of the Web page content and the input from the user. Much of the activity may be devoted simply to providing static information that is stored in HTML pages on the Web server. Some pages will require computations or may enable the user to enter data. Pages with input or computed content will be produced and processed by servlets or a special form of servlet called a Java Server Page (JSP).
Servlets may process input and compute pages independently, but in most cases, they will obtain content from other sources or capture data for an application. A servlet may interact directly with a database to retrieve content o r update the database. This mode of operation can be used for small applications. Larger, more complex applications are implemented on application servers.

Business Processes

In conventional systems, some business processes are managed manually, and some of them are embedded in computer applications. In an integrated enterprise, all business processes will be managed by workflow management systems. This makes all processes visible for tracking and analysis and provides a greater degree of flexibility in the definition and ad hoc alteration of processes.
Consequently, the workflow management facility keeps track of the state of all processes within the scope of the BSD; it has information on all work in process for each participating employee as well as work currently not assigned. A user can sign on and select an assigned activity to begin or resume. Typically, the state of a process is committed when each activity starts or completes and potentially at intermediate points as well. When the user signs off, the current, committed state of the process is preserved and available when the user returns.

Component Containers

Component containers provide the environment for application components. Containers incorporate a number of services to minimize the complexity of the application component. The container makes the component transactional, it resolves concurrent access conflicts, it provides for event management, it provides persistence of the component state, and it implements life-cycle operations. A container manages objects of a particular type. A single container may manage many instances of a type.
EJB and CORBA component specifications include specifications for XML documents that define how components are to be assembled into composite components or complete systems. Currently, these specifications provide little assurance that the components being integrated are compatible. Specifications for a Unified Modeling Language (UML) Profile for Enterprise Distributed Object Computing (EDOC) developed by the OMG provide more robust specification of the components and the elements necessary to define the interactions between components. Component containers provide two types of application interfaces:

- **Instance interfaces**
- **Type Interface**

Persistence Service

Recoverability of a system relies on persistent storage. Various elements of a BSD must be saved in persistent storage to ensure the integrity of the system. This includes the state of workflow processes, the content of message queues, information about users and security, and system configuration information, as well as the state of objects representing the business domain. Here we are concerned primarily with preserving the state of objects in support of the business applications. The persistent state service provides a consistent interface to storage facilities that would allow applications to employ different database management systems over time or in different installations.
The state of an object must be retrieved from persistent storage when it becomes active, and its state must be updated in persistent storage when it has been changed and the associated transaction commits. The component container can handle these operations, so the application programmer does not need to implement the capabilities or determine when database accesses are required.

Object Mapping

The data structures of objects must be mapped to the physical storage structures of persistent storage. This may be quite straightforward if persistent stor age is an object-oriented database, but most often it is a relational database. This mapping can be quite complex, particularly if the mapping is to a legacy database.

Create, Delete, and Update

Database entries must be created, deleted, and updated to correspond with the creation, deletion, and update of objects in the computing environment. However, the actual database create, delete, or update should not occur until the associated transaction commits. The database should be read and objects instantiated whenever messages are sent to objects that are not currently active; this should be transparent to the application programmer. Likewise, the application programmer should not be concerned with writing updated objects to the database. The updated objects should be written to the PSS update operations when they receive a synchronize message from the transaction service commit operation. The relationships between objects and the tables they are stored in may create update sequencing requirements that must be observed for successful completion.

Queries

Queries are a fundamental aspect of application processing. Without knowing specific instances, an application would need to read or update all instances of type that comply with selection criteria.

Queries cannot be performed without reference to the system's persistent storage. First, not all objects of the specified type will be active when the query is requested; it would be an impractical and sometimes impossible burden to activate all objects of a particular type in order to evaluate each against selection criteria. Second, database management systems are designed to optimize queries, and this capability should be exploited.

Connection Management

Database management systems often define connections or paths for database operations associated with individual transactions. Each connection may allocate storage space and manage updates for one transaction at a time. The number of connections allocated also may be related to software licensing fees. The opening and closing of connections also will have associated system overhead and will affect performance.

Organization Directory

The organization directory provides information on people, their authorized enterprise-level roles, and their relationships to the organization structure. This information is the basis for business process assignments and ad hoc access to enterprise systems when the person does not have a specific role with respect to the system.

For workflow management, a person may be selected according to authority, location in the organization, relationship to others involved in the process, geographic location, or skills. The organization directory should support search/query capabilities for selecting appropriate people.

Name Service

The name service provides the mechanism by which the persistent identifier associated with a business entity or application service can be associated with the internal reference to the component that represents that entity for communication of requests through an object request broker. The name service provides the capability to classify and organize entity identifiers in a manner similar to a file directory. This supports the creation of a hierarchy of name spaces. Basic name spaces can be used for the keys of each type of identifiable entity such as purchase orders, employees, internal part numbers, vendor part numbers, and so on. A name service specification has been adopted by the OMG.

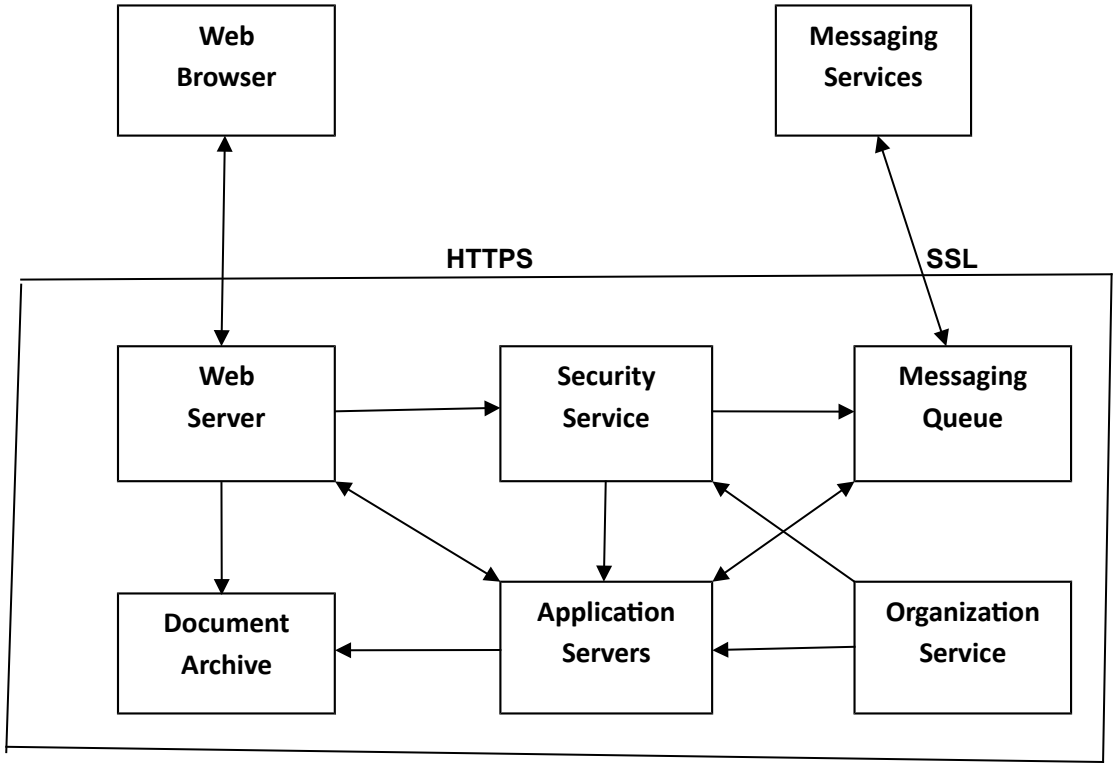
Exception Service

The exception service is invoked by an application when an exception occurs. The exception service may be implemented as a component application itself. This service provides for consistent capture and presentation of exception information. Web access should be provided to support monitoring and analysis of exceptions. Certain exceptions should initiate business

processes for follow-up to address immediate problems with the system or the state of the application activity and to prevent future occurrences.
Exceptions can occur in a variety of ways. An application may encounter a business rule violation. An assignment or computation may cause an exception due to a type error or arithmetic overflow, an error may occur in the communication of messages, or a transaction may be terminated due to a deadlock.

Security Service

The BSD security service supports the authentication and authorization activities of the BSD, as depicted in Figure. When a message queue server connects to the messaging service, digital certificates should be used to authenticate the identity of each and to engage in secure communications using SSL. When the Web server establishes a connection, the certificate of the remote user should be available for authentication. For both the message queue and the Web server, certificates must be checked against the revoked certificates list provided locally by the security service.



Transaction Service

In order to maintain a BSD in a consistent and recoverable state, operations must be performed in a transactional context. The transaction service establishes this context by providing a unique identifier for each transaction. A transaction (that is, a computational transaction as opposed to a business transaction) transitions a system from its current consistent state to a new consistent state. The changed state of the system is stored when the transaction is completed because this represents a new state to which the system can be restored if it later fails. If the system fails before a transaction completes, then the state of the system will be restored to the consistent state that existed prior to the start of the incomplete transaction, and the same or a different transaction can then proceed with system integrity preserved.

A number of factors must be considered when executing in a distributed transactional environment:

- **Serialization**
- **Deadlocks**
- **Concurrency service**
- **Lock modes**
- **Transactional context**
- **Callback**
- **Transaction control operations**
- **Phased commits**
- **Recovery**
- **Nested transactions**

Message Queues

Message queues are the primary mechanism of communication with other BSDs and business partners in electronic commerce. Message queues support asynchronous, store-and-forward messaging. They support loose coupling because a message originated by one BSD is not required to receive immediate action by the destination BSD, and one BSD can continue to operate when another fails because the only requirement of consistency between them is that no messages be lost or duplicated. Additional flexibility is provided by the enterprise infrastructure through message routing and transformation capabilities.

Application Design Issues

The performance and flexibility of a BSD depend on the allocation of functionality and exchange of data between the Web sever, the supporting applications, and the database facilities. We are not going into details of application design here, but we will discuss a number of issues to be considered in designing a distributed computing BSD.

Application Server Interface

In order to incorporate an application into Web services, the Web server must be able to access data and functionality of the application. There are two primary modes of communication: synchronous messaging and asynchronous messaging.

Synchronous

Synchronous messaging involves communication of a request and response while the requesting process waits. This is often in a transactional context.

Asynchronous

Asynchronous messaging is sometimes used to separate the Web server from the supporting application for security purposes. Asynchronous messaging can reduce security risks but also may reduce responsiveness. The Web server program will post a message to an output queue. The messaging middleware then retrieves the message from the queue and forwards it to an application input queue. The application retrieves the message from the queue to perform a requested operation and posts its response in its output queue. The response is forwarded to the Web program queue, where it is then retrieved by the Web program to create the response to the Web client.

Object-Sharing Mode

In a distributed objects environment, messages are sent to objects as active computational elements. Over time, many different users may perform operations on the same object. There are two fundamentally different modes of sharing objects:

- **Database-oriented**
- **Object-oriented**

Database Oriented

Most conventional systems rely on a database for concurrency control. Each transaction retrieves the data it needs to perform its operations, and the database manager locks the data to keep other transactions from accessing them. The data are held in memory for the duration of the transaction, and when the transaction commits, the database is updated and the locks are released.

Object-Oriented

Database accesses can be reduced and programming can be simplified if concurrency control is implemented by each object. Each object (that is, its container) must keep track of its lock status, the transaction that holds the lock, and any transactions waiting on the lock.

Network Chatter

One of the most common performance problems with distributed computing applications is the excessive use of messaging for interactions between the client and application server(s). Particularly with synchronous messaging interfaces, the client developer is inclined to request individual attributes from remote objects in order to populate user displays. This network chatter, particularly when there are many concurrent users, can have a major performance impact.

The solution to this problem lies in providing application methods designed to gather a number of elements in response to a user-interface requirement. For example, a method might be provided to request a list of attribute values returned in a name-value pair list. In other cases, methods might be implemented to return commonly used collections of attribute values. If the application designer anticipates the user-interface needs, the Web page developers will be less likely to fall into the trap of creating network chatter.

Page Content versus Format

In general, it is desirable to assign implementation of the Web page format to the Web server program and delegate preparation of the content, that is, the application data, to the application. This separation of responsibilities corresponds to the differing skill requirements of the developers and also improves flexibility.

This approach supports the design of Web pages to meet the needs of different users. For example, the same content may be used to design different Web pages for users in different countries. The layout and scope of content may differ for persons performing different job functions.

Security

Security is another common source of performance problems. It is also a potential source of application complexity and administrative overhead. The core issue is the granularity of control. If granularity of control is detailed, then authorization may be checked many times in responding to a user request. It is possible to check authority for access to every method on every object. For some applications, it also may be necessary for the application to apply rules to determine if the particular user is allowed to access the particular object. On the other hand, if authority can be checked once for the request, then the overhead and complexity will be minimized.

Sessions

HTTP is described as a stateless protocol. This means that under normal operation, each browser request is treated as an original request, even if it is a result of a previous request. The server does not remember the user actions from one request to the next. HTTP does not support a session, where a user is expected to perform a sequence of actions.

Personalization

Personalization involves tailoring the user interface and the functionality of the application to meet the needs of the individual user. The particular requirements will be specific to the application, but some common techniques may be employed.

Personalization relies on retention of settings of variable values that define the particular user's preferences. These preferences may be saved in a variety of places. For the ad hoc user, not normally known to the system, settings could be preserved in a cookie, saved by the browser.

Chapter 6
Providing The Messaging Infrastructure

In Chapter 5 the focus was on the implementation of business functions in a transactional environment. These business system domains (BSDs) represent large components in the overall operation of the business. They generally are organized around domains of responsibility, access to shared data and specialized expertise, or computations.

BSDs cannot operate in isolation. They must receive inputs either from other BSDs or external sources, and they must provide outputs to other BSDs and external recipients. The overall operation of a business relies on the appropriate communication of information to coordinate and drive the business function operations.

This chapter focuses on the messaging infrastructure that supports exchanges between BSDs as well as exchanges with business partners. While most of the interactions within a BSD are synchronous and occur within the scope of a computational transaction, interactions between BSDs, as well as business partners, should occur asynchronously. With asynchronous messaging, an originating domain is not suspended waiting for immediate action in another domain over which it has no control. The facilities that implement asynchronous messaging are sometimes called message-oriented middleware (MOM).

This chapter is organized in three parts:

- **Design objectives**
- **Application of JMS**
- **Design considerations**

Design Objectives

This section outlines a set of design objectives for the messaging infrastructure. This will provide an overview of the use of this technology in the enterprise integration architecture. The following objectives are discussed:

- **Store and forward**
- **Message broker**
- **Guaranteed delivery**
- **Message sequence**
- **Symbolic routing**
- **Request-response**
- **Event messages**
- **Message transformation**
- **Ad hoc destinations**
- **Exception resolution**
- **Standards**
- **File transfers**
- **Business-to-business (B2B) communications**
- **Security**

Store and Forward

A message must be accepted by the messaging infrastructure and held until the recipient or recipients are ready to accept the message. The sender must not be blocked once the message is accepted for delivery. This allows the sender to proceed without waiting for the recipient to be active or ready to accept the message. It is equivalent to the mode of communication used by e-mail.

Message Broker

Primitive messaging facilities provide direct communication from one sender to one receiver. This is acceptable for integration of very limited scope. Once several systems are communicating with

several other systems, there is considerable economy of scale, as well as substantial improvement of flexibility, by providing a service where messages from any system can be routed to any other system.

Guaranteed Delivery

The mechanism for the delivery of messages between senders and receivers must be able to guarantee that each message will be delivered to each recipient once and only once. The operation of the sender must be able to proceed on the assumption that the message eventually will be received and processed by the recipient. If this cannot be ensured, then the operation of the business will be unreliable, or considerably more overhead will be incurred to verify delivery and repeat the request if delivery has failed.

Message Sequence

The sequence of messages from a single source should be preserved when received by a single recipient. Often messages reflect a sequence of events such that the last message received reflects a transition to the current state. If messages are received out of sequence, the current state may be misrepresented, and the path to the current state could be misunderstood, potentially resulting in erroneous computations. While the receiving application can be designed to compensate for out-of-sequence situations, the application likely will become much more complicated.

Symbolic Routing

If messages were directed explicitly by one system to another using physical addresses, such as Internet Protocol (IP) addresses, the operation of the enterprise would be very inflexible. Changes in system configurations and networks, as well as the implementation of new systems, could require changes t o all related systems. Instead, the routing and delivery of messages must be based on symbolic recipient addresses defined in a directory.

Request-Response

Certain business or computing services will be shared by many applications. These common services may receive requests in the form of messages from many sources and provide results back to the message senders. It is important that these services need not be programmed explicitly for each potential source. Instead, each request should define the recipient of the response.

This is similar to messaging performed synchronously where the sender expects a response, except that here the message and the response are communicated asynchronously. While the sender could communicate a return address within the message, this puts an additional burden on the sending application to provide a valid return address because the requests may be fulfilled on different computers at different times, or the sending application may be executed at multiple sites.

Event Messages

Event messages communicate the occurrence of an event. An event message may come from an external source or may represent the occurrence of activity or a change of state within the system. Events may be used to initiate or alter system processes. The infrastructure must allow certain events to be monitored selectively on an ad hoc or continuing basis. Generally, the originator of these events is not specifically aware of the recipients, but only provides notice of the events when they occur.

Message Transformation

When independently developed systems are integrated, particularly when some are commercial-off-the-shelf (COTS) products, the format of messages that different systems accept or produce

may be inconsistent. Furthermore, it is not possible to define fixed formats for all messages and never change them. A message transformation service can accept a message in the sender's format and transform it to the format(s) required by each recipient. For large enterprises with diverse operations, it is essential that the infrastructure provide mechanisms for the transformation of messages so that as new systems are introduced and old systems are eliminated, the effects of changes in senders and receivers can be isolated and managed over time.

Ad Hoc Recipients

Some messages between business functions are part of the mainstream operation of the business and always must be communicated between certain business functions. On the other hand, some applications are outside the mainstream operation of the business or need information on business operations on a more ad hoc basis. It must be possible for these systems to obtain ad hoc subscriptions to the distribution of certain types of messages in order to monitor the business operation.

In some cases, the recipients will only be interested in current events that occur while the recipient is active. For example, a performance monitor might provide a display of current activity. When the recipient is terminated, it no longer exists for the receipt of messages.

Exception Resolution

The messaging infrastructure should resolve exceptions as much as possible to minimize the need for concern by the application developer while maintaining the integrity of the applications. The principal exception of concern to applications is where guaranteed delivery, once and only once, could be compromised.

Delivery failure could occur if the network or the receiving application were not available for an extended period of time. The list of messages to be delivered could become excessive and have an impact on system performance or exceed the capacity of the system to hold the messages for delivery. The messaging infrastructure must define an orderly approach to resolving this situation.

Standards

The integration infrastructure should be comprised of shared facilities that achieve economies of scale in development, operation, and support and use purchased components to deliver solutions quickly and incorporate new technology as it becomes available. The application of industry standards will help ensure flexibility in the configuration of facilities as well as in the ability t o incorporate new and different products to exploit technological improvements.

File Transfers

The general concept of the messaging infrastructure is to communicate information about business events and transactions as they occur. Typically, these messages are relatively small; large messages might be a few thousand bytes in length. Unfortunately, some systems will still operate in batch mode, and other systems that communicate events and transactions also must transfer large files of associated data, such as graphics or product design specifications. The messaging infrastructure must provide reliable transfer of files without a significant adverse impact on the performance of the rest of the system. The file transfers must have the same guarantee of delivery as the messages with which they are associated.

B2B Communications

Communications between business functions must extend to the business functions of business partners. This requires communication through firewalls, which place restrictions on the communications. At the same time, the communications must be in a nonproprietary mode so that the enterprise can communicate with a variety of external systems with few compatibility requirements.

Security

Security must be considered at two levels:

- **Messaging within the enterprise**
- **Messaging with external systems**

Within the enterprise, we will assume that the systems that participate in messaging are in environments that are appropriately secure for their business function. However, messaging often will occur between business systems over less secure communications facilities and possibly the public Internet. Communications facilities and protocols must ensure the privacy and integrity of communications between systems. In addition, since any source might send a message to any destination, an application must be assured that a message is being received from an authorized source, that the source is authorized to send to the specified destination, and that subscribers are authorized to receive the messages they are requesting.

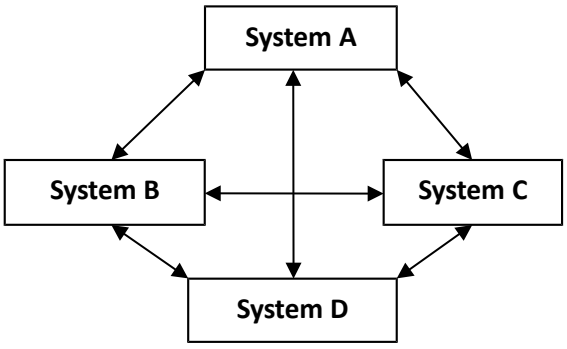
Application of JMS

This section will describe the core technology for the messaging infrastructure in terms of JMS. The goal is to achieve an understanding of the technology and how to use it, not how toil implement it. This discussion will describe how a JMS-compliant product will be used to support enterprise integration, and it will identify product requirements that resolve relevant ambiguities allowed in the JMS specification. The discussion will cover the following facilities:

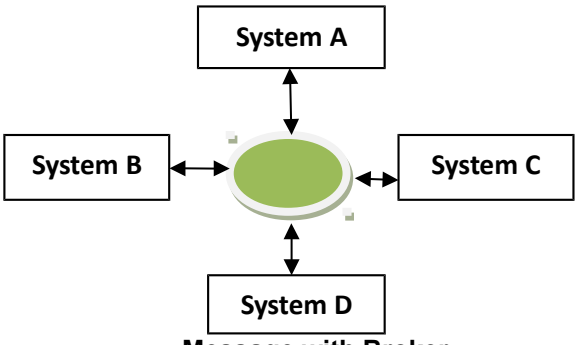
- **Message queues**
- **Basic messaging facilities**
- **Point-to-point messages**
- **Publish-and-subscribe messages**
- **Message format abstraction**
- **API object model**

Message Queues

The concept of message queues is fundamental to asynchronous messaging. A sender places a message in a queue for the service to deliver. The message is transmitted to a destination queue, where it is held until the recipient is ready to process it. This provides a pure point-to-point communication. This primitive form is used to link a specific source and a specific destination.
Recoverability and guaranteed delivery are achieved by making message queues persistent and transactional. When a message is submitted to a queue for sending, the submission is performed in a transactional context. The message is not logically accepted into the queue until the transaction is committed. At that time, both the message queue database and the database of the source application are updated. If the commit fails, both the application activity and the message queuing will be backed out.



Message without Broker



Message with Broker

Basic Messaging Facilities

JMS defines two modes of messaging: point-to-point and publish and subscribe. First we will examine the facilities that are common to both these styles, and then we will consider the styles individually. This will not be an exhaustive examination because details can be obtained from the JMS specification. Instead, we will examine the features and options that are particularly relevant to support enterprise integration.

Applications communicate with JMS through a connection. Messages maybe sent and received (in other words, placed in and removed from queues) through a connection, and concurrent processes to consume or produce messages may operate through the same connection. The connection operations of each process thread are managed by a session.

Point-to-Point Messages

The terminology point-to-point can be misleading because it suggests that an application communicates directly with another application, as in the scenario discussed earlier, without a message broker. To the contrary, point-to-point in JMS simply means that the sender of the message directs the message to a specific recipient.

JMS allows, but does not require, that a product supporting JMS use a destination to designate a destination list. In this case, the implementation message broker product would receive the message, and copies of the message would be directed to each destination on the list. This is a desirable feature but generally not essential.

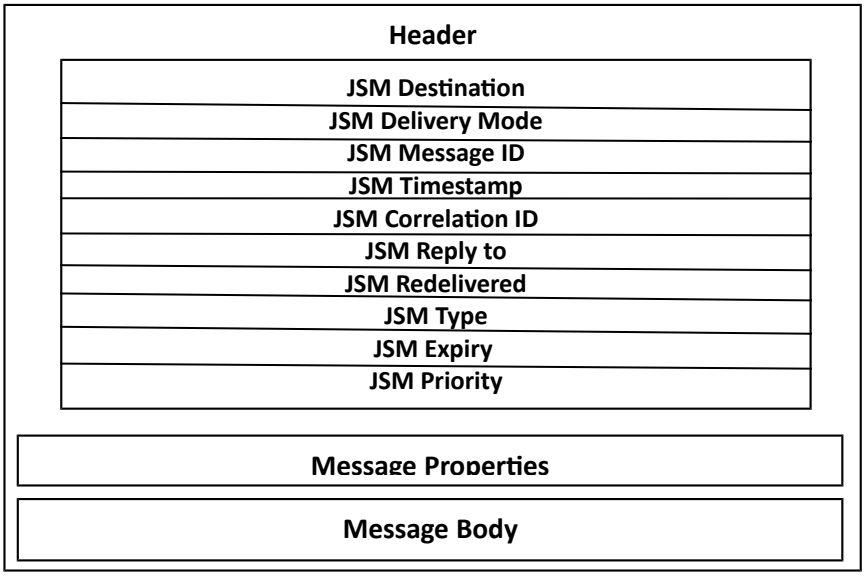
Publish-and-Subscribe Messages

In publish-and-subscribe messaging, messages are not directed to specific recipients, but are directed to topics. Interested recipients subscribe to a topic and can receive all messages directed to the topic or only those that meet specified selection criteria. Each subscriber is an independent recipient of messages, so messages consumed by one subscriber are also delivered to other subscribers. Messages that do not meet the selection criteria of a subscriber are not delivered or are held for that subscriber.

Message Format Abstraction

When different products implement JMS, they may use different message formats; a message format is not specified by JMS, so a variety of existing products can implement JMS. However, JMS does define a message format abstraction, which defines how an application views a message as well as associated data that must be communicated to the recipient. That abstraction is depicted in Figure.

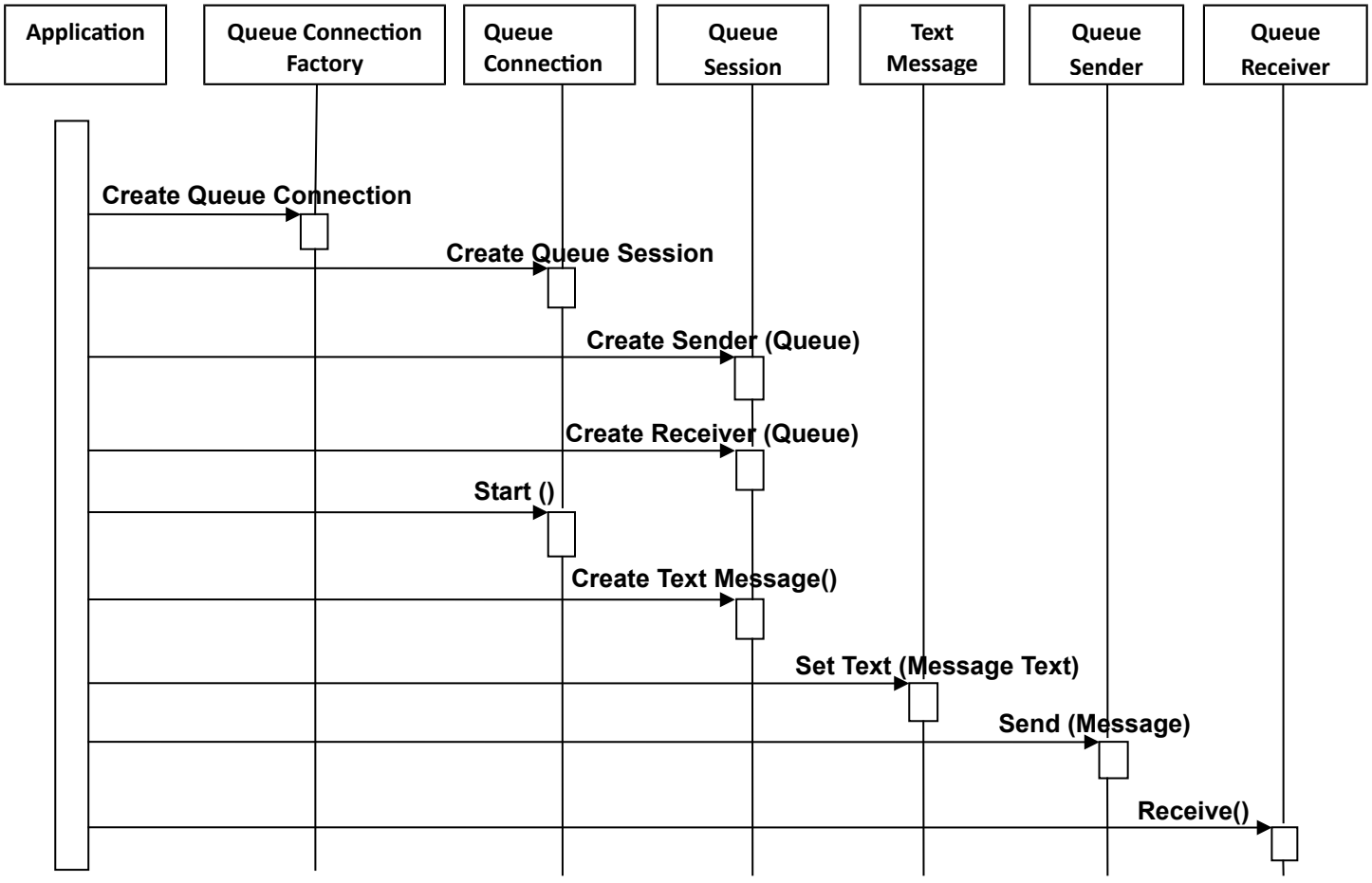
We will not go into detail on the elements of the message format because that information is available from the JMS specification. However, we will discuss some of the elements as they relate to the enterprise architecture.



API Object Model

The JMS API is expressed by a number of objects that provide the messaging service interface. The object model is defined in detail by the JMS specification and will not be repeated here. However, Figure 6.3 is included to illustrate the interactions between some of the key objects in the JMS model. The diagram depicts the sending and receiving of messages in a point-to-point style of communication. The messages depicted by the arrows in the interaction diagram are all synchronous messages executed in Java.

The application obtains a reference to a queue connection factory (from JNDI) and sends a create Queue Connection message to obtain a queue connection. It then requests a queue session from the queue connection with the create Queue Session message. It could request additional sessions if it intends to operate multiple threads. It obtains a queue reference (from JNDI) for a queue to which messages are to be sent and creates a queue sender with the create Sender message to the queue session. It obtains a queue reference for a queue from which it accepts messages (from JNDI) and creates a queue receiver with a create Receiver message to the queue session. It then sends a start message to the queue connection, enabling the sending and receiving of messages.



Design Considerations

The JMS specification defines a vendor-independent messaging service. We have discussed it as a generic capability. The following design considerations provide additional insights on issues that arise in practical applications:

- **Product interoperability**
- **Transformation service**
- **File transfers**
- **B2B messaging**
- **Security**
- **Scalability**
- **Application execution**
- **Exception handling**

Product Interoperability

There are currently no standards for interoperability between message broker products. The solution is to create a bridge between the products where messages are received in a message queue from each product and forwarded to t he output queue for the other product. Some of this can be accomplished within the functionality defined by JMS.
This works for point-to-point messages. The bridge functions in each message broker domain as a proxy for the other message broker domain. The proxy queues in one domain must function as destinations for all messages destined for destinations in the other domain. This creates an additional maintenance burden to create new proxy queues whenever a destination queue may receive messages through the bridge.

Transformation Service

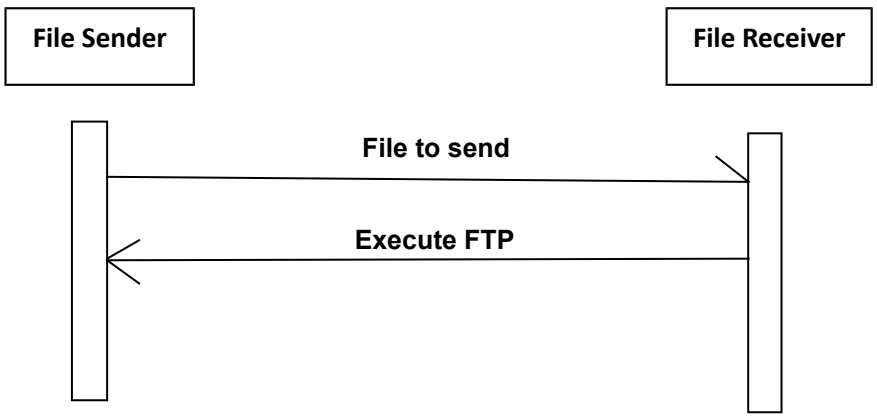
Transformation services have been developed to support the incompatibility of messages to be exchanged between systems for EAI. These products are either integrated into or used to extend messaging facilities. They enable the sender and receiver to communicate even though they may be using different message body formats. Of course, the content of the message body provided by the sender must have sufficient content with equivalent semantics that the format expected by the receiver can be created.

File Transfers

File transfers over the Internet are commonly performed using the File Transfer Protocol (FTP). FTP provides no guarantee that the file will be delivered once and only once, but using messaging to directly carry the content of large files could create serious performance problems for the messaging system. The answer is to use the two facilities together-use the messaging facility to manage the file transfer.

B2B Messaging

The infrastructure messaging service will communicate messages within the enterprise, but messages to business partners require an even more loosely coupled mode of message communication. Communication with business partners will be over the public Internet and must be assumed to be with incompatible message broker products. The industry direction is to communicate between enterprises using the Hyper Text Transport Protocol (HTTP) as the least-common-denominator protocol. HTTP can easily pass through firewalls, it is supported by all Web servers, and, like HTTPS, it supports secure communication based on digital certificates.



Security

The messaging infrastructure requires consideration of the following six security concerns:

- **Authentication of a participating server**
- **Protecting the privacy and integrity of the message**
- **Authority of a sender to send to a specified queue**
- **Authority of a publisher to send to a topic**
- **Authority of a subscriber to receive from a topic**
- **Providing non-repudiation of message content**

Scalability

Scalability must be considered at two levels:

- **The scalability of the JMS implementation product**
- **The scalability of the applications that use it**

We will not examine the scalability of messaging products here; while it may be possible to make some assessment by examination of the product specifications, for the most part, the determination of product scalability must be through testing. Here we will examine factors to be considered in the design o f applications and the features that messaging products must provide to support the scalability of applications. There are four primary concerns:

- **Multithreading**
- **Load balancing**
- **Queue overload**
- **Unsubscribed topics**

Application Execution

JMS does not provide any mechanism to initiate execution of an application to process a message. An application must be active to receive a message from a queue or a topic. It is an unnecessary use of resources and an additional burden on the system administrator to have a number of applications executing just in case there might be a message for them.

To address this need, an execution service should be implemented. The execution service executes as a daemon within each BSD to monitor an execution queue. When it receives a message, it executes the appropriate application. The particular execution environment will determine the manner in which the application is executed and the mode by which the message is relayed to it.

Exception Handling

There are two levels of exceptions that a JMS application must consider:

- **Exceptions arising in response to the invocation of a JMS function**
- **An exception communicated to a JMS exception listener**

Exceptions returned from a function call must first be addressed in the context of the application. Unless they are exceptions that are anticipated and resolved programmatically, they should be reported to the infrastructure exception handling service.

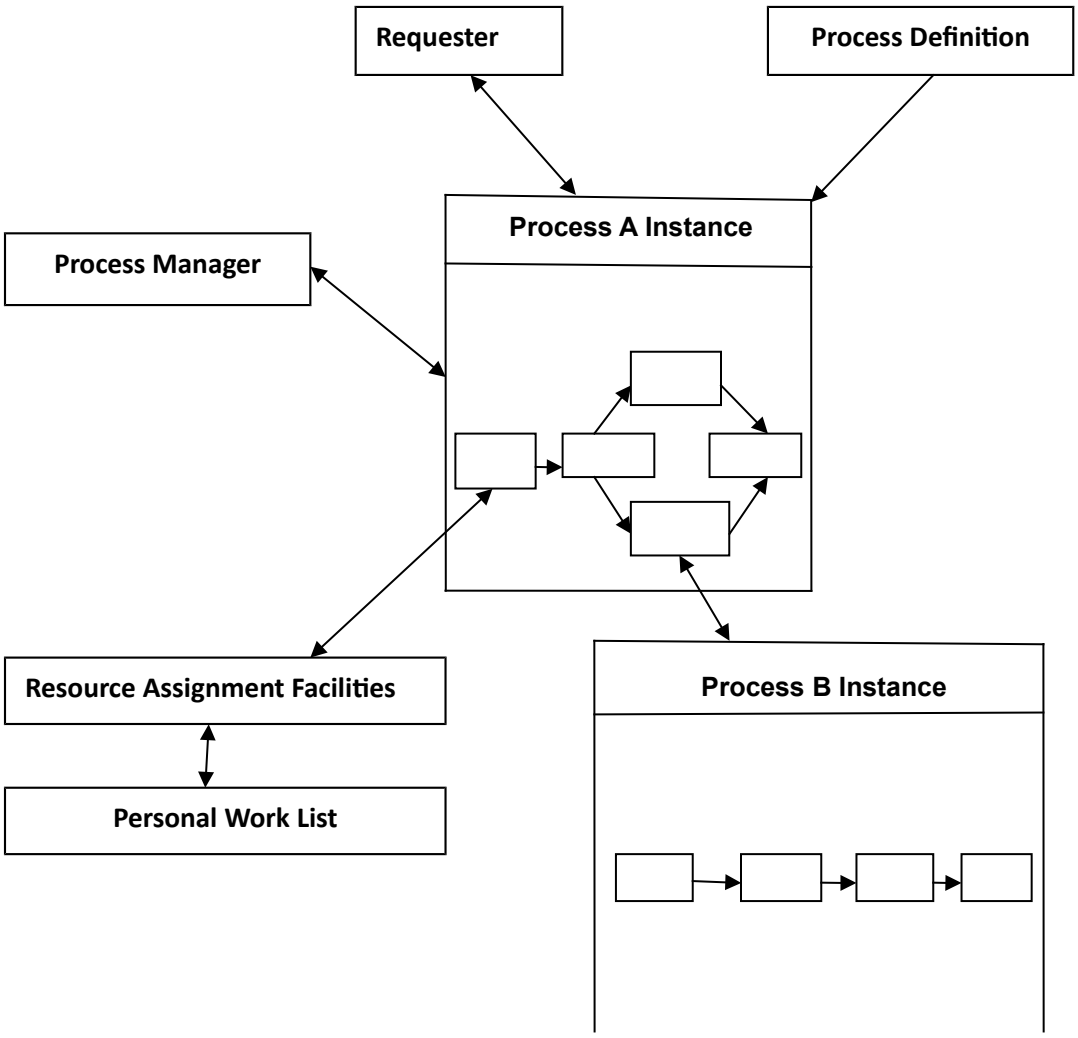
Chapter 7
Integration Workflow Management

Workflow management is a key element of the enterprise integration architecture. It provides the automation of business processes to coordinate the activities of people and applications, potentially at all levels.

Workflow management products have been available for more than a decade. However, early products focused on the automation of paper flow within a local organization, often in conjunction with document archiving technologies. More recently, workflow management products have become more scalable and provide Web access. Work on standards has opened the door for the integration of different products, thus further enabling enterprise level integration, particularly where multiple products are already in use.

General Workflow Model

This section will examine the nature of workflow management based on the workflow model discussed in Chapter 3. The workflow model diagram from Chapter 3 is repeated here as Figure for easy access. The following paragraphs review and elaborate on the roles of the components.



Process Design Considerations

Many factors must be considered in the design of processes to meet business function requirements. Examining all such factors would require a separate book. Here we will focus on more general design considerations associated with enterprise integration:

- **Process closure**
- **Process scope**
- **Process state versus subject matter**
- **User roles**
- **Accountability and control**
- **Process back-out**
- **Long-running transactions**
- **Ancillary actions**

Process Closure

A process should be viewed as a closed loop that accomplishes an objective of the organization. A request for a process implies certain expectations on the part of the requester. Completion of the process should yield the response to that request-the fulfillment of the organization's responsibility. This means that a process is not completed by passing responsibility for remaining work to another process. Instead, if work is to be delegated, the delegating process should make the request and wait for completion of the sub-process. The sub-process has responsibility for fulfilling the delegated work, but the requesting process has responsibility for responding to the initial request.

Process Scope

The scope of a process should be confined to a business system domain (BSD). We discussed the nature of a BSD in detail in Chapter 5. It is one or more applications that are tightly coupled, maintain a shared, consistent database, and are owned by a single organization. This means that the process is owned by the organization that owns the BSD, and the execution of the process occurs within the BSD. There are several implications here. First, the organization that owns the BSD likely will use a single workflow management product.

Process State versus Subject Matter State

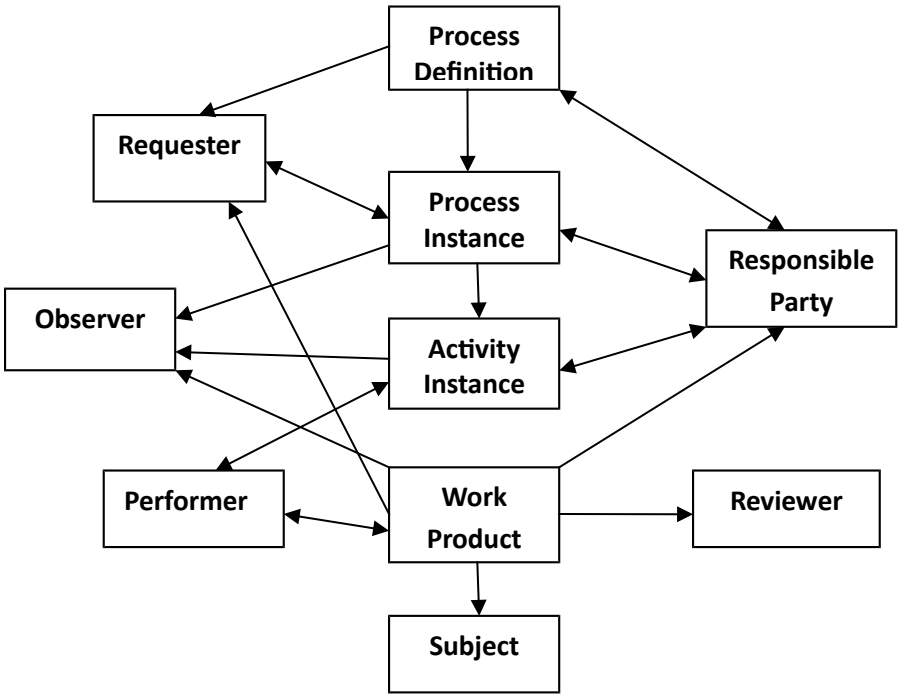
A process has a state that reflects its progress in the execution of activities along with other variables that define the context in which the process is being performed. A process also has subject matter, that is, information about the things the process is acting on. There is a subtle distinction here that should be reflected in the design of the process state and the subject matter object(s).

The process controls the sequence of activities. Its state reflects the activities currently active and variable values that are needed by the process to perform the requested function. The subject matter reflects the state of a business entity, something that has existence in the business, independent of the process. A purchasing process may be in an issue request for proposals activity. A purchase order and the associated product specifications are the subject matter; they contain the information necessary to perform the purchase.

User Roles

A workflow process may involve a number of different users in different roles. These roles define the relationship of the user to the process and the work product, as depicted in Figure 7.2. As a result, roles relate directly to the manner in which each user participates in the process and the user's authority in the context of the process.

These roles are presented as a foundation for the definition of roles in specific workflow processes. Some processes will not realize all these roles; some processes may realize additional roles. Roles need not be defined in exactly the same way in every process. However, the use of consistent role types and names will make it easier to organize and understand the business processes.



User Role

Requester

A requester is a person responsible for initiation of the process. The requester could have been assigned this role in one of three ways:

- **The requester initiated the process directly.**
- **The requester initiated another process that requested this process.**

The requester was the responsible party for a process that requested this process. In each case the person filling the requester role is determined when the process is initiated. There can only be one requester for each process instance, and that requester will seldom change. The requester provides the basis of authority for execution of the process.

Observer

An observer is simply an interested person. An observer can obtain information about the state of the process or the work product, but is not authorized to change anything. Observers are not actively engaged in the process but have an interest for other reasons, such as concern about the duration of the process or the quality of the result.

Depending on the nature of the process, there may be several different observer roles. An observer may be able to view only the state of the process. Another observer may be allowed to view the state of the process and the work product. Yet another observer may be able to view the process definition and history of the process instance.

Performer

A performer is a person assigned direct responsibility for the result of one or more activities. There can be only one current performer for an activity so that there is only one person directly responsible for the result. A person becomes a performer when he or she selects the associated activity from a work list.
The performer is the only role authorized to change the work product. If others were authorized to change the work product, then the performer could not be held responsible for the result.

Subject

The subject is a person the process is about. For example, this could be a personnel process where the subject is an employee recommended for a raise, or it could be a customer whose car is being serviced. The subject may have very limited access to the state of the process.

Reviewer

A reviewer is somebody who will examine the work product and make an assessment. A reviewer could be a peer or a manager of the performer, or it could be someone concerned about the quality of the work product based on a relationship to the requester. A reviewer could be a person with authority to accept or reject the work product, thus affecting the resulting activities of the process. A reviewer may be selected or assigned when the process reaches a point where a review is required.

Responsible Party

A responsible party is responsible for the overall execution of the process. This person will monitor the process instances, receive alerts if there are errors or delays, and make changes in assignments if a current performer is unable to continue the process. The responsible party may be able to redirect the flow of a process instance to resolve a problem.

Accountability and Control

As with application programs, computers execute workflow processes. However, unlike most application programs, where humans are involved primarily in data entry, workflow processes may involve the participation of humans in many activities along the way. This requires closer attention to accountability and control, particularly where money or other resources may be at risk.
Actions reported by humans should be recorded for future reference, including the person, time, and date. In some cases, mechanisms may be required to ensure non-repudiation, such as where the person is a customer or the employee of a business partner.

Process Back out

When application programs execute within a transactional context, a failed series of actions can be backed out automatically by the transaction control mechanism. The states of affected objects will be restored to their states prior to the beginning of the transaction.
Workflow processes involve many computational transactions. Typically, the results of each activity will be committed when the activity is completed, and additional commits may be performed at intermediate points for some activities. Consequently, when a process fails or is canceled, there is no automatic mechanism to undo the actions already performed. The back out for a process may be one or more alternative paths in the process, depending on its state when canceled, or there may be one or more separate processes designed to perform the necessary actions.

Long-Running Transactions

On the other hand, workflow processes can be used to break up long-running transactions into more manageable chunks. An application process may involve multiple interactions with a person or delegation of operations to other applications with unpredictable response times. If the full

process is undertaken as a single computational transaction, referenced objects may be locked for an extended period of time, and if an error occurs, all work up to that point could be lost. If a workflow process is used to manage the activity, then units of work can be accomplished by individual activities and committed as they are completed. The workflow process ensures that the sequence of activities is performed and that there is no risk that completed activities will need to be redone in the event of a failure. Of course, it may be necessary to design back out processes to handle the occasional cancellation of a process.

Ancillary Actions

Sometimes circumstances in a process will require ancillary actions to be performed. For example, in fulfilling an order, the fulfillment process may determine that the inventory is now below a reorder threshold. The fulfillment process could invoke the reorder process, but this might delay the fulfillment process. There are two ways such actions may be initiated:

- **With a non-blocking process request**
- **By issuing an event**

If an activity performs a non-blocking process request, the requested process will be initiated, but the requesting activity will not wait for its completion. Support for this type of request will vary among workflow management products. The workflow management system will assure that the process is performed, but the requesting process does not depend on the completion or the

Integration Elements

Workflow management must function in the context of a BSD and interact with applications and other BSD facilities and services. This section will discuss the following elements of workflow integration:

- **Process initiation**
- **Workflow context**
- **Application adapters**
- **Event detection**
- **Asynchronous process invocation**
- **Shared work products**
- **Security**

Process Initiation

Processes are initiated in one of three ways:

- **By a programmed request**
- **By a user**
- **By an event**

A programmed requester has the same form as an invoking process, but it could be an application or other facility presenting the interface. A requester requests the instantiation of a process and passes parameters that form the context of the process instance execution-directly or indirectly defining the subject matter and parameters that will affect the execution of the process. The process will notify the requester when the requested process is completed. The requester also will be given a reference to the process instance so that it can direct messages to it during execution, such as status requests.

Workflow Context

The context of a workflow process is the set of parameters associated with the execution of a process instance. These parameters will include references to persons in different roles, subject matter objects, and variables that will affect computations or how the process is performed.
The Object Management Group's (OMG) Workflow Management specification defines the process context as a sequence of name-value pairs. The initial names and their values are determined by the process definition and the particular request. Additional names and values may be added as

the process executes to capture intermediate and final results. With a minor modification, the process context can be much more flexible and useful. Rather than specifying the parameters in name-value pairs, simply pass one parameter that specifies an Extensible Markup Language (XML) document.

Application Adapters

While workflow management products will provide ways for a process to execute an application, it is desirable to define a consistent interface to applications using adapters. The consistent interface will simplify programming and can facilitate the better recording of activities and communication with users.

It is suggested that the interface presented to the workflow process be equivalent to a workflow process interface. The application is then invoked in the same manner as a sub-process. If the workflow context is specified in XML, as described earlier, then the application parameters and results can be captured and accessed in an XML document defined for the application. The benefits derived from the XML context document described earlier then apply here as well.

Event Detection

Processes frequently wait for events. In some circumstances, an event may signal the completion of a necessary action such as execution of an application. In other cases, an event may signal an unexpected situation that should alter the execution of the process. Activities that normally fail to complete may wait for these unexpected events while other activities proceed with the normal work of the process. In either case, the waiting activity must be informed of the event.

Different workflow management products may provide different mechanisms for event detection. The simplest form is for the activity to check periodically for a change in a persistent data value. This could be a file or database entry uniquely identified for the event of interest to the activity, or it simply could be a check for the existence of a file produced by an invoked application.

Asynchronous Process Invocation

A process in one BSD may invoke a process in another BSD, just as it may invoke a process in the same BSD. However, the process in the remote BSD must be invoked asynchronously so that the BSDs remain loosely coupled. T he key is to use XML so that the information exchanged can be passed through messaging, but it also can be passed through a firewall using the Hyper Text Transport Protocol (HTTP) to invoke a process in a business partner BSD.

Shared Work Products

When applications execute in a shared object environment, their transaction context provides the basis for resolving concurrent accesses. Objects are locked when in use and released when the transaction commits, thus forcing serialization of access.

Workflow processes involve a series of computational transactions. In order to achieve predictable results over these long-running transactions, the process must retain exclusive control over shared work products. This typically is achieved by a checkout, check in mechanism, but it could be achieved using an independent concurrency control service or optimistic locking. The choice of mechanism may depend on the availability of a document management facility and the nature of the shared work products and their usage.

Security

In some ways, workflow management can simplify security authorization. Earlier we discussed six different types of roles: requester, observer, performer, subject, reviewer, and responsible party. Individuals will take on different roles in different processes and in different process instances.

The process logic is primarily responsible for determining the assignment of roles. Some role participants are determined by the source of the process request. Some role participants are

determined by the process by which work is assigned. Still other role participants are determined by their relationships to the existing participants, the process, or the subject matter.

Assessing Scalability

Most workflow management products were designed originally to meet the needs of managing departmental paperwork. As the scope of workflow management is expanded, the scale of activity may increase orders of magnitude. Enterprise-level workflow will become key to the operation of the business. It is important to assess the scalability of any product being considered.

The scale to be considered must go beyond the processes initially identified for workflow management. Even though the scope of application of a workflow product is limited to a BSD, once the technology is available and integrated, the number of defined processes and users will increase, with associated increases in the volume of activity. Consequently, the scalability assessment should anticipate considerable growth.

The following subsections will examine key factors to be considered in assessing scalability:

- **Process definitions**
- **Active processes**
- **Number of users**
- **Threading model**
- **Execution distribution model**
- **Workload balancing**

Process Definitions

Workflow management may be targeted for one primary business process. This may not seem like much. However, a well-designed process depends on many sub-processes, just as a well-designed computer program often will have many subordinate functions or subroutines. Special processes should be defined for the assignment of tasks, for handling exception situations, for accessing applications, and so on. A mainstream process could spawn the definition of a hundred supporting processes.

This may still not seem like a lot of process definitions, but it could have a performance impact if the workflow management product is designed assuming a small number.

Active Processes

A more significant scalability factor is the number of active processes. These should be considered at two levels:

- **The number of processes that have been initiated and not completed (pending)**
- **The number of processes that are interacting with users or applications at a single point in time (active)**

Consider, for example, an order-processing process. If an enterprise handles a million orders per year and it takes a week to complete handling the order, including shipping and invoicing, then almost 20,000 process instances will be pending at any point in time (1 million/52 weeks). If the number of orders is higher or the order cycle time is longer, then the number of pending processes will increase.

Number of Users

Again, traditional workflow management installations involve small numbers of users-the members of a department. In enterprise installations, the number could be in the thousands, particularly if the workflow process has customers or suppliers as participants. Even if the participants are within the enterprise, some processes, such as for financial and personnel functions, could include all employees as potential participants.

As with the number of processes, the number of users could exceed table limits if information is held in computer memory. Similarly, the number of concurrently active users will have an impact on performance both in terms of computational requirements and in terms of network bandwidth. The design of Web pages can be a significant factor if access involves the downloading of large graphics or applets. Again, the workload of workflow management must be combined with the workload of associated applications that may generate additional Web pages.

Threading Model

Large-scale systems require multithreading-the capability to process many independent requests concurrently-without requiring that one request be completed before the next can be handled. Some workflow management systems may have been developed for single-thread operating systems and may not be able to take advantage of multithreading. When there are many users and many processes, a multithreading implementation will be essential.

Execution Distribution Model

Workflow management products for large-scale applications should be designed for distributed processing. Even if they are, the manner in which the execution is distributed may have a significant impact on performance. The key performance factors will be database accesses and network messages. If the execution elements of a single process are finely distributed over multiple servers, then there may be excessive network activity. If process elements (such as process context, subject matter objects, or user information) are retrieved repeatedly from the database for every transaction, then there may be excessive database activity.

Load Balancing

Load-balancing facilities for the execution of workflow processes may affect the capability of the system to adapt to shifts in the level of activity in different types of processes or communities of users. Shifts may occur gradually over time, or they may be cyclic on a daily, weekly, monthly, or annual basis. The importance of load balancing may be a function of the nature of the business. However, more adaptive load balancing will reduce the risk of performance problems during peak periods.

Product Requirements

There are a great many workflow management products in the marketplace. They have diverse capabilities and features. It is important that the product or products selected provide robust process definition capabilities, meet scalability requirements, accommodate integration with other system components, include appropriate management tools, and provide appropriate user interfaces and security controls.

While most workflow management systems in the past focused on document management, two new classes of workflow management have emerged more recently. One class has emerged from enterprise application integration (EAI), where tools have been developed to manage the flow of messages between applications. These products generally do not deal with human participation, but focus on message transformation and content-based routing. The other class has emerged from the development of Web portals. They focus on guiding a particular user through a series of steps to complete a desired action. These processes are limited in scope and generally involve only the initiating user.

The capabilities and features are discussed under the following topics:

- **Process definition elements**
- **Process definition tool**
- **Workflow execution**
- **Ad hoc changes**
- **Runtime user interface**
- **Compute environment**
- **Security**

Process Definition Elements

The process definition elements define the language syntax for the definition of processes. The following list describes features and functions that are needed for the definition of processes:

- **Polling/status check activity. An activity can be designated to poll or check a status indicator periodically on a specified time interval. This could be an ongoing check for a variable out of limits, or it could be, for example, a check on the activity of an application to determine if it has reached a desired status. Polling typically is used in the absence of triggering by events.**

Process Definition Tool

Workflow management products currently include a build-time tool for the creation of process definitions. In the future, these tools may become independent products. In addition to supporting the definition of processes with the elements just described, the tool should have the following characteristics:

- **Graphic interactive tool. The processes are displayed graphically, and the process elements are added and connected with drag-and-drop interactive editing.**
- **Visual distinction of task types. The type of each process element must be visually identifiable in the graphic representation of the process.**

Workflow Execution

The following list describes features that a workflow product may incorporate in the process execution:

- **Process definition versions. When existing processes are changed, active instances of the same process could be affected. Versioning distinguishes between old and new processes. This is particularly important when the processes are long-running because the new and old processes may both be active for some time in the future.**
- **Process version effectively control. Process definitions will be changed from time to time. In some cases, the effective date must be coordinated with changes to other processes or business circumstances. A new process definition should replace the existing process with the same identity only after the effectively date.**
- **Multiple active versions. When a process definition is replaced, there may be active instances. Changing the process definition for these process instances could produce undesirable results. It should be possible to continue the execution of active processes using the obsolete version (or versions) until each process instance is completed.**
- **Event-triggered processes. A process can be initiated as the result of the receipt of an event. This capability typically is not integrated into a workflow product due to the lack of consistent sources and formats of event messages.**
- **Event-triggered activities.* An activity within an active process can be started or removed from a wait state as a separate path in the process as the result of the receipt of an event. This capability, based on the receipt of an external event message, is not typical of current products. Similar results should be possible when an event can be detected with an activity start rule.**
- **Interactive debugging support. The environment should support interactive examination of processes and activities as well as context variables to support debugging.**
- **Stepped execution control. It should be possible for a developer to execute a process step by step to examine the flow and changes in variables for testing.**
- **System environment library functions. A library of functions must be provided for access to common system environment variables and functions, independent of the platform.**
- **Propagation of preemptive completion. A process should forward a notice of termination to active sub-processes when it is brought to either normal completion or abnormal termination so that the sub-processes will be terminated.**

Ad Hoc Changes

Sometimes the normal sequence of events may not be appropriate for a particular process instance. The workflow management product should allow an authorized person to effect changes. A number of capabilities might be provided:

- **Skip or restart activities. One or more activities can be skipped or restarted. Restarting an activity may be appropriate when an external action failed and must be repeated.**
- **Reassignment/replacement of resources. It must be possible to remove or replace resources, particularly people, for a particular process instance.**
- **Alter the sequence of activities. A different sequence of activities can be defined for a specific process instance beyond those activities that are currently active.**

Runtime User Interface

The following list describes the desired characteristics of a runtime user interface:

- **Browser-based. The user interface should be Web browser-based so that it can be accessed from diverse workstations and from remote locations.**
- **Application user-interface integration. The workflow user interface should be compatible with the user interfaces of associated applications for ease of use.**
- **Display of process status. There should be a display capable of depicting the current activities of nested subprocesses.**
- **Graphic display of an active process. There should be a graphic display of an active process depicting the status of the activities.**
- **Process/activity context display. It should be possible to display process and activity variables for an active process instance.**
- **History/trace display. A display should depict the sequence of activities executed within a process instance up to the current time. An extended display capability might depict activities remaining to be executed. However, this is unpredictable for tools that base activity initiation on rules instead of explicit activity dependencies.**
- **Process definition. It should be possible to display the process definition for an active process so that the user can see all activities that could be executed and their relationships as the process proceeds.**
- **Work lists. Personal work lists should indicate the nature and priority of pending tasks and the current status of processes and activities for which a person is assigned responsibility. The person should be able to access additional information about each item from the work-list display.**
- **Runtime statistics. The user should have access to current runtime statistics such as queue lengths (for example, work lists), the number of active processes, activity and process durations, and processes completed within a time period.**

Compute Environment

The workflow management facility must function in a computing environment in a manner that is compatible with the environment and the other applications and services with which it interacts. The following list describes several aspects of this integration:

- **Transactional context. The execution of workflow process steps and the associated applications and services invoked by workflow processes, as well as those that invoke workflow processes, must function in a transactional context so that concurrent actions are serialize-able (concurrent accesses are resolved) and actions completed (committed) are recoverable.**
- **Messaging. The workflow process steps must be able to interoperate with applications and other objects through an object request broker (synchronous messaging over a network).**
- **Database. The state of workflow processes and the associated context data must be made persistent in a database to ensure recoverability.**

- **Multiple platform implementations. Compatible versions of the workflow management system should be available for deployment on a variety of platforms. Specific platforms of interest should be identified for the enterprise.**

Security

Using appropriate security facilities and services, the workflow management system must support access controls as described in the following list:

- **Integration. The workflow processes and the data on which they operate must be secure from unauthorized access or modification. It must be possible to appropriately integrate security controls with those of the BSD computing environment.**
- **Process initiation. Processes must be initiated only by authorized individuals or by other authorized processes or services.**
- **Process changes. Changes to process models as well as ad hoc changes to process instances must be restricted to authorized individuals.**
- **Process status. Access to process status information must be restricted to authorized individuals based on the process context and assigned roles.**
- **Process assignments. The control of assignments to workflow processes as well as access to information about assignments must be restricted to authorized individuals.**
- **Work-list status. Access to a personal work list must be restricted to the individual assigned to the work list and authorized managers of the individual or the process.**
- **Data access. Access to process context data must be restricted to authorized individuals based on the process context and assigned roles.**
- **History access. Access to process history data must be restricted to authorized individuals based on the process and assigned roles.**
- **Employee privacy. The workflow management system must support the need to protect employee privacy in compliance with local governmental regulations and company policy.**

Analysis Tools

Workflow products provide various tools for analyzing process activity. The following are key capabilities:

- **Runtime statistics. It must be possible to report statistics on the status of the active system, such as queue lengths, the duration of activity executions, the number of active participants, process durations, and so on.**
- **Variation exception reporting.* Exceptions should be reported for process variables that exceeded defined limits, such as the duration of activity, queue length, or percentage utilization.**
- **Process flow analysis. Analysis tools should be available to analyze process history and report the use and duration of processes and activities, the use and workload of resources, particularly people, and the points at which bottlenecks have occurred.**
- **Process integrity check. A tool should be provided for build-time analysis of a process model to identify logical inconsistencies such as loops or open ended paths. Such tools can expedite the development of reliable business processes.**
- **Simulation.* It should be possible to use process models to simulate process activity. Such models can be used to explore the impact of alternative processes, the allocation of work, or applications of technology. This requires components to simulate the submission of requests, interaction with external applications, and the assignment and participation of resources.**

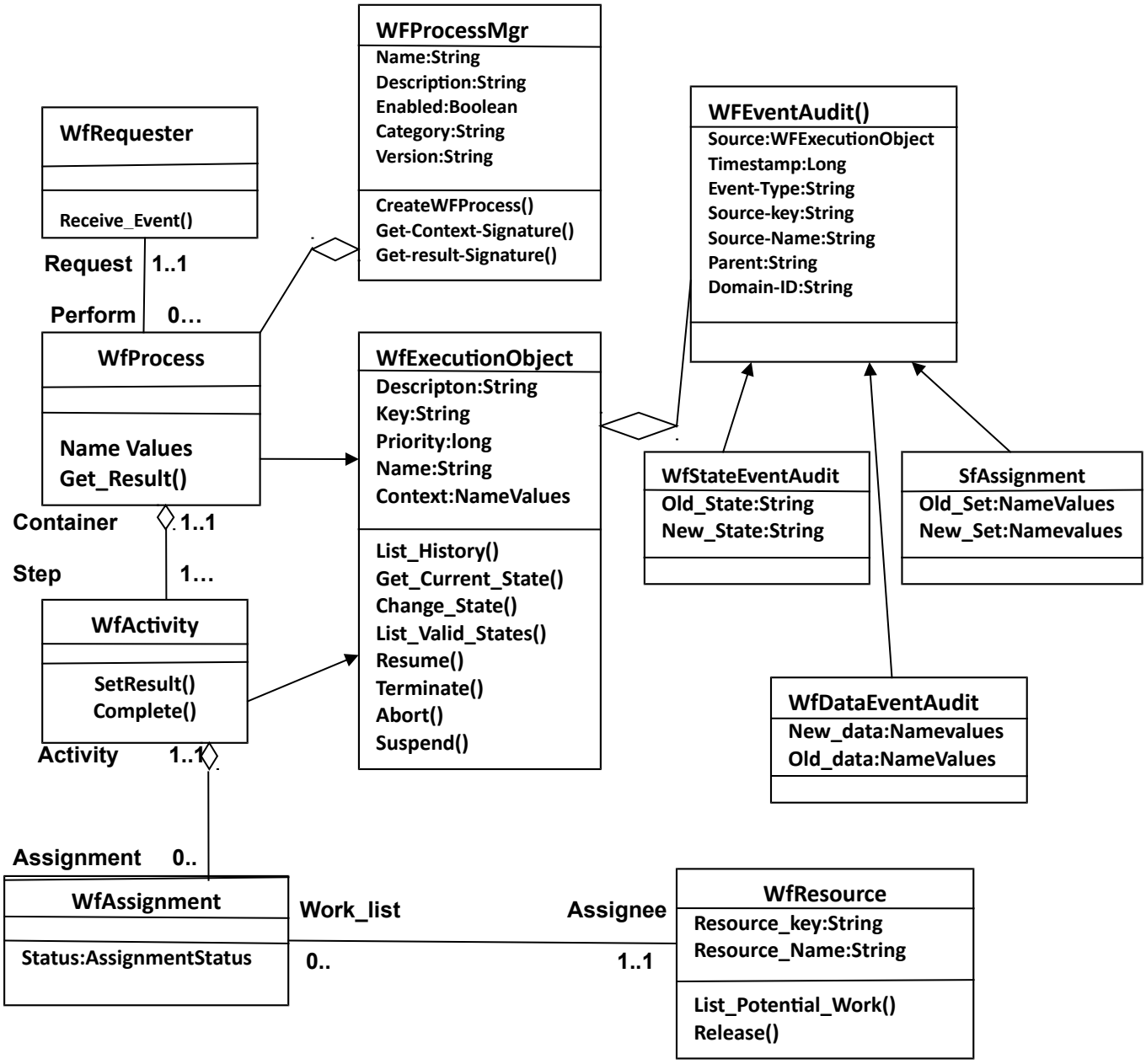
Standard

The BSD architecture and integration of workflow products should be based on industry standards so that products will be interoperable, they can be replaced by better products in the future, and process designers and users can rely on some degree of consistency in design principles and concepts. In this section we will discuss the following standards:

- **Workflow management interoperability**
- **Resource assignment**
- **Organization structure**
- **Unified Modeling Language (UML) workflow process definition**
- **Loosely coupled process requests**

Workflow Management Interoperability

The OMG has adopted a specification for interfaces to support interoperability between independently developed workflow products. Figure shows the defined interfaces in a UML diagram. The key interfaces are WfProcess, WfActivity, WfAssignment, and WfResource.



OMG Workflow Interface

These interfaces allow a BSD to have multiple workflow management products that work together. They also facilitate the replacement of a workflow management product in the future. If a selected product does not conform to these specifications, it would be advisable to implement adapters for interfaces used in the integration in order to support interoperability, maintain vendor independence, and prepare for the future adoption of these interfaces.

Resource Assignment

The OMG is developing a specification for interfaces to workflow resource assignment facilities. This specification will provide a standard way to request the assignment of resources so that shared resources could be assigned to processes implemented in different workflow products. This is particularly important for the assignment of people, where each person should have one work list regardless of the source of assignments.

In lieu of the adoption of such a specification, the allocation of resources should be implemented as workflow processes. The interface is then simply a standard process interface with parameters defining the resource requirements. One workflow product can perform resource assignments, while others can perform the business processes. Implementing a resource assignment as a workflow process makes the assignment process visible and more flexible. This implementation can be replaced later by more a sophisticated resource allocation product if appropriate.

Organization Structure

The OMG has adopted a specification for an organization structure facility. Organizational information is important for the assignment of work and the routing of requests for approval. When incorporated in a workflow management product, these specifications will allow an organization structure facility to be purchased as an independent product and be used to support workflow management as well as other enterprise applications. Current products have organization information embedded in their process specifications. This should be separated to be a more consistent and robust representation that is available for other purposes as well.

UML Workflow Process Definition

The OMG is currently developing a UML profile (modeling subset) for the specification of workflow processes. This will enable the independent development of process definition tools and the portability of process definitions between different runtime workflow management products. When workflow processes are specified in UML, they will implicitly be expressible in XML Model Interchange (XMI) language. This will enable the exchange of process models between different tools and workflow execution products.

A standard modeling language for workflow process definition will not have an immediate effect on the application of workflow systems, but in the future it will improve flexibility and probably the quality of process modeling.

Loosely Coupled Process Requests

The Workflow Management Coalition (WfMC) has adopted a specification using XML and Simple Object Access Protocol (SOAP) as the form of messages to be exchanged for interoperability between workflow management products. This specification is based on the OMG/WfMC interoperability specification discussed earlier. The messages (and response messages) are asynchronous, and the XML form facilitates communication through firewalls.

This was viewed as a way of extending the reach of workflow integration across the Internet.

However, this level of granularity of interactions is seldom, if ever, necessary between loosely coupled domains. A process request should be expressed as a single message with return of an acknowledgment. The acknowledgment should include a process instance identifier for future reference. A later message should return the process result. The Uniform Resource Locator (URL)

of the request may determine the selection of the appropriate process, and a result URL may determine the routing of the result.

Chapter 8
Providing Web Based User Access

The Web is rapidly becoming the universal medium for user access to business systems. This is true for employees as well as customers and business partners. Web-based access allows users to interact with systems from anywhere as long as they have a computer running a standard Web browser.

Web-based user access brings four important elements:

- **Users can have a variety of computing platforms.**
- **Users need not be members of a defined community.**
- **Users can access systems from anywhere.**
- **User access costs are low.**

For these reasons, we should expect all systems in the future to provide Web based user access. Alternatives are less adaptable, restrict user mobility, and are more expensive.

In this chapter we will examine the client and server facilities that bring the user and the application together through Web-based user access. We will examine these complementary roles and their interactions under the following topics:

- **Web access environment**
- **Client facilities**
- **Web server facilities**
- **Session management**
- **Specialized client devices**

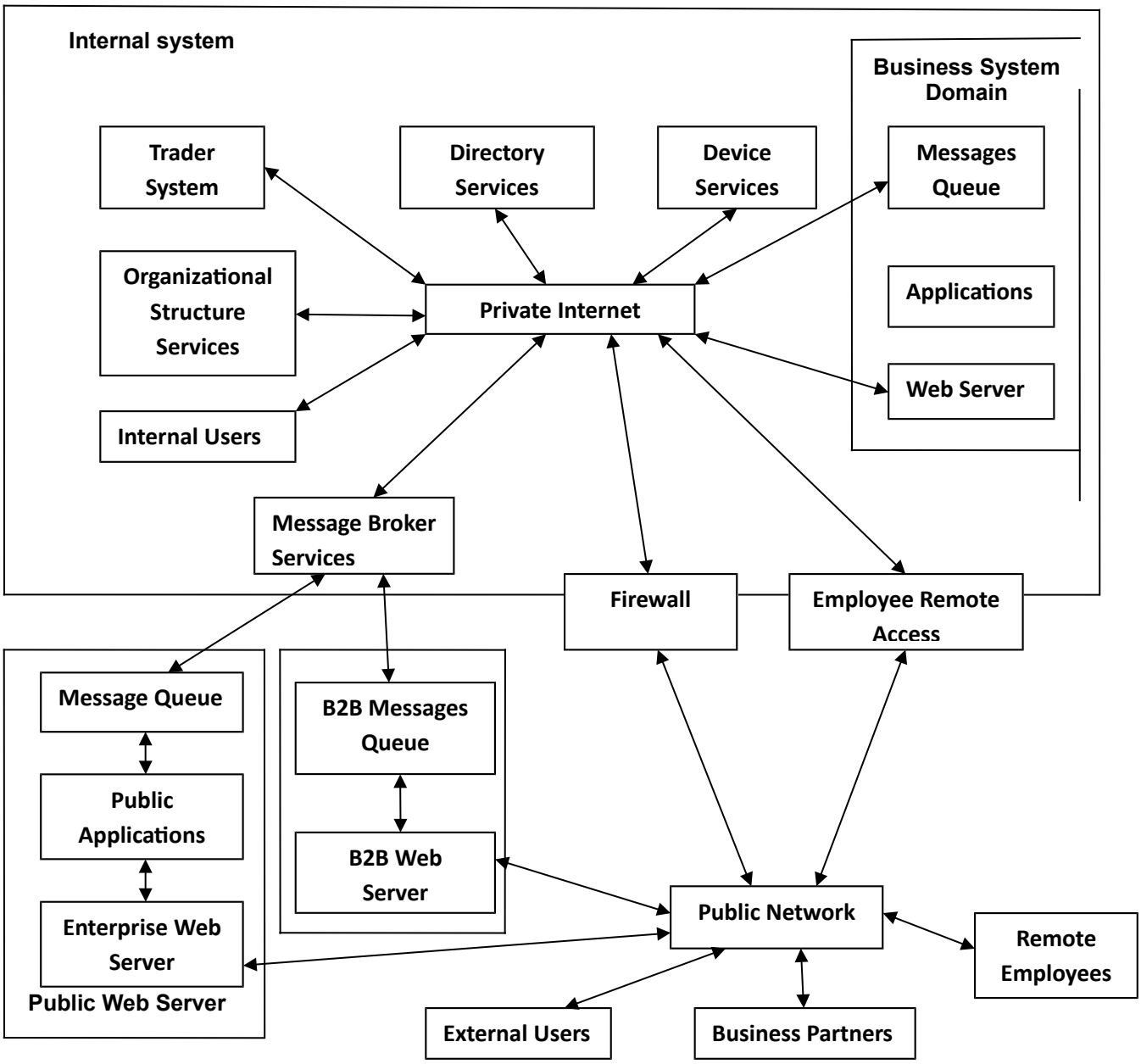
Web Access Environment

Traditionally, access to business systems has been confined to a small, well defined group of users in one organization and in one office facility. The n ew generation of systems has an expanded community of users. Many members of the expanded community are employees from diverse activities of an enterprise who need information or who need to coordinate their activities. For some applications, the community of users extends to business partners and customers. A single business function can no longer dictate the device or software used to access the system, nor can the physical location of the user be predefined. Instead, a user comes equipped with a Web browser as the primary mechanism of access to any authorized system.

Within the enterprise, employees will have access to enterprise systems through internetworking of local-area networks (LANs). Outside the enterprise, customers, business partners, and employees away from the office will access enterprise systems over the public Internet and through enterprise portals. From the application standpoint, the same protocols will be used whether the user is inside or outside the enterprise. The accessibility of systems will be determined only by security constraints.

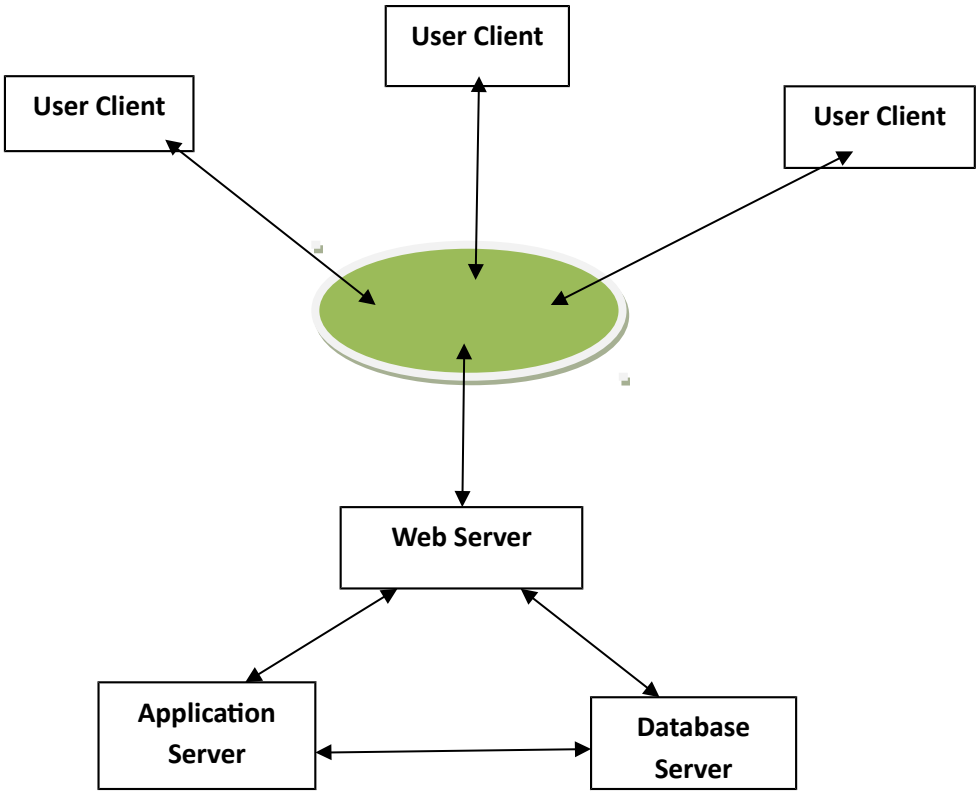
Most enterprises are in a state of transition to the Internet era. Legacy systems were not designed for Internet access. Some of these legacy systems will continue to exist for many years to come. At the same time, competitive pressures will demand that many legacy systems either be Web-enabled or be replaced by systems that provide Web access. The pace of business has quickened. Email and cell phones have increased the timeliness of communications between humans, but demands for streamlined business processes cannot be met by people acting as intermediaries between systems or by batch file transfers. The availability of Web technology allows people to access many different systems directly, and inputs are not expected to be held for a day until accepted by a batch process. The demand for streamlined response makes Web-based access to business systems a competitive necessity. Web servers now can provide highly functional user interfaces through browser-based thin clients. The functionality of the Web-based

interface has become key to streamlining interactions between humans and systems and thus the streamlining of business systems.



There are systems where a client is tightly coupled with an associated business application using synchronous message communication. This makes the client another node in a tightly coupled distributed computing environment. The goal is improved performance, but this extends the functionality of the application into less secure environments, and mechanisms for message exchange and security are more complex. This tight-coupling approach seldom will provide advantage with the increased functionality of current Hyper Text Markup Language (HTML). In addition, the ability to extend browser functionality with plug-in applications enables the

development of highly functional and interactive applications for the client while preserving a loose coupling with the Web server through HTTP.



Web Application Configuration

Client Facilities

The participation and support provided to users and the performance, security, and functionality of enterprise applications depend on an appropriate balance of functionality and communications between the client system (that is, the user's computer), Web servers, and application servers. In general, the primary factor is the level of functionality and performance required on the client. The goal is to minimize the complexity of the client, avoiding application-specific software and security exposures while meeting the user-interface business requirements.

There are a number of factors to be considered:

- Provide adequate interactive performance. If the user is engaged in the composition of a complex work product, this may require local computational support to provide an appropriate level of performance.
- Ensure the integrity of system updates. Client systems likely will be at greater risk of exposure and tampering because they are typically in less secure environments. This is particularly true of portable computers. The design should minimize dependence of the integrity of enterprise systems on client applications.
- Minimize the risk of exposure of confidential data. In order to minimize the risk of exposure of confidential data, it may be appropriate to minimize the amount of data downloaded to client systems. By minimizing client functionality, the data downloaded can be limited to that to be viewed by the user. The data also may be kept in a transient form rather than being stored in client files that could create a risk of later exposure.

- **Minimize network activity. While the typical office Minimize network activity. While the typical office environment may provide high-speed Internet communications, the system design should not assume that users will always be in the office environment. Remote communications often will be slow and unreliable. In addition, even with high-speed communications, the number of exchanges performed increases the load on servers significantly.**
- **Enable disconnected user activity. If a user is mobile or communication facilities are unreliable, it is desirable that the user be able to continue to conduct business without always being connected. This may call for periodic download of relevant information and upload of user inputs so that the user normally can operate in stand-alone mode.**

In the subsections that follow we will examine implementations of different levels of client functionality:

- **Static pages**
- **Forms**
- **Active content**
- **Browser plug-ins**
- **Stand-alone applications**

Static Pages

The simplest form of user interface is a textual or graphic display. A user specifies a particular Uniform Resource Locator (URL) to the Web browser, the browser requests the document from the Web server, and the browser formats the display as specified within the HTML document returned.

For example, a retail customer might want to access specifications on a product using a specified URL: http://www.ExampleRetailer.com/products/specs/lawnmower/turbo2.htm

There are few Web pages that are simply text-only documents. More often, a Web page is composed of a number of components including both text and graphics. The composite Web page consists of a primary document retrieved from the specified URL but with embedded references to other URLs for related components such as images and sound clips. In addition, some Web servers will recognize server-side include commands that will cause data from another Web server resource to be embedded in the HTML document at the server.

Some of the embedded URLs are links to other Web pages for the user to select; others are inline images or other components to be incorporated into the display. When the browser encounters an inline component, it issues another request to the server to retrieve the component. Consequently, one initial request may result in a number of requests, all occurring within the same network connection, in order to complete the requested display.

HTML defines a fixed set of element types for the specification of Web pages. For static Web pages, these include elements that define the document structure, such as head, body, and paragraphs, elements to define layout and appearance, such as headings, fonts, frames, lists, and tables, and links to other Web pages.

Cascading Style Sheets (CSS), an extension to HTML, provides a separation of formatting information from content. The basic concept of a markup language is to define the content of elements of a document so that it can be displayed appropriately in different formats. Thus, HTML defines headings, paragraphs, frames, tables, and so on so that a browser can format these elements to fit in a window. At the same time, various formatting features, such as relative positioning, fonts, and colors, are needed as formatting instructions. A number of formatting instructions have been included in the language to be embedded in the specification of individual elements. CSS formatting specifications also can be included inline in elements, but a style sheet can be embedded between the header and body to apply to the entire body of the document. Alternatively, a linked style sheet can be specified in the header of the document, incorporating a style sheet that may be shared by many documents.

Forms

HTML not only provides for display of information, but it also provides for the display of forms to accept user input. User input may be used to retrieve selected data from a database, to perform certain computations, or to create or update enterprise records. The browser does not know how the data will be used; it only provides the mechanism by which data elements can be entered and communicated to the server.

The request for a form looks the same as any other Web page request. The difference is in the document returned. Suppose, for example, that the Turbo2 lawnmower in our customer request for specifications earlier actually came in several models. The initial response might be a form from which the customer could select a particular model. The form could provide an input box for the model number, or it might provide a list of available models with a radio button selection for the model of interest.

Form input cannot be processed by a static Web server resource. The URL for submission of the form data must invoke logic to process the input data. This processing logic will be invoked as a Common Gateway Interface (CGI) script. We will discuss the processing of input later in the section on Web server facilities.

Active Content

Up to this point our discussion has focused on HTML pages with static content-actions are taken by the browser based on the document format specifications and element attributes. In this section we will examine two ways that the Web page can include executable code: JavaScript and applets. These are not the only ways available, but they are supported by the popular browsers, are used commonly, and are safe; that is, the user is protected from code that might corrupt the client platform.

The benefit of active content is that more application logic can be performed on the client, potentially reducing network activity and providing more effective interaction with the user.

JavaScript

JavaScript is a language for embedding executable code directly in an HTML document. It is not a dialect of Java but is a language interpreted by the browser. Code at the document level is used for initialization operations and to define functions invoked from elsewhere in the document. Within individual elements, event attributes specify the functions to be invoked when events occur for the associated elements. Functions can invoke subroutines to support more structured code.