LEARNING MODULE IN IT-SIA01

SYSTEM INTEGRATION AND ARCHITECTURE

AUTHOR:
CRIS NORMAN P. OLIPAS

EDITOR:
DR. RUTH G. LUCIANO

REVIEWER;
HENRY T. ROQUE

FOREWORD

In an era where digital solutions have become a necessity, integrating solutions is very essential. In order to maximize the potentials of different Information Technology (IT) solutions employed in an organization or business entity, integrating them is a must. This course covers the essential knowledge in which students need to acquire to integrate system effectively. Topics include the construction of strong foundation and understanding on system integration, the different system integration approaches, middleware, and Enterprise Resource Plan (ERP) Systems processes, solving integration problems using patterns, XML Application Integration, Web Services, and Advanced Web Services.

At the completion of this learning module, students are expected to: (1) analyze the appropriateness of a decision to in-source or out-source IT services given situation by understanding ERP and the importance of System Integration and the different architectures related to it; (2) create a testing environment and design a stress test using appropriate tools and techniques that impact system performance in relation to available ERP and Web Applications, and (3) implement an enterprise integration middleware platform and/or conduct research activities to apply concepts relevant to System Integration and Architecture (SIA).

TABLE OF CONTENTS

LESSON 1
OVERVIEW OF SYSTEMS INTEGRATION: CHALLENGES AND DRIVERS

INTRODUCTION

As the world continuously advance, business and organizations must keep pace and immediately adopt to these changes in order to take advantage of the benefits that technological advancement brings. In this lesson, you will be introduced to important concepts relating to systems integration, enterprise resource planning, silos in business and IT solutions, different information system applications in organizations, the benefits and limitations of system integration and its implications to management. This lesson also covers the different roles of ERP, its benefits and limits, as well as the types of vendors of ERP. An assessment will be conducted at the end of this lesson to assess the level of your understanding on the topic discussed.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to:

1. Identify what is Systems Integration and Enterprise Resource Planning Systems;
2. Determine what Silos are all about and the different Information Systems in an organization;
3. Understand the benefits and limitations of Systems Integration and its Implications for Management; and
4. Construct deeper understanding on the different roles of ERP, benefits and limits of ERP, and different types and vendors of ERP

DISCUSSION

WHAT IS SYSTEM INTEGRATION?

Systems integration implies that users permit an interdependent Information System (IS) to communicate or connect as well as exchange information (or data) effortlessly amongst each other. Seamless exchange of information (or data) results to increase in productivity, efficiency, and effectiveness of process and procedures performed in an organization. (Motiwalla & Thompson, 2012).

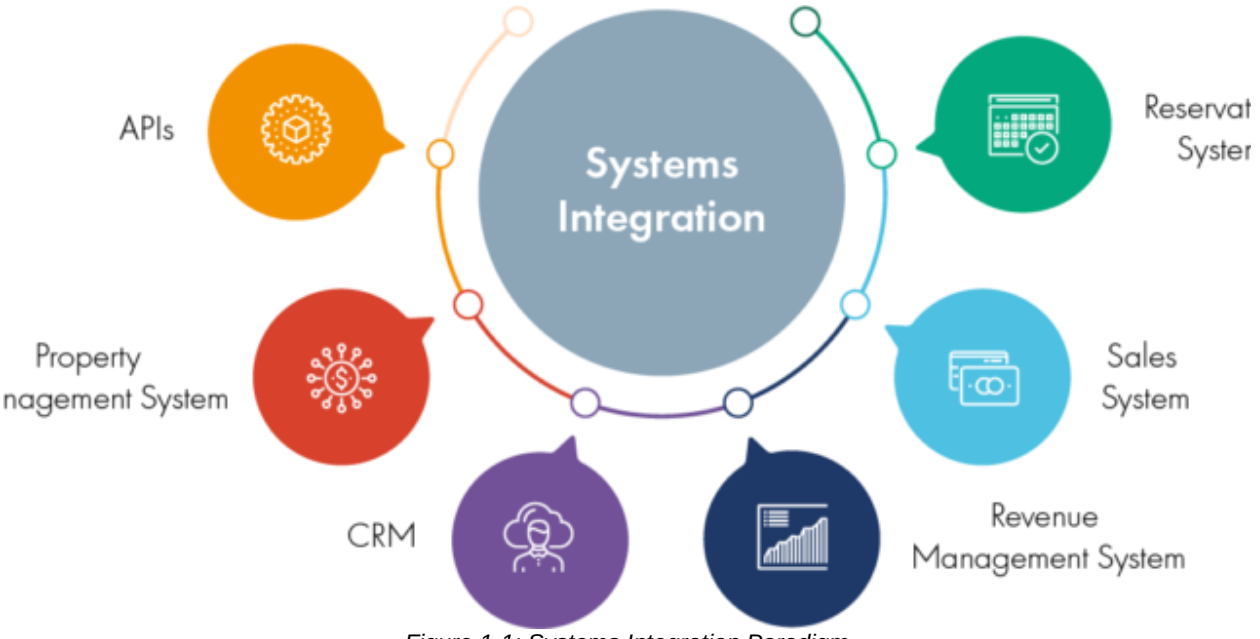


Figure 1-1: Systems Integration Paradigm
Image Source: https://srutatechnologies.com/system-integration/

Lehtonen defined system integration, in its broadest sense, as the act of linking several sub-systems (components) into a cohesive bigger system that functions as a whole. When it comes to software, system integration is often understood as the practice of connecting disparate IT systems, services, and/or software to make them all operate together operationally. (Lehtonen, 2018).

Figure 1-2: Samples of Applications for Integration
Image Source: https://www.indiamart.com/proddetail/custom-it-system-integration-service-19282067162.html

System integration involves combining all of the organization's physical and virtual components. Machine systems, computer hardware, inventories, and other physical components make up the physical components. Data stored in databases, software, and

apps make up the virtual components. The major emphasis of system integration is the process of integrating all of these components so that they operate as a single system.

Integration of systems is a critical problem for an organization's growth. This is a problem that management must pay particular attention to. System developers and integrators face several obstacles as the intricacy of systems integration grows. The integration aspect of the process becomes exceedingly tough as the project has become more complicated, with more users, suppliers, internal corporate procedures, and functions and subsystems. System integration refers to the interoperability and compatibility of items from many sources / firms in a system.

Figure 1-3: Integration
Image Source: http://www.integrationwizards.com/

According to Silva & Loureiro (2011) in an article published during the IEEE International Conference on Industrial Engineering and Engineering Management held in 2011, the following causes of problems in systems integration.

A. Management
    1. Poor Configuration Management
    2. Establishment of inadequate testing philosophy
    3. Improper organization and assignment of responsibilities
    4. Deficient interrelationship of assembly, integration, and test in the project development
    5. Test strategy and integration plan is not developed in time
    6. Test tools and test infrastructure not available for system tests
    7. Insufficient time for testing
B. Technical
    1. Maturity of process
    2. Incomplete requirements
    3. Development of testing software
    4. Deficient test tools at subsystem level

5. Lack of standardization of engineering data
6. Deficient project design
7. Use of technology that requires complex integration project

ENTERPRISE RESOURCE PLAN (ERP)

ERP (Enterprise Resource Planning) systems are a type of information technology that enables businesses to connect many systems into a single, enterprise-wide application with an integrated database management system (DBMS).

Figure 1-4: Enterprise Resource Planning
Image Source: https://www.manifera.com/nl/custom-enterprise-resource-planning-erp-system/

ERP stands for enterprise resource planning, and it is a sort of software that businesses often use to handle day-to-day operations including project management, accounting, supply chain operations, procurement, risk management, and compliance.

Numerous ERP software systems are critical to businesses because they assist them in implementing resource allocation by combining all of the activities necessary to manage their businesses into a centralized platform. (Manifera.com, nd). Some of the advantages of ERP include focused IT cost, total visibility, improved reporting and planning, and improved efficiency. On the other hand, disadvantages of ERP include the cost of the ERP software, the cost of implementation and maintenance, and the customization process. The general types of ERP include custom ERP and the off-the-shelf ERP.

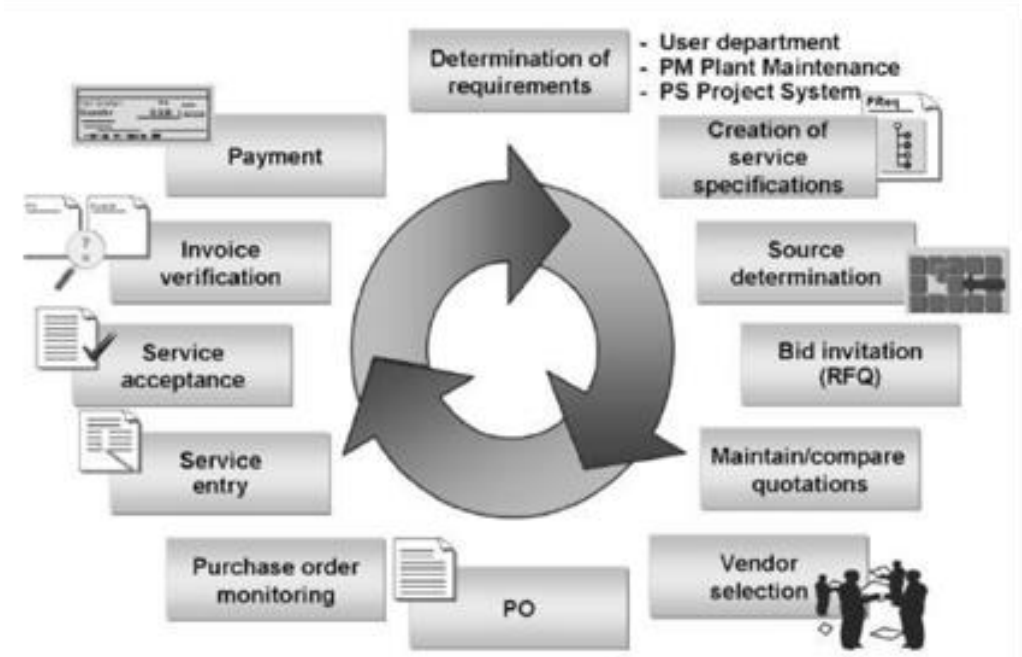


Figure 1-5: Example of ERP Contents
Image Source: https://www.manifera.com/nl/custom-enterprise-resource-planning-erp-system/

FUNCTIONAL SILOS

A **silos** is an airtight pit or tower for preserving foodstuffs, according to Webster's definition. Silos are simply segregated functioning units that are cut off from the rest of the environment. In a firm, functional silos are groups of personnel organized by function that operate independently of one another and without cross-collaboration. (Monday.com, nd). Classification of functional silos include horizontal silos and vertical silos.

HORIZONTAL SILOS

Henry Fayol, a management philosopher, was the first to separate functionalized organizations into five fundamental areas: planning, organizing, coordinating, commanding, and controlling in the early 1900s. Luther Gulick expanded and theorized Fayol's categorization in the 1930s, resulting in the POSDCORB functional model (**p**lanning, **o**rganizing, **s**taffing, **d**irecting, **c**oordinating, **r**eporting, and **b**udgeting). Starting in the late 1930s, the POSDCORB category became increasingly popular, resulting in a set of formal organization roles such as control, management, supervision, and administration. The language of organizational functions evolved over the following 50 years, for example, from planning to management to strategy, but the principle of organizing complicated tasks into structured functions persisted for control and coordination purposes.

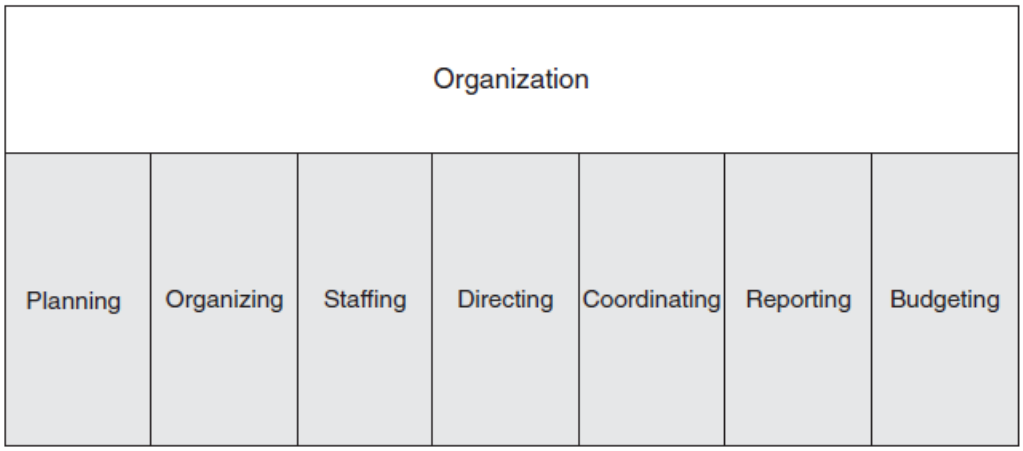


Figure 1-6: Horizontal Silo
Image Source: Adapted from Bernard, C. (1938). The Functions of the Executive. Cambridge: Harvard University Press.

VERTICAL SILOS

In the late 1960s, Harvard University's Robert Anthony discovered that businesses split responsibilities in hierarchical stages, from strategic planning through managerial control and operation control. Most institutions, for instance, have upper executives such as CEOs and presidents who manage long-term strategic plan, so although middle level leadership (e.g., vice presidents or general managers) concentrates on strategic concerns and the implementation of company's strategy to ensure that the company meets its organizational plans. The job of middle positions (for example, supervisors) is to concentrate on the business's daily operations functions. Although not independent organizational roles, this vertical category does entail a different set of activities.

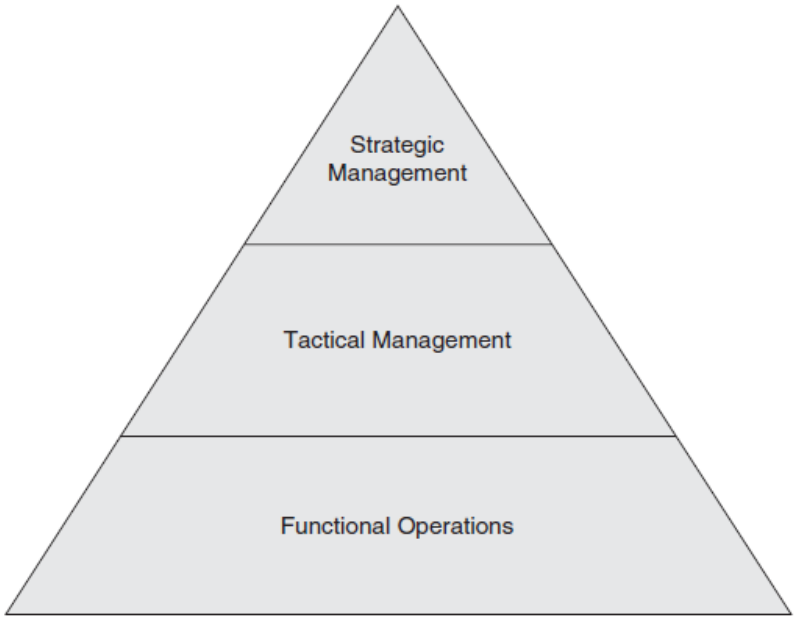


Figure 1-7: Vertical Silos
Image Source: Motiwalla & Thompson, 2012

As businesses get larger and more complicated, they prefer to split operations down into manageable units and give responsibility for these tasks to employees, allowing them to manage complexity while focusing on tasks that boost productivity and efficiency.

INFORMATION SYSTEMS (IS) IN ORGANIZATIONS

Today's successful organizations rely heavily on information systems. Information systems (IS) are important because they process data from corporate inputs to provide information that can be used to manage business operations. Some of the applications of information systems in organizations are the following: Business Communications System, Business Operations Management, Company Decision-Making, and Company Recording-Keeping (Markgaf, 2019).

In the core and secondary operations of an organization's value chain, information systems play a critical role. The development of IS implies that its primary purpose has been to serve the organization's changing information demands. Information Systems assist company processes including such accounting, finance, marketing, customer service, human resource management, operations, and manufacturing by providing a high level of computer automation (Supporting horizontal silos). Management is divided into three levels: strategic, middle, and operational, with information systems providing analytical and decision-making assistance (Supporting vertical silos) (Motiwalla & Thompson, 2012). Each company function and management level have its own set of requirements.

FUNCTIONAL SILOS AND INFORMATION SYSTEMS IN AN ORGANIZATION

Each functional area has its own set of information and reporting demands. There are numerous layers of management in each functional area of a company, which needing varying levels of analysis and knowledge depth. Organizations established diverse information systems to support each key operation and duty in order to boost efficiency and production.

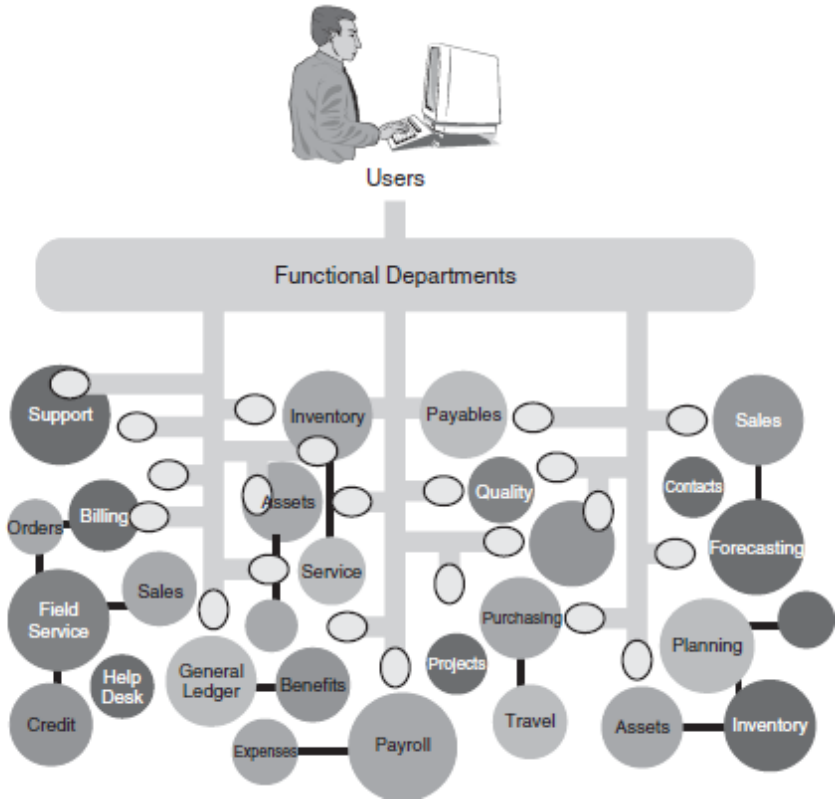


Figure 1-8: Functional Silos
Image Source: Motiwalla & Thompson, 2012

Each management level has different information requirements.

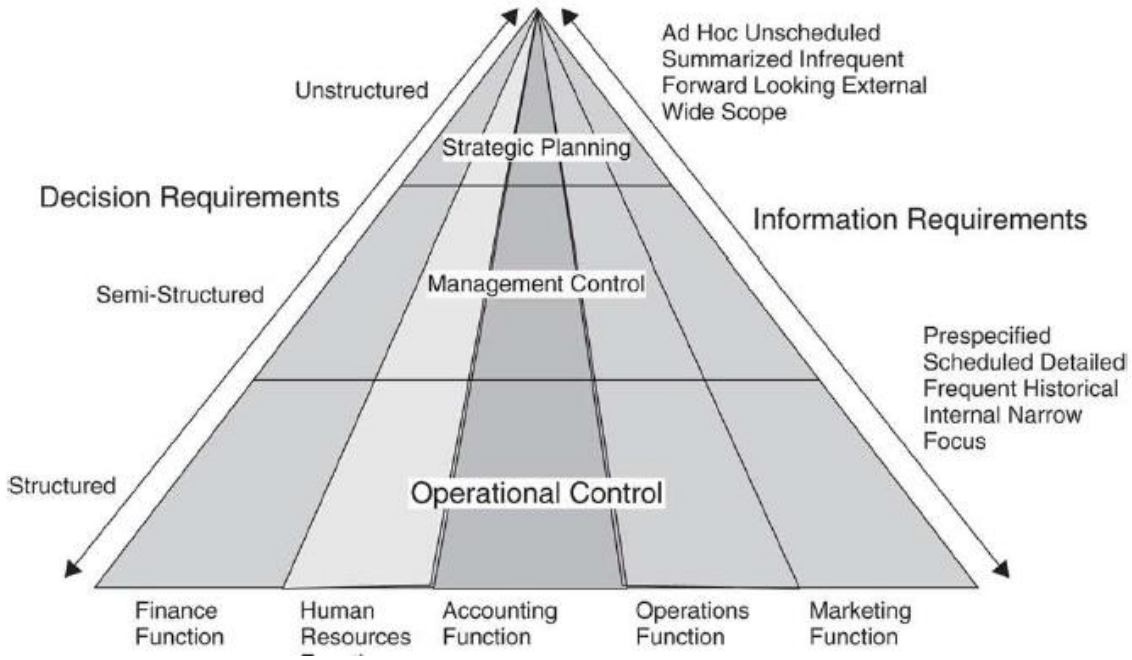


Figure 1-9: Management levels and information requirements
Image Source: Motiwalla & Thompson, 2012

INFORMATION SILOS AND SYSTEMS INTEGRATION

Over time, corporations developed a jumble of unconnected, non-integrated systems, which resulted in bottlenecks and slowed production. Organizations must be nimble and adaptable, and their information systems must include data, applications, and resources from several departments. A data silo system is inefficient, incorrect, and costly. Everyone has bottlenecks as a result of the system, and information is not available in real-time. Organizations must be customer-centric in order to compete effectively. This necessitates cross-functional collaboration between the company's accounting, marketing, and other divisions. People and resources from diverse functional areas can collaborate and share information at a certain level of the organization through cross-functional integration. By allowing information to flow freely from one unit to another, the cross-functional organizational structure breaks down functional silos.

IMPLICATIONS OF SYSTEM INTEGRATION FOR MANAGEMENT

Many new ethical dilemmas arise as a result of system integration. There's a chance that some workers will use information for personal gain or get unauthorized access to it. Implementing rules on ethical information usage, installing suitable security software and hardware (such as firewalls), and allocating resources for information access training and education are all possible remedies.

INTEGRATED SYTEMS – ENTERPRISE RESOURCE PLANNING (ERP)

ERP's purpose is to bring together all of an organization's departments and operations into a single infrastructure that shares a common database and meets the demands of each department. ERP systems are designed to replace a variety of systems that are often seen in businesses. Furthermore, ERP handles the difficult challenge of integrating data from many sources and making it available in real-time.

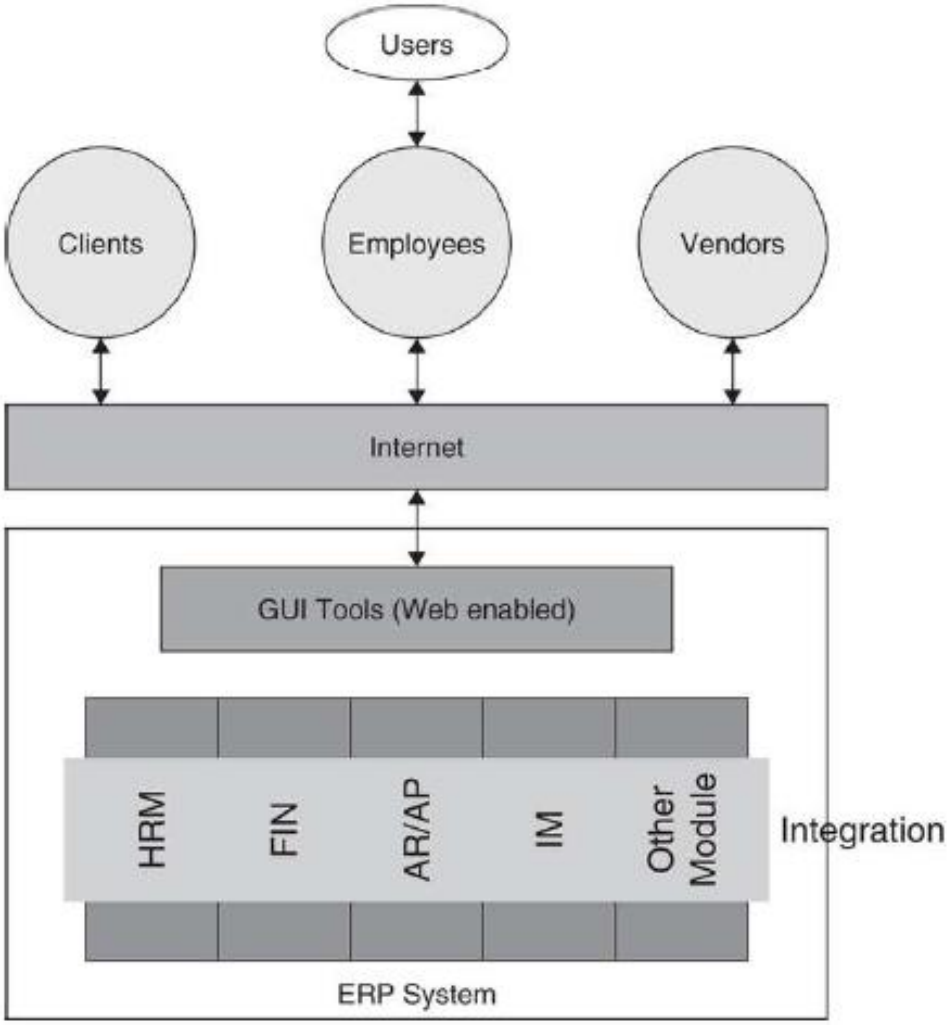


Figure 1-10: Integrated Systems - ERP
Image Source: Motiwalla & Thompson, 2012

ERP AND SYSTEMS INTEGRATION

ERP systems are multi-module, integrated software packages that service and support a variety of business operations across a company. They are generally commercial software packages that allow for the collection and integration of data from multiple departments within a company. This system enables the company to standardize and optimize its business operations in order to apply industry best practices. ERP systems are the first phase of enterprise applications that were needed to implement information that supports all of an organization's primary operations. ERP systems combine the organization's many functional areas and also the systems of its partners and suppliers.

An ERP system's purpose is to make flow of information more dynamic and instantaneous, improving its utility and value.

Figure 1-11: ERP and example systems
Image Source: https://www.digitalcorn.com/erp-application/

ERP's ROLE IN LOGICAL INTEGRATION

ERP solutions force companies to concentrate on business processes rather than functions. ERP systems have procedures built in for a wide range of common company operations. In terms of processing a client order, an ERP system applies best practices via particular built-in stages for: (1) order input, (2) routing via departments, and (3) transmission of output to various stakeholders.

ERPs ROLE IN PHYSICAL INTEGRATION

An enterprise may need to upgrade or install middleware, as well as get rid of their old system's hardware and software, before deploying the ERP system. Data integration, client integration, and application integration are all essential components of integration. With superior business procedures that focus on company goals rather than particular organizational objectives, an effective ERP installation enhances efficiency and productivity. Increased productivity through a seamless workflow and a business-to-business (B2B) transaction ecosystem with collaborators.

EVOLUTION OF ERP

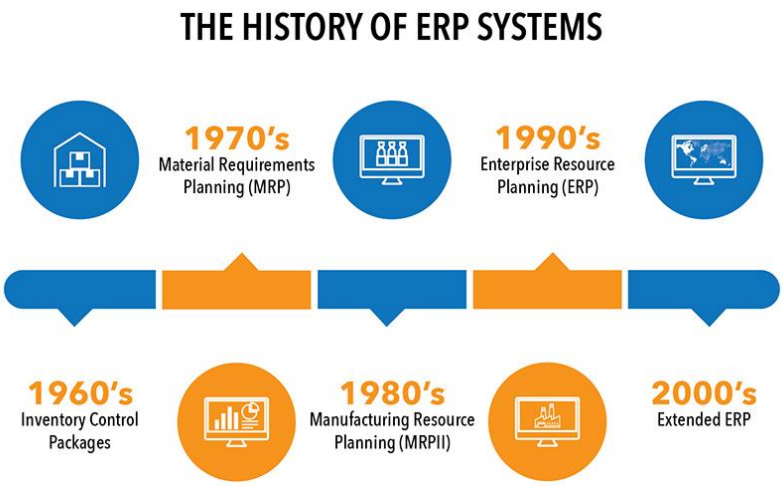


Figure 1-12: History of ERP
Image Source: https://www.omniaccounts.co.za/articles/history-of-enterprise-resource-planning

Timeline	System	Platform
1960s	Inventory Management & Control	Mainframe legacy systems using third generation software- (Cobol, Fortran)
1670s	Materials Requirements Planning (MRP)	Mainframe legacy systems using third generation software- (Cobol, Fortran)
1980s	Material Requirements Planning (MRP-II)	Mainframe legacy systems using fourth generation database software and manufacturing applications.
1990s	Enterprise Resource Planning	Enterprise Resource Planning
2000s	Extended ERP or ERP-II	Client-server systems using Web platform, open source with integration to fifth generation applications like SCM, CRM, SFA.

E-BUSINESS VS. ERP

E-Business	ERP
Focuses on linking a business with its external partners and stakeholders	Focuses on integrating the internal functional silos of the organization into an enterprise application
Disruptive technology—Totally transformed the way a business operates in terms of buying and selling, customer service, and relationships with suppliers	Adaptive technology—Merged the early data processing and integration efforts within an organization

ERP SYSTEMS COMPONENTS

An ERP System is composed of Hardware (servers and peripherals), Software Process (operating systems and database), Information (organizational data from internal and external sources), Process (business processes, procedures, and policies), and people (end-users and IT Staff).

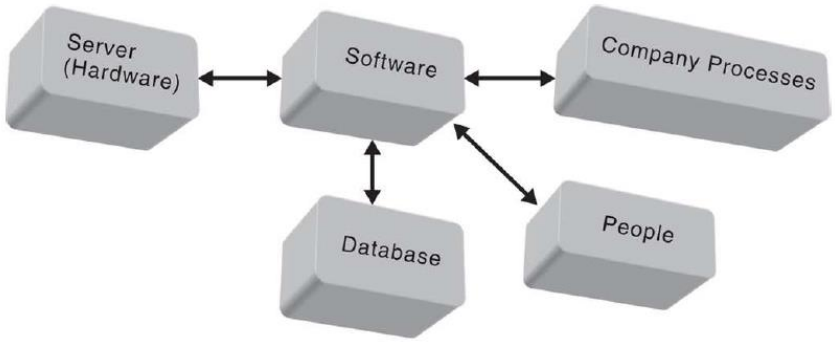


Figure 1-13: ERP components
Image Source: Motiwalla & Thompson, 2012

ERP ARCHITECTURE

The budget, operation, and the use of an ERP system are all influenced by the system's architecture. The ERP architecture aids the deployment team in developing the institution's ERP system. When ERP is acquired, the vendor is frequently in-charge of the architecture (Package-Driven Architecture). There are two types of architectures.

(1.) Logical focuses on the supporting needs of the end-users.
(2.) Physical focuses on the efficiency of the system.

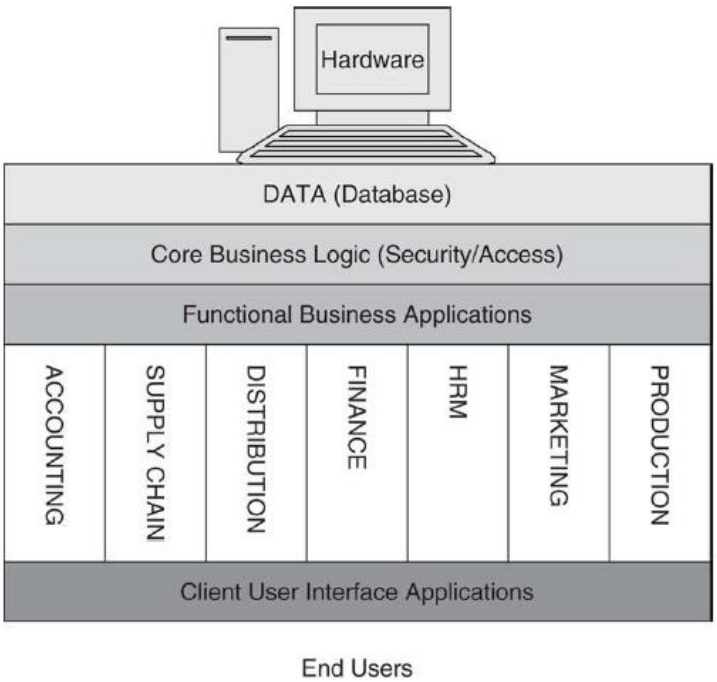


Figure 1-14: ERP Architecture
Image Source: Motiwalla & Thompson, 2012

TIERED ARCHITECTURE EXAMPLE OF ERP SYSTEM

1. Presentation Logic Tier
2. Business Logic Tier
3. Data Tier

SYSTEM BENEFITS OF ERP

1. Cross-functional integration of data and applications (i.e., data can be entered once and used by all applications; thus, improving accuracy and quality of the data). Improvements in maintenance and support as IT staff is centralized.
2. User interface coherence across programs means less staff training, more productivity, and cross-functional job mobility.
3. Better controls and centralization of hardware improve data and application security.

SYSTEM LIMITATIONS OF ERP

1. The complexity of installing, configuring, and maintaining a system grows, necessitating the use of expert IT personnel, hardware, and network infrastructure.
2. IT hardware, software, and human resource consolidation can be time-consuming and difficult to achieve.
3. Moving data from an old system to a new one can be a time-consuming and difficult procedure.
4. IT employees and end-users of the new system may need to be retrained, which may cause resistance and lower productivity.

BUSINESS BENEFITS OF ERP

1. Organizational agility in terms of adapting to changes in the environment in order to gain and sustain market share
2. Employee cooperation is aided by information sharing across functional domains.
3. Improving efficiency by linking and sharing data in real time with supply-chain partners leads to cheaper costs.
4. Improved customer service as a result of faster information flow across departments
5. The re-engineering of business processes improves the efficiency of business operations.

BUSINESS LIMITATIONS OF ERP

1. It might be costly and time consuming to retrain all personnel on the new system.
2. Upheaval and opposition to the new system might result from changes in business responsibilities and department boundaries.

GUIDELINES ON IMPLEMENTING ERP

1. A company must plan and understand the life cycle of ERP systems before installing it.
2. The key to a successful deployment is to follow a tried-and-true methodology, take things slowly, and start with a grasp of the ERP life cycle.

3. ERP system installations are extremely dangerous, and employing a well-defined project plan with a tried-and-true methodology can help mitigate those risks.
4. To make the switch from existing information systems/applications to an ERP system, there must be a compelling and well-communicated requirement.

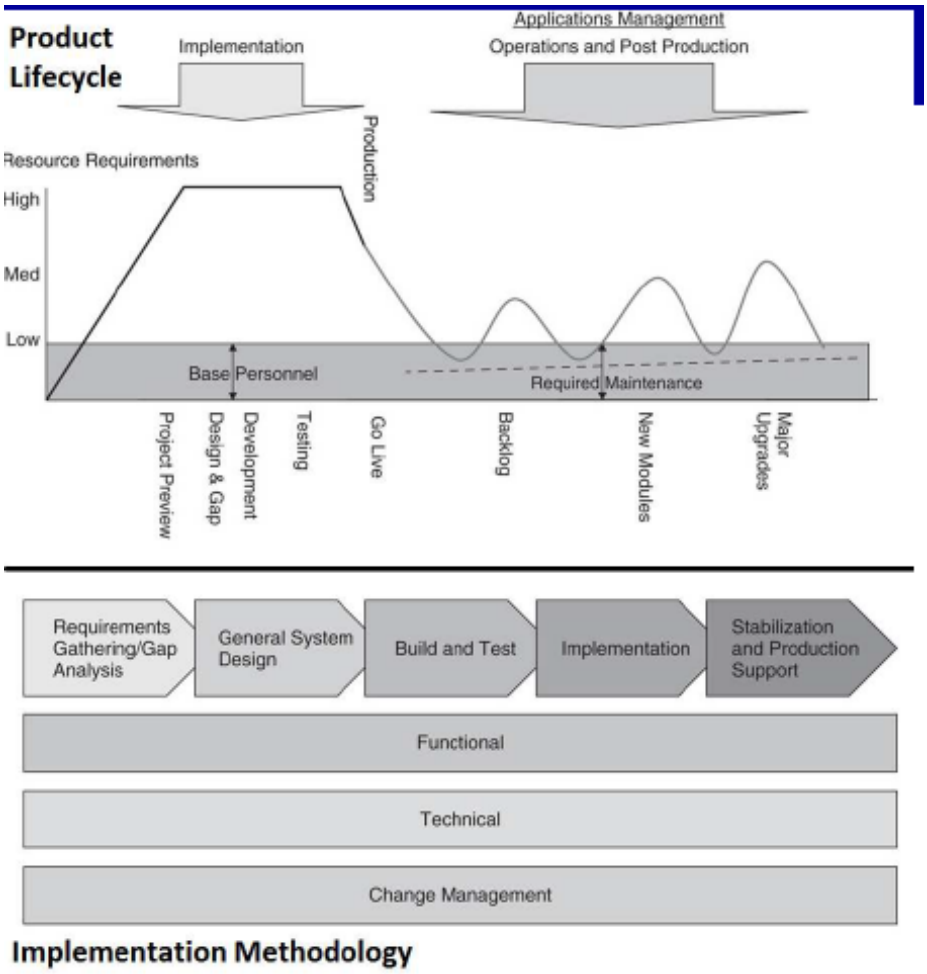


Figure 1-15: Guidelines when implementing ERP

SOFTWARE AND VENDOR SELECTION

It is preferable for a company that does not have experience designing ERP systems to buy one off the shelf. Before deciding on a provider, the company must assess its existing and future enterprise management system requirements. Examine the organization's existing hardware, network, and software infrastructure, as well as the implementation resources available.

Vendor Evaluation Criteria

1. Business functions or modules supported by their software
2. Features and integration capabilities of the software
3. Financial viability of the vendor as well as length of time they have been in business
4. Licensing and upgrade policies
5. Customer service and help desk support

6. Total cost of ownership
7. IT infrastructure requirements
8. Third-party software integration
9. Legacy systems support and integration
10. Consulting and training services
11. Future goals and plans for the short and long term

Operation and Post-Implementation

1. One of the most significant milestones in a project's success is when it goes live (or "go-live").
2. It is critical that all project team members coordinate their efforts to ensure that all tasks and activities are accomplished prior to going live.

Five areas of stabilization are important

1. Training for end-users
2. Reactive support (i.e., help desk for troubleshooting)
3. Auditing support to make sure data quality is not compromised by new system
4. Data fix to resolve data migration and errors revealed by audits
5. New features and functionalities to support the evolving needs of the organization

ERP VENDORS

1. **SAP**

   SAP is the world's most popular ERP software, with over 12 million users worldwide. Its products are suitable for a wide range of industries and markets.
2. **Oracle/Peoplesoft**

   Oracle, the second largest ERP provider, offers solutions categorized by industry and pledges long-term support for PeopleSoft users (acquired in 2004)
3. **Microsoft Dynamics**

   Microsoft Dynamics, formerly known as Microsoft Business Solutions or Great Plains, is a complete business management system based on the Microsoft platform.
4. **Infor**

   The world's third largest corporate software vendor. It provides integrated supply chain, customer relationship, and supplier management solutions.
5. **Lawson**

   Enterprise performance management, distribution, financials, human resources, procurement, and retail operations are all examples of industry-specific software solutions.

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson in different platforms like Google Forms, Schoology, and the like.

LESSON 2
TYPES OF SYSTEM INTEGRATION

INTRODUCTION

At the information and service levels, system integration connects information systems. Information sharing is facilitated via system integration. It allows you to do business in real-time. System integration encompasses both technical and strategic values. Electronic marketplaces, supply chain enablement, web visibility, and customer relationship management (CRM) are activities that require integrated solutions. Application integration's success and usefulness are determined by how well the issue domain is understood, the type of architecture used, and the technology used.

In this lesson, students will gain understanding on the different types of system integration and the related activities and concepts.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to

1. Identify the different approaches to systems integration; and
2. Understand how information-oriented, business process integration-oriented, service-oriented, and portal-oriented approach of integration differs

DISCUSSION

SYSTEM INTEGRATION APPROACHES

Different ways can be used to integrate software applications. It includes:

a. Information-oriented
b. Business process integration-oriented
c. Service-oriented
d. Portal-oriented

A. INFORMATION-ORIENTED

The convergence of two or more systems by permitting easy data flows across applications is known as **information-oriented**, a system integration approach. Some of its sample activities are: connecting databases, dealing with basic data transfers between two or more applications, and migrating data from source database to destination database. While this kind of approach allows seamless transfer of data, all connected systems must be thoroughly understood by designers. This is a disadvantage of an information-oriented system integration approach.

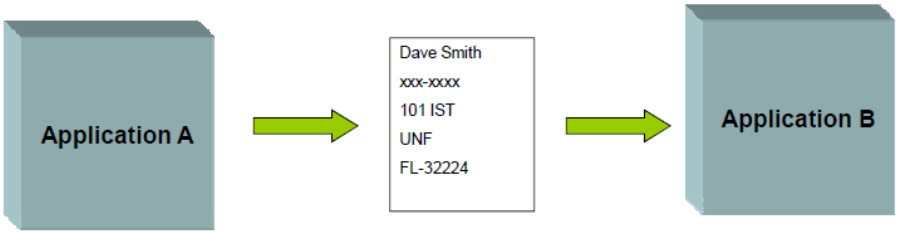


Figure 2-1: Information-Centered Approach
Image Source: Next Generation Application Integration Textbook

Moving data across systems, for example, may necessitate changes to both the content and the schema on the fly.

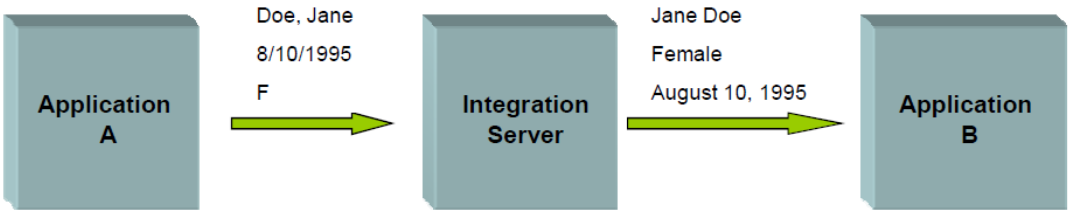


Figure 2-2: Flow of data across systems
Image Source: Next Generation Application Integration Textbook

INFORMATION-ORIENTED INTEGRATION CONCEPTS

1. Coupling

Coupling connects applications in such a way that they are reliant on one another, exchanging methods, interfaces, and sometimes data. Coupling necessitates significant application modifications. Modifications in the source or target system need changes in connected systems as well. Reusability is enabled via coupling, allowing the reuse of basic business procedures.

2. Cohesion

The term “cohesion” refers to the “act or state of sticking together” or “logical agreement.” Applications and databases are distinct of one another in cohesiveness. Modifications to the origin or target system should not have a significant effect on others. Integration benefits from cohesion since it allows for more flexibility. It enables the installation, modification, and removal of systems without impacting the overall system.

INFORMATION PRODUCERS AND CONSUMERS

The entities that create and consume information are known as source and target systems. Database (integration using SQL, JDBC), Application (API, adapters), User interface (screen scraping), and Embedded Devices (temperature sensors, call-counting machines) are examples of systems that create and consume data. Because they are designed to create and consume data, these systems are considered “point of integration”.

STEPS TO APPROACH INFORMATION INTEGRATION

1. Identify the data
2. Catalog the data
3. Build the enterprise metadata model – this model will be used as master guide for integrating the various information stores that exists within the enterprise.

In order to successfully implement an integrated solution, the company must first establish how information flows through it and how it does business. Data Replication, Data Federation, and Interface Processing are some of the several ways to connect.

1. Data Replication

The process of shifting data across two or more databases is known as data replication. By deploying software between databases, data replication may be performed. Data is extracted from the source database and placed in the target database by software. Data replication has the advantages of being low-cost and simple to implement. Data replication, on the other hand, is not suited for integrating functionalities in applications. That is, if data is linked to methods or if data is shared with methods. It also necessitates modifications to the source and destination apps.

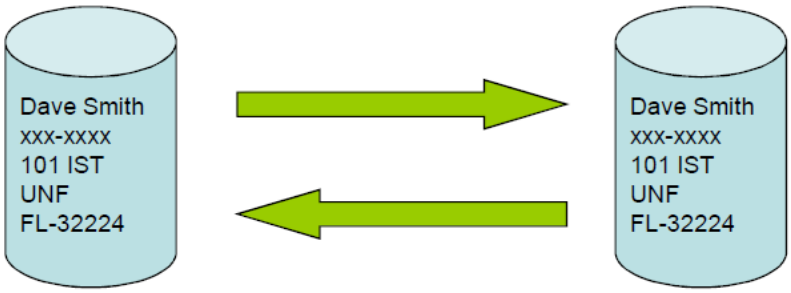


Figure 2-3: Data Replication
Image Source: Next Generation Application Integration Textbook

2. Data Federation

The practice of combining various databases into a single virtual database is known as data federation. The program makes use of virtual databases. The data gathering and dissemination to the physical database is handled by integration software. Data federation has the benefit of being able to combine many types of databases. In data federation, however, the interface between the application and the database must be altered.

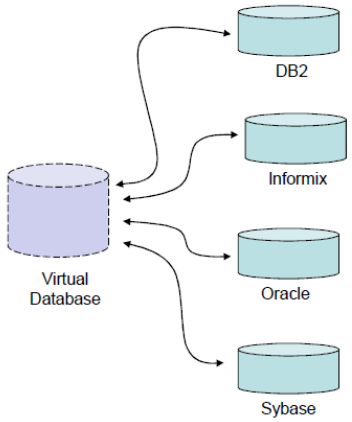


Figure 2-4: Data Federation
Image Source: Next Generation Application Integration Textbook

3. Interface Processing

Integrating packaged and bespoke programs is part of interface processing. Enterprise Resource Planning is one example (ERP). This is the most often used method of integration. The benefits include easy connection with commercially available applications and the use of screen scrapers as connection points. When converting information to transfer between systems, API solutions accommodate for variations in schema, content, and application semantics. However, there is a disadvantage in terms of business logic.

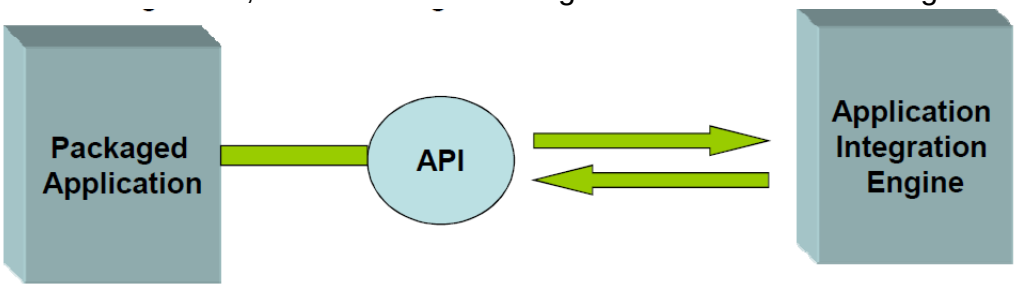


Figure 2-5: Interface Processing
Image Source: Next Generation Application Integration Textbook

B. BUSINESS PROCESS INTEGRATION-ORIENTED

The purpose of business process integration is to enable integration not just via the sharing of information, but also via the management of that information using simple tools. It is concerned with coordinating or controlling the flow of data between source and target applications. It also emphasizes process logic while keeping application logic separate. It is described as the application of relevant rules in a logical sequence to transmit data across participating systems and to visualize and exchange application-level operations. It's the capacity to build a common business process model that takes into account the sequence, hierarchy, events, execution logic, and data flow between systems.

Process-to-process solutions are created by connecting separate processes. It automates human-assisted tasks. An advantage is that it supports information and control

logic flow, as well as automating duties formerly handled by people. On the other hand, it has the drawback of focusing solely on process flow and integration. It doesn't focus on user interface, updating databases, or executing a transaction.

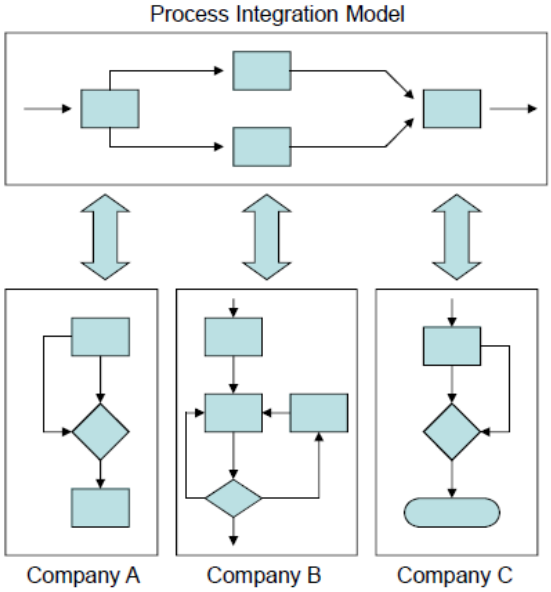


Figure 2-6: Business Process Integration Oriented
Image Source: Next Generation Application Integration Textbook

APPLICATION INTEGRATION

It's the capacity to build a common business process model that takes into account the sequence, hierarchy, events, execution logic, and data flow between systems. The goal is to create a single logical model that spans several applications and data stores by introducing the concept of a common business process that governs how computers and humans interact to meet a specific business need. Support for information and control logic flow, as well as the automation of human functions are some of the advantages of this model. Focusing solely on process flow and integration procedures is a disadvantage.

Objective

Provides a control method that describes and implements information flow and the invocation of processes across multiple systems.

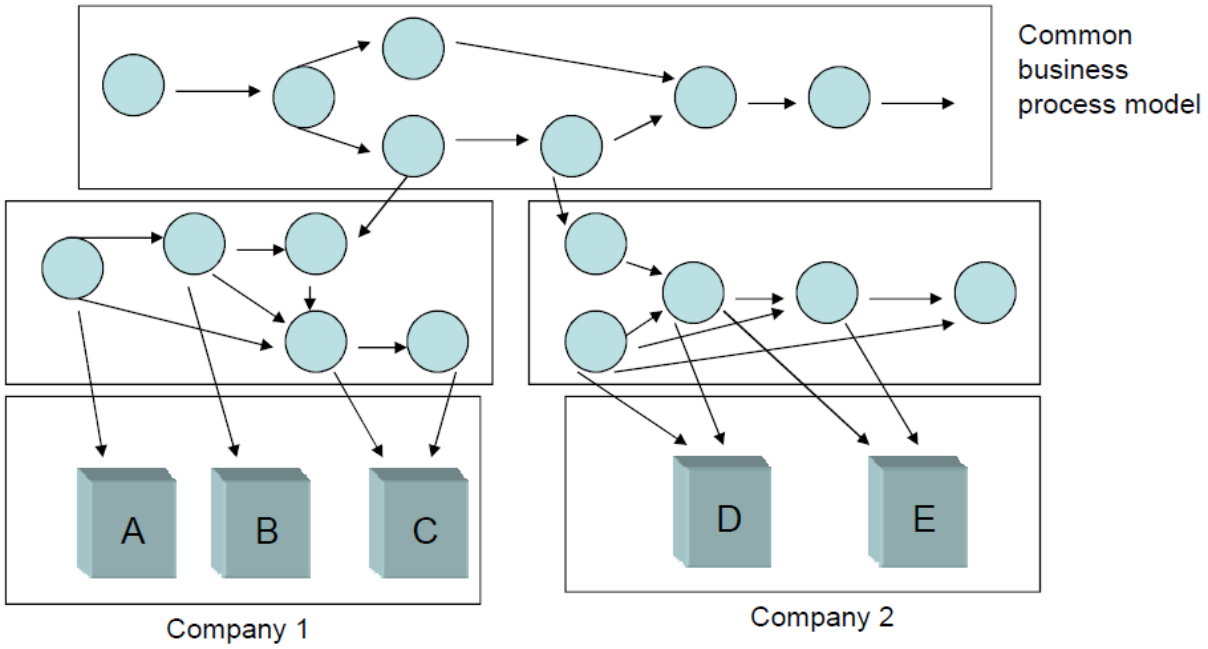


Figure 2-7: Common business process models
Image Source: Next Generation Application Integration Textbook

Technology Components

1. Graphic modeling tool – This is where business model is created and defined.
2. Business process engine – It controls the execution of the multi-steps business processes and maintains state and the interactions with the middleware
3. Business process monitoring interface – Allows end-users to monitor and control execution of a business process in real time and optimize where needed
4. Business process engine interface – Allows other applications to access the business process engine
5. Integration technology (middleware) – Connects the source and target system

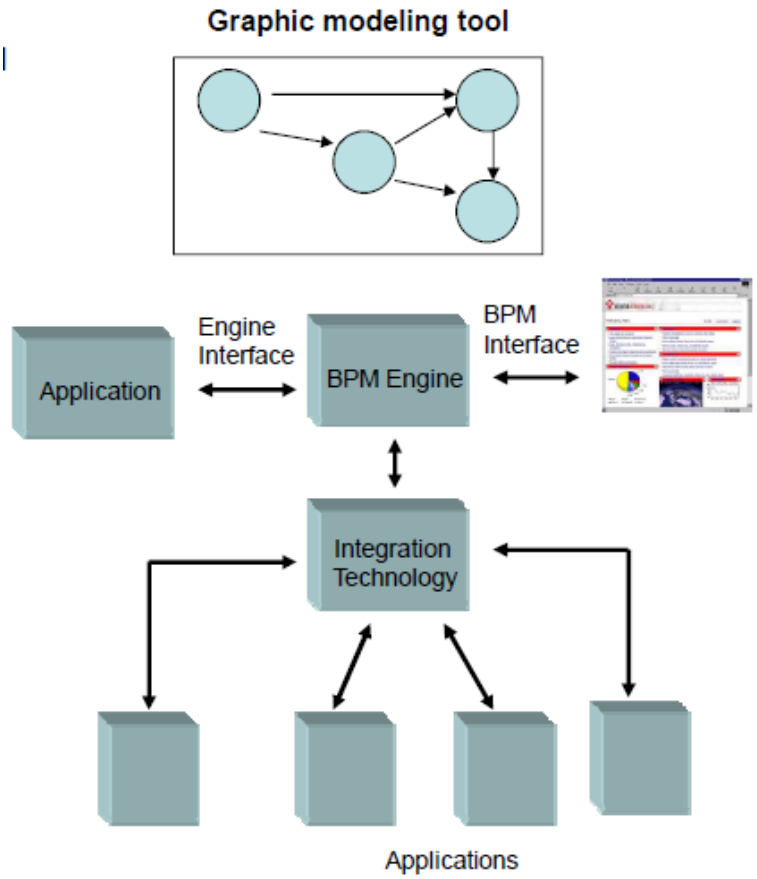


Figure 2-8: Technology Components
Image Source: Next Generation Application Integration Textbook

Three Levels of Technology

1. Process modelling

   Information mobility is specified in process modeling. The common process model, real things such as corporations, organizations, or individuals, and the source and destination systems are all components of process models.

2. Transformation, routing, and rules

   Information movement and formatting occur as a result of transformation, routing, and rules. Routing makes it possible to retrieve important data from any source application, target application, or data repository.

3. Message service

   The messaging service is in-charge of transferring data across all linked apps.

C. SERVICE-ORIENTED APPLICATION INTEGRATION

At the service level, service-oriented application integration provides a framework for connecting applications. The idea is to use the Internet's capacity to provide well-defined interfaces and directory services for remote application services. The technology

to achieve the aforementioned aim is web services. The future of application integration is Web services.

Service-Oriented

Service-oriented integration allows apps to share business logic and procedures, allowing them to work together. One example of this is the web services. Application reusability is enabled via service-oriented architecture. However, it is necessary to update the application logic, and the implementation cost is substantial.

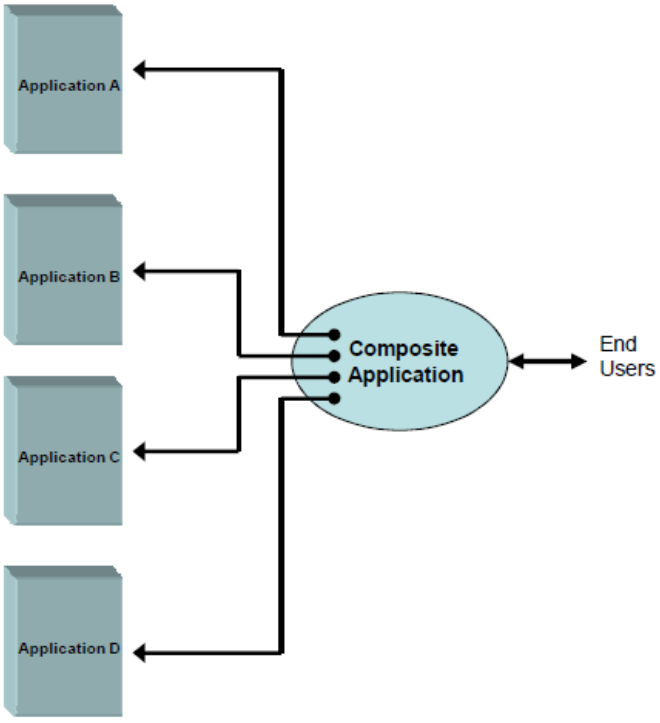


Figure 2-9: Service-Oriented Approach
Image Source: Next Generation Application Integration Textbook

The Basics of Service-Oriented Application Integration

Enterprises can share common application services and information with the help of service-oriented application integration. Web services (distributed objects) are the center of infrastructure. The use of a standard set of application services across corporate applications encourages reusability. This decreases the requirement for duplicated application services and/or applications dramatically.

Application Service

Sub-routines or procedures in applications are known as application services. It must be called in order for something to happen in the application. Remote services that create or consume data are known as **application services**. Composite applications made up of local and remote application services are created by combining application services.

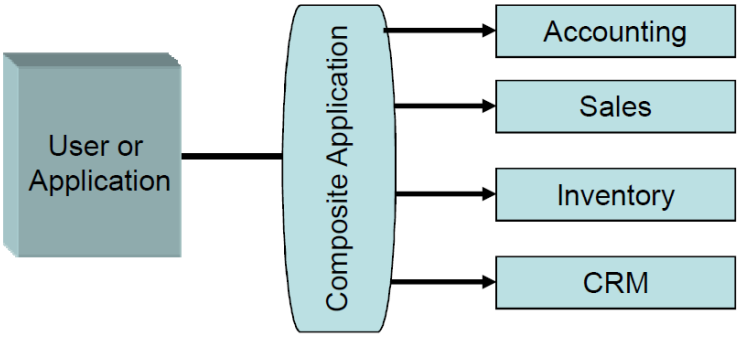


Figure 2-10: Application Service
Image Source: Next Generation Application Integration Textbook

When to Leverage Service-Oriented Integration

The simple binding of two or more applications in order to integrate both business processes and data is a potential benefit of service-oriented integration. The following conditions are some of the key considerations when to leverage service-oriented integration.

1. When two or more firms need to share similar program logic, such as calculating shipping prices from a common supplier that varies often.
2. When two or more firms wish to split the expenses and benefits of developing a shared application.
3. When the issue domain is small and specialized, and when all organizations can work on a common application

Solutions Architecture

1. **Event-Drive** - Refers to designs that focus on data mobility rather than application service aggregation. Data is sent from one system to another in support of a specific business transaction, but application services must also be accessed.
2. **Composite-Application** - Describes architectures that require application services to be aggregated into a single application instance.
3. **Autonomous-distributed** - Refers to web service architectures that have been so closely interwoven that they seem to be one application. Binding inter- and intra-company apps into a single, coherent entity.

D. PORTAL-ORIENTED APPLICATION INTEGRATION

The ability to access a variety of systems (both internal and external corporate systems) through a single user interface or application is provided by portal-oriented application integration. This group is the most likely to use a web browser. It completely eliminates back-end integration.

Steps to create portal

1. Create a portal application that includes the user interface as well as the program's behavior.

- From the user interface to the back-end systems, the portal program must be able to govern user interaction, capture and handle errors, and control the transaction. The interface development environment (IDE) for creating the user interface, defining application behavior, and connecting to the back-end is provided by application servers.

2. Specify which data from the back-end systems should be exchanged with the portal application.

Portal-Oriented

Portal-Oriented integration combines programs through a single user interface or application, which is often accessed via a web browser. It consolidates data from several apps into a single application. No back-end connectivity and ease of usage are some of its advantages (browser user interface). On the other hand, there is no such thing as real-time integration on this type of integration.

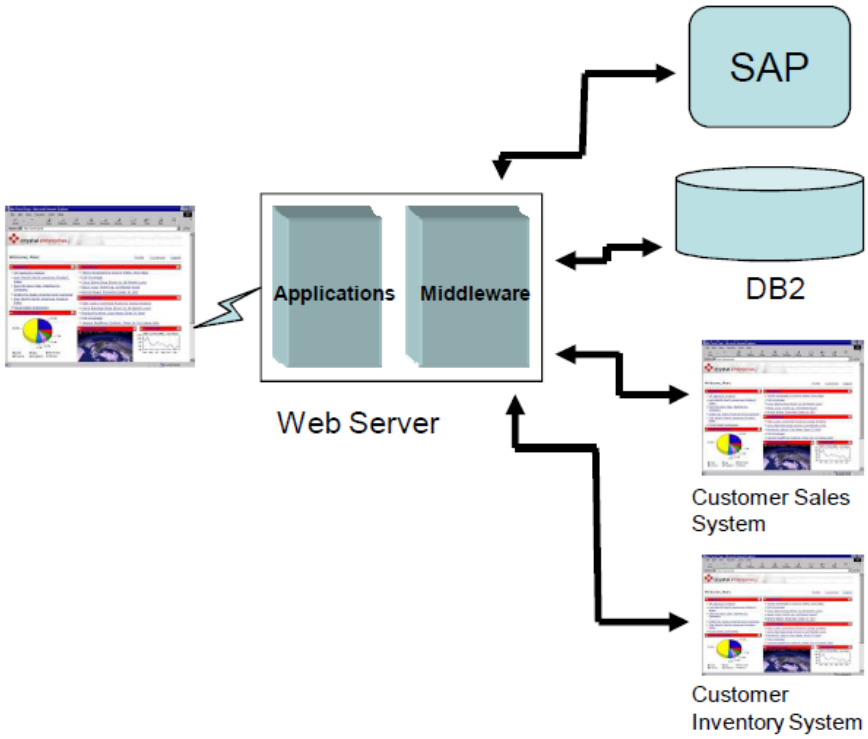


Figure 2-11: Portal Oriented Approach
Image Source: Next Generation Application Integration Textbook

The Power of Portals

The primary benefit of using portals is that it eliminates the need to directly interconnect back-end technologies across organizations or inside corporations. It eliminates the costs and hazards that come with it. Using portals is a non-intrusive method of allowing other organizations to connect with a company's internal processes through a secure online interface. Other integration techniques take longer to implement than portals. However, there are certain disadvantages such as: Information does not flow in

real time, necessitating human contact; information must be abstracted through another application logic layer (e.g., application servers); and security is a major problem when extending company data to consumers over the internet.

Portal Categories

1. **Single-System Portals** - Single-system portals are businesses that have extended their user interfaces to the web. Application servers, page servers, and technologies for converting basic displays to HTML are all options.
2. **Multiple-Enterprise-System Portals** - Multiple Enterprise System Portals expands the design of a single-system gateway to multiple enterprise systems. Application server architecture is used. Users may pull data from these systems and update it using a single web browser interface that may be accessible via an extranet or the internet.
3. **Enterprise Portals** - Enterprise Portals broaden the scope of a multi-enterprise system portal to include systems from multiple companies. Application servers are a good choice for businesses because they funnel data from connected back-end systems.

Components of Portal Architecture

1. **Web Clients** - PC or any other device that can display HTML and images and runs a web browser.
2. **Web Servers** - At their heart, web servers are file servers. They reply to web client queries and subsequently transmit the needed file. Web servers fulfill two functions: they provide file material to web clients and they run basic applications.
3. **Database Servers -** Database servers respond to requests and return information.
4. **Back-end Applications** - Back-end apps are corporate apps that reside within a single company or across many companies. Example, ERP.
5. **Application Servers** - Application servers provide middle layer between back-end applications, databases, and the web server. It communicates with both the web server and the resource server using transaction-oriented application development

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson in different platforms like Google Forms, Schoology, and the like.

LESSON 3
SYSTEMS INTEGRATION TECHNOLOGIES

INTRODUCTION

Middleware bridges the gap between applications, tools, and databases to provide a unified service and optimize solutions to every user. In this lesson, you will be introduced to different systems integration technologies to fully understand how system integration works taking middleware in consideration. This lesson will also inform you of the different integration models and types of middleware.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to

1. Identify what is a middleware, its models, and
2. Compare and contrast the different types of middleware for system integration

DISCUSSION

WHAT IS MIDDLEWARE?

Any form of software that allows communication between two or more software systems is known as **middleware**. It might be as basic as an application-to-application communication link or as complex as information exchange and logic execution methods. Middleware is a technology that enables us to transfer data across different businesses.

Two types of Middleware Models

1. **Logical Middleware Model** - The logical middleware model conceptualizes how information travels within an organization.
2. **Physical Middleware Model -** Both the actual mechanism of information flow and the technology used are depicted in the physical middleware model.

POINT-TO-POINT MIDDLEWARE

To connect one program to another, point-to-point middleware employs a simple pipe. It establishes a direct link between a source and a target application. The inability to link more than two apps, the lack of capacity to house application logic, and the inability to modify messages as they travel through the pipe are all drawbacks of point-to-point middleware.

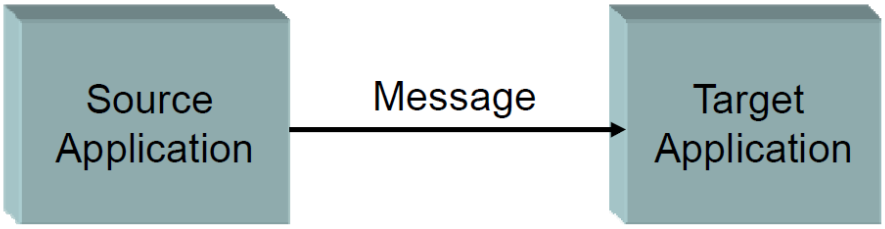


Figure 3-1: Point-to-Point Middleware
Image Source: Next Generation Application Integration Textbook

MANY-TO-MANY MIDDLEWARE

Many-to-many middleware connects a large number of applications. It can handle many sources or target apps. As a result of this capabilities, it is the ideal choice for application integration. It has a wide range of applicability and flexibility in the application integration sector.

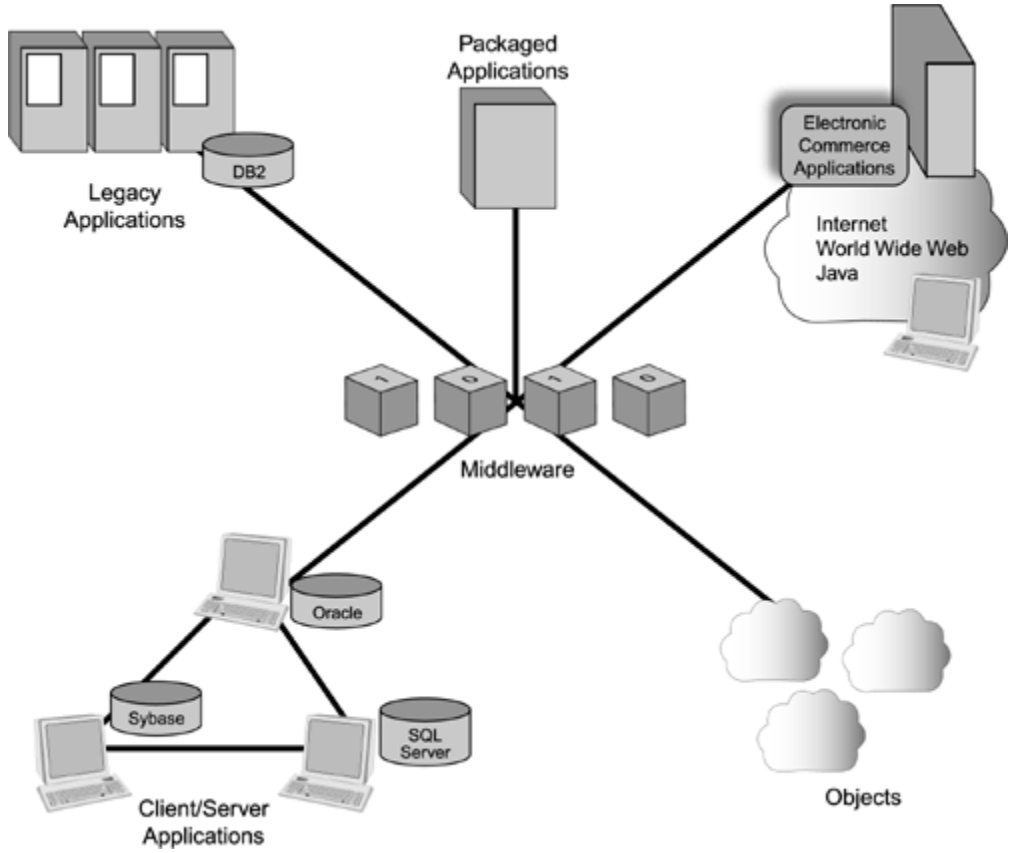


Figure 3-2: The Many-o-Many Middleware Model
Image Source: https://flylib.com/books/en/2.672.1.52/1/

ASYNCHRONOUS VS SYNCHRONOUS MIDDLEWARE

There are two types of communication mechanisms in which middleware employs: **asynchronous** and **synchronous**.

In an asynchronous mode, middleware transfers data between one or more applications. The middleware software is therefore separated from the source and destination applications. Apps in asynchronous middleware are not reliant on the

processing of other linked applications. Regardless of the condition of the other apps, the application can always continue processing.

Applications and synchronous middleware are inextricably linked. Applications rely on synchronous middleware to handle one or more function calls at a distant application. To wait for the remote application to reply, the calling application must pause processing.

Asynchronous application integration is recommended over synchronous application integration. When synchronous middleware encounters issues such as network or remote server issues, the application must halt processing. Furthermore, synchronous middleware consumes bandwidth since multiple network calls are required to enable a synchronous function call.

Connection-Oriented and Connectionless

Two parties connect, exchange messages, and ultimately disengage in connection-oriented communication. This is usually a synchronous procedure, although it can be asynchronous as well. The caller application does not establish a connection with the destination process in connectionless communication. The receiving application merely processes the request and, if necessary, responds.

Direct Communication

The middleware layer takes the message from the caller application and sends it straight to the remote application in direct communication. With synchronous processing, either direct or deferred communication is employed. Direct is often synchronous, whereas queued is typically asynchronous. The direct communication paradigm is used by the majority of RPC-enabled middleware.

Queued Communication

In most cases, queuing communication necessitates the use of a queue manager to place a message in a queue. The message is then retrieved by the remote application either immediately after it has been sent or at any point in the future (barring time-out restrictions). The information travels back through the queuing mechanism if the caller application demands a response (such as a verification message or data). Queued communication is used by the majority of Message-Oriented Middleware products.

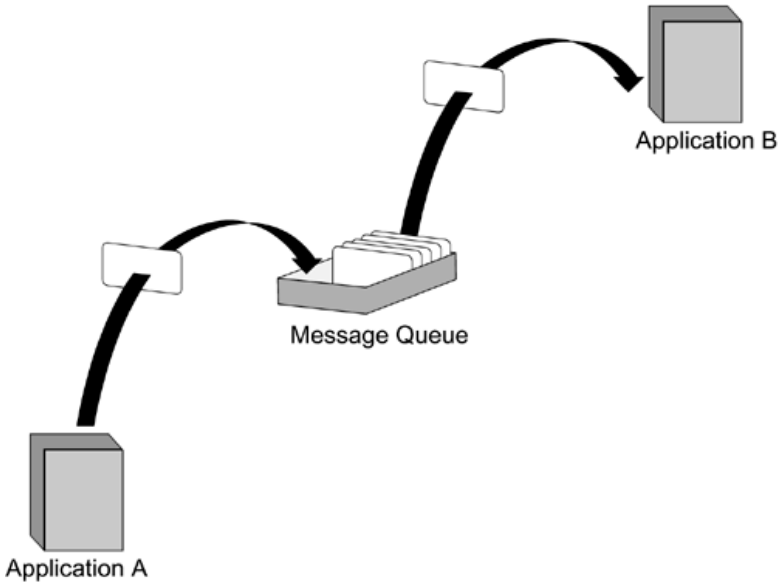


Figure 3-3: Queued Communication
Image Source: https://flylib.com/books/en/2.672.1.52/1/

The benefit of queuing communication over direct communication is that the distant program does not need to be running in order for the calling program to send it a message. Furthermore, queuing communication middleware generally does not prevent the calling or remote applications from completing their tasks.

Publish/Subscribe

Publish/subscribe (pub/sub) removes the requirement for an application to know anything about the destination application. All it needs to do is transmit the data it wants to share to a destination within the pub/sub engine, sometimes known as a **broker**. The information is subsequently redistributed by the broker to any apps that are interested. For instance, if a financial application wants to make all accounts receivable data available to other apps, it would notify the pub/sub engine. The engine would then let everyone know that this data was accessible, and any application could subscribe to that subject to get accounts receivable data.

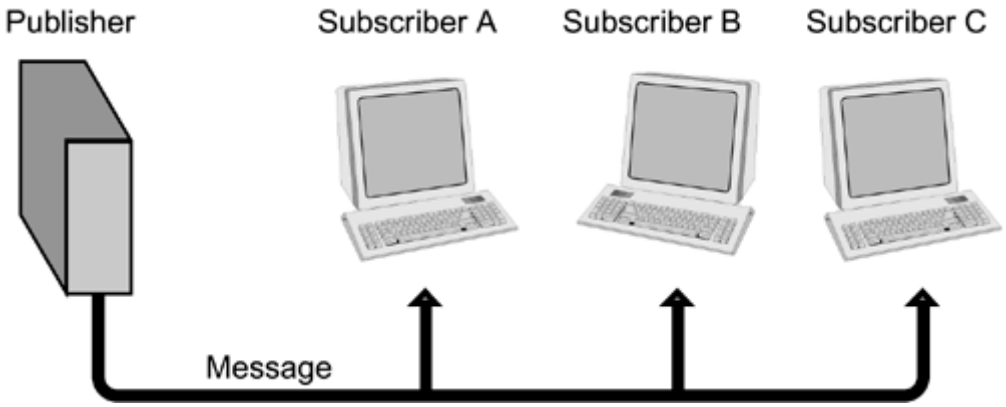


Figure 3-4: The publish and subscribe model
Image Source: https://flylib.com/books/en/2.672.1.52/1/

In Figure 3-4, the publisher is the information supplier. Publishers provide information on a topic, but they are not required to know anything about the applications that are interested in the data. The subscriber is the information's receiver or consumer. When the material is published, the publisher defines a topic. The subscriber chooses a subject that interests them. Only accounts receivable information is sent to the subscriber in this case.

Request Response

The request-response approach does exactly what it says on the tin. Request response middleware is used to send a request to an application, and the application responds. Any middleware that may support a response from a request between applications, such as integration servers or application servers, are examples of request and response middleware.

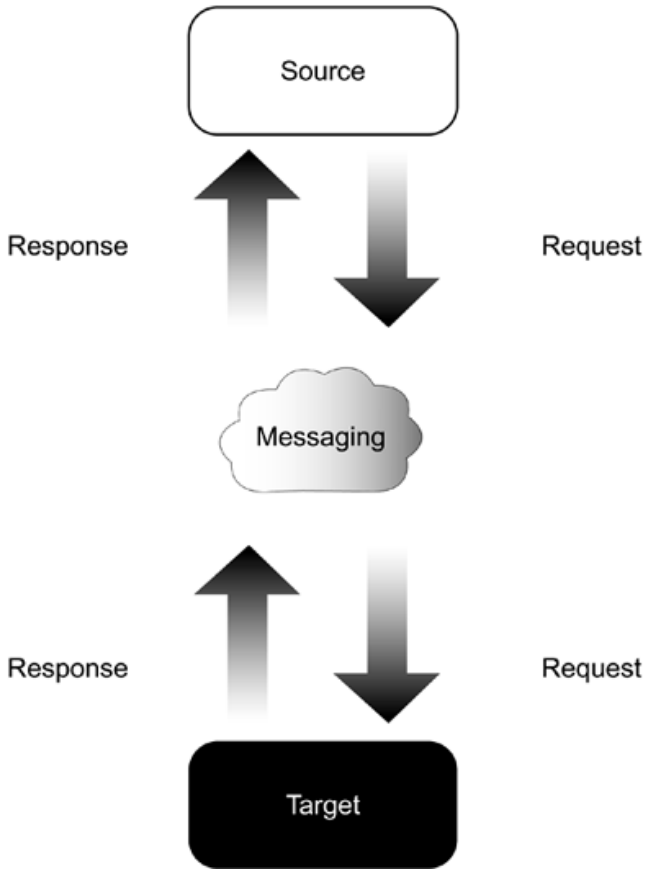


Figure 3-5: Request Response Model
Image Source: https://flylib.com/books/en/2.672.1.52/1/

Fire and Forget

The fire-and-forget approach enables a middleware user to "fire off" a message and then "forget" about it, without having to worry about who gets it or even if it is ever received. Another instance of an asynchronous method is this. The goal of fire and forget is to let a source or destination application sends out particular sorts of messages to a large number of people while avoiding auditing and response features. It also enables central servers to send messages out.

TYPES OF MIDDLEWARE

A. RPC
B. Message-oriented middleware (MOM)
C. Distributed objects
D. Database-oriented middleware
E. Transactional middleware
F. Integration servers

A. Remote Produce Calls (RPC)

The oldest kind of middleware is the Remote Produce Calls (RPC). It allows you to execute a function from one application and have it run in another application on a distant system. RPCs operate in a synchronous manner. They also need more bandwidth than other middleware kinds. The advantage of RPC is its ease of use in terms of mechanism and programming. Its disadvantages are the enormous performance cost and its difficulty to scale.

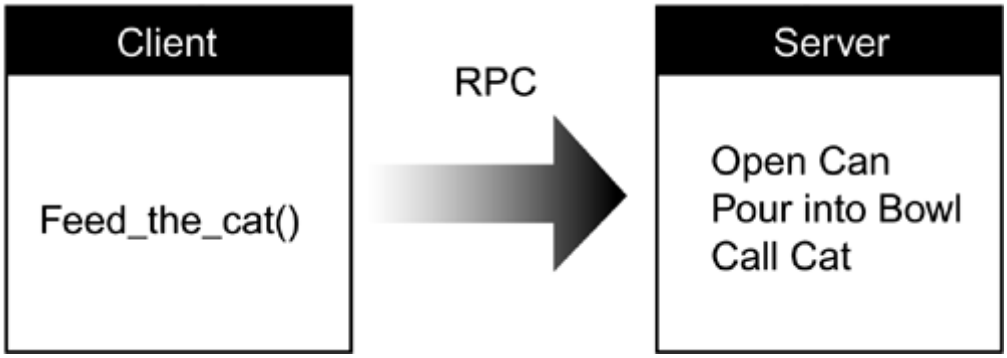


Figure 3-6: RPCs allow local function calls to execute on remote computers.
Image Source: https://flylib.com/books/en/2.672.1.53/1/

B. Message-Oriented Middleware (MOM)

MOM is queuing software that uses messages to transport data from one location to another. MOM communicates amongst apps via the concept of messages. It is not necessary to have a direct connection between the middleware mechanism and the application. The asynchronous paradigm is used by MOM. This enables the program to run on its own. Messages are dispatched to a queue message in MOM, which ensures that they are delivered to their intended recipients. Messages returned to the caller application are dealt with as soon as the calling program determines the time. MOM

makes it simple to maintain since it has a structure (schema) and content (data). MOM may be viewed as a single-record database that is shared among apps. Message-passing methods are used to send and receive information. Point-to-point and message queuing are the two communication models supported by MOM (MQ).

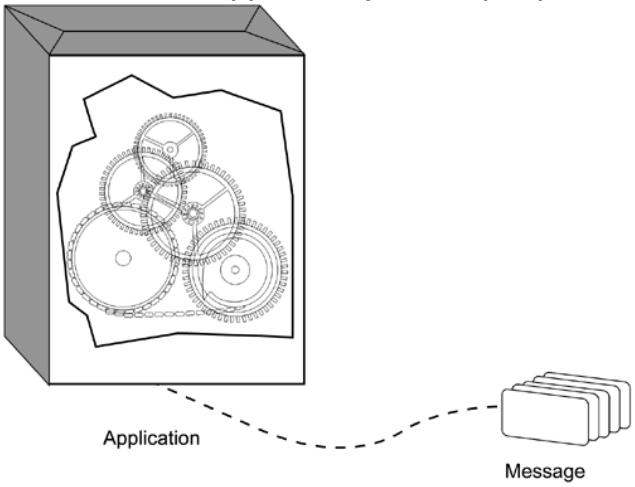


Figure 3-7: Message-oriented middleware does not stop the application from processing
Image Source: https://flylib.com/books/en/2.672.1.53/1/

C. Distributed Objects

Small application programs that interact with one another using common interfaces and protocols are known as **distributed objects**. It provides methods for application development, as well as enterprise-wide method sharing and supporting technologies. In today's market, there are two sorts of dispersed items: Common Object Request Broker Architecture (COBRA), and Component Object Model (COM).

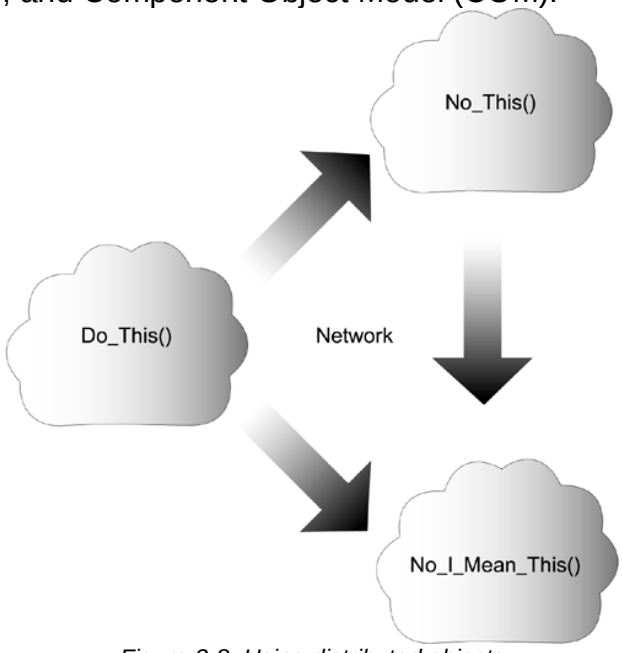


Figure 3-8: Using distributed objects
Image Source: https://flylib.com/books/en/2.672.1.53/1/

D. Database-Oriented Middleware

Database-oriented middleware makes it easier for an application or two databases to communicate with each other. It may be used as a technique to extract data from both local and distant databases. Call-level interfaces (CLI) and native database middleware are the two types of databases with which it interacts. CLIs are standard APIs that cover a wide range of database formats, allowing users to access several databases through a single, well-defined interface (Example: Open Database Connectivity (ODBC)).

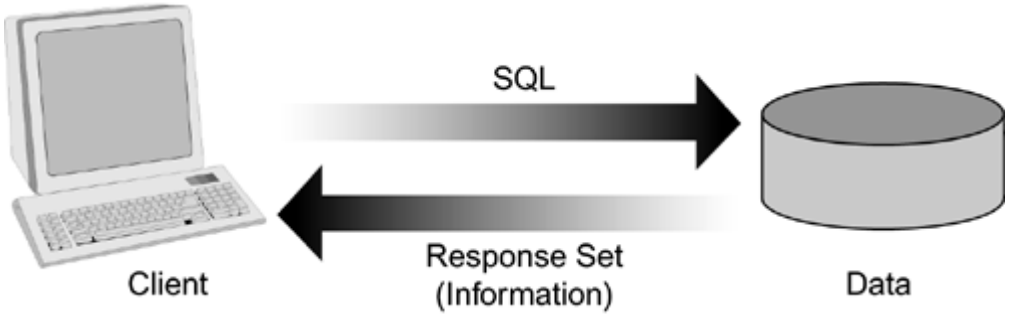


Figure 3-9: Database-oriented middleware
Image Source: https://flylib.com/books/en/2.672.1.53/1/

E. Transaction-Oriented Middleware

Transaction-oriented middleware (TOM) provides a framework for coordinating information flow and sharing methods across many resources. It enables integration that is strongly linked. It necessitates modifications to the source and target apps. Transaction-oriented middleware is built on the notion of a transaction, which is a discrete unit of work with a start and end date. A transaction is a container for application logic. The transaction either completes or is entirely turned back. TP monitors and application servers are two forms of transaction-oriented middleware.

TP Monitors

TP monitors serve as a hub for communication between two or more applications as well as a repository for application logic. It enables scalability by allowing other connected TP monitors to share and execute transactions. Connectors to databases, other programs, and queues are also available through TP monitors. In order to interface with these diverse resources, these connectors require some application development. These resources are incorporated into the transaction and are used as part of the transaction once they are linked. As a consequence, if a failure happens, the system may recover.

TP monitors services

1. Guarantee the integrity of transactions
2. Resource management and run-time management services

The load-balancing functionality of TP monitors provides the most benefit in terms of performance. It enables them to smoothly respond to a bombardment of transactions. As demand grows, the transaction manager adds new server

processes to accommodate the load while shutting down those that are no longer needed.

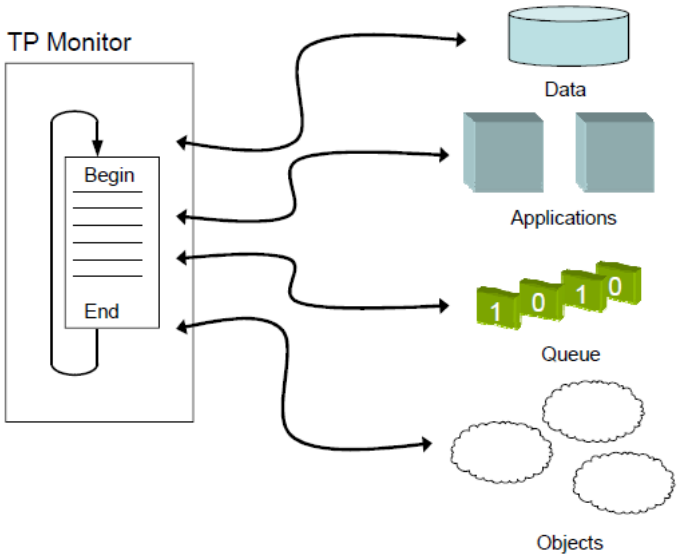


Figure 3-10: TP Monitor
Image Source: https://flylib.com/books/en/2.672.1.53/1/

Application Servers

Application servers provide the sharing and processing of application logic as well as access to back-end resources like as databases, ERP systems, and even mainframe applications. It provides user interface tools for deploying web applications.

F. Integration Servers

Integration Servers allow data to flow between two or more resources while accounting for variations in application semantics and platforms without requiring any application to comprehend the other applications with which it shares data.

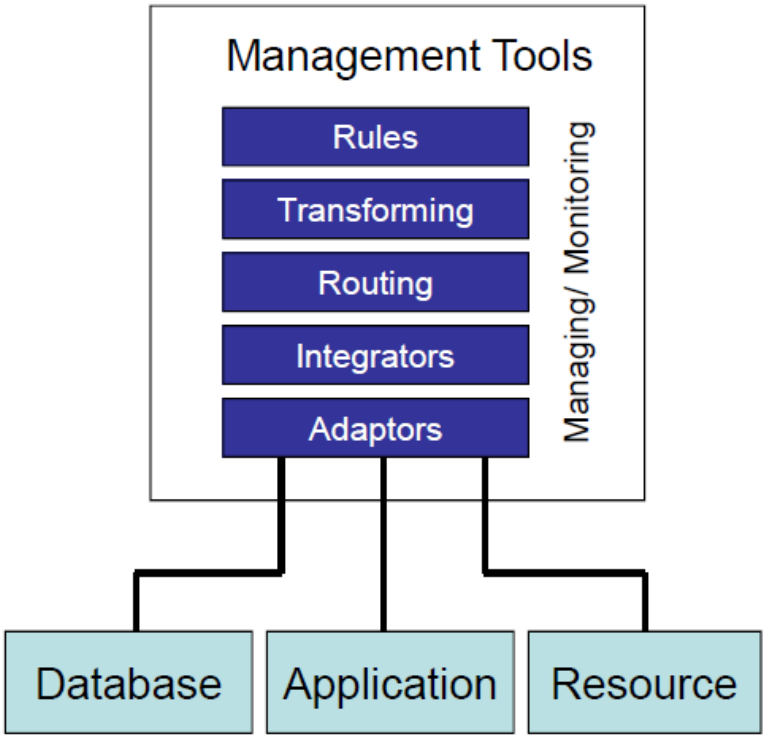


Figure 3-11: Integration Servers
Image Source: Next Generation Application Integration Textbook

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson in different platforms like Google Forms, Schoology, and the like.

LESSON 4
ENTERPRISE RESOURCE PLANNING SYSTEMS AND BUSINESS PROCESS MODELS

INTRODUCTION

Enterprise Resource Planning (ERP) software is the backbone of most businesses, allowing them to manage data across all departments. It aids in the administration of company-wide business operations and the usage of a shared database and management reporting tools. In this lesson, enterprise resource planning systems and business process models will be tackled to provide a broader perspective for students to deeply understand system integration.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to

1. Identify the key role of ERP in business functions;
2. Understand how ERP works in the functional areas of operation and business processes; and
3. Define the functional areas of IS and Enterprise System Architecture

DISCUSSION

Business processes are a set of actions that take an input and turn it into a useful output for the client. An ERP system's primary job is to assist business functions. To comprehend ERP, one must first comprehend how a firm operates. It's usually broken down into operational and business process functional categories.

Marketing and sales, supply chain management, accounting and finance, and human resources are all functional areas of operation. Business functions, on the other hand, include activities specific to a functional area of operation.

ERP Modules

ERP vendors such as SAP, Oracle, and Microsoft offer modules that support a company's major functional areas. The ERP software incorporates best business practices that use business rules to implement the organization's policies and procedures.

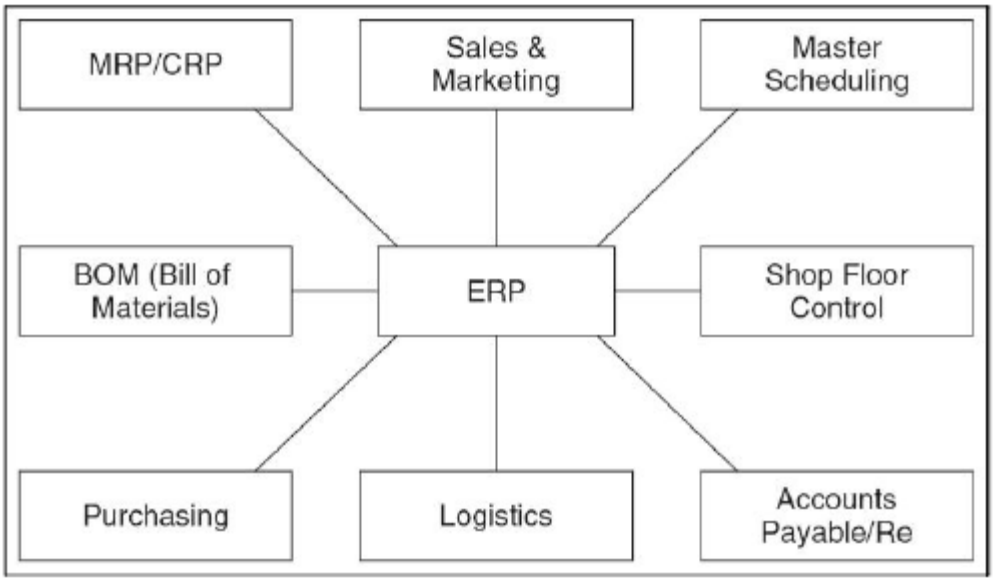


Figure 4-1: Example of ERP Modules
Image Source: Next Generation Application Integration Textbook

COMPARISON OF ERP MODULES FROM VENDORS

Function	**SAP**	**Oracle/PeopleSoft**	**Microsoft Dynamics**
Sales	Sales and Distribution, Sales Opportunity	Marketing and Sales, Supply Chain Management	Retail POS, Field Service Management
Procurement	Purchasing, Supplier Relationship Management	Procurement and Supplier Relationship Management	Supply Chain Management
Production	MRP, Product Life Cycle Management	Manufacturing	Manufacturing
Accounting	Financial Accounting	Financial Accounting	Financial Accounting
Distribution	Warehouse Management	Supply Chain Management	Distribution Management
Customer Service	CRM	CRM	CRM
Corporate Performance and Governance	Governance, Risk, and Compliance Management	Corporate Performance Management	Analytics
Human Resources	Human Capital Management	Human Capital Management	HR Management

HOW ERP MODULES WORK

1. Marketing and Sales

   Functions of marketing and sales module in an ERP includes advertising and marketing products, determining pricing, promoting products to

customer, taking customers' orders, helping create a sales forecast, customer support, and customer relationship management.

The Marketing and Sales module needs information from all other functional area. Customers communicate orders via Marketing and Sales in person or by telephone, e-mail, fax, the web, etc.

2. **Supply Chain Management**

   The functions within supply chain management includes making the product (manufacturing/production), buying raw materials(purchasing), receiving goods and raw materials, transportation and logistics, scheduling production of products, and plant maintenance.

3. **Accounting and Finance**

   Accounting and finance module includes recording raw data about transactions (including sales), raw materials purchases, payroll, and receipt of cash from customers, planning & budgeting, cost allocation & control.

4. **Human Resources**

   Human resources functions include recruit, train, evaluate, and compensate employees, benefits, and government compliance. HR uses data from other modules to plan personnel development path. Systems integrated using ERP software provide the data sharing necessary between functional areas.

5. **Production**

   Helps in the planning and optimizing of the manufacturing capacity, parts, components, and material resources using historical production data and sales forecasting.

6. **Purchasing**

   Streamlines the procurement process of required raw materials and other supplies

7. **Inventory Management**

   Facilitates the processes of maintaining the appropriate level of stock in a warehouse

8. **Miscellaneous Modules**

   Nontraditional modules such as business intelligence, self-service, project management, and e-commerce.

COMPONENTS OF THE ENTERPRISE SYSTEMS ARCHITECTURE

- **a.** Functional – Defines the ERP modules that support the various business functions of the organization. Examples include Accounting, Human Resources, Procurement, etc.
- **b.** System – Defines the ERP Architecture through the physical components of hardware, software, and networking angle.

Enterprise Systems Architecture

The architecture aids implementation teams in fully comprehending the enterprise system's features and components. It depicts the intricate system interfaces that exist between the ERP program and databases, operating systems, legacy applications, and networking. If the needs for system infrastructure, training, change management, and business process reengineering are clear, management may design a stronger IT plan.

Three-Tier Architecture

A three-tiered design, consisting of a Web Tier, an Application Tier, and a Data Tier, is used in the majority of modern ERP installations. Benefits include:

1. Scalability – Easier to add, change, and remove applications.
2. Reliability – Implementing multiple levels of redundancy
3. Flexibility – Flexibility in partitioning is very simple
4. Maintainability – Support and maintenance costs are less on one server
5. Reusability – Easier to implement reusable components
6. Security – IT staff has more control system to provide higher security

However, it can be very expensive and complex.

TIERS

1. The Web Tier

   Web-based portal allows users the ability to access and analyze information through their web browser.
2. The Application Tier

   The application tier consists of a web browser and reporting tool where business processes and end-users interact with the system. It shields the business users from the inner workings of an ERP system, but still provides the information relevant to their job and business process.
3. The Data Tier

   Data tier focuses on structure of all organizational data and its relationship with both internal and external systems.

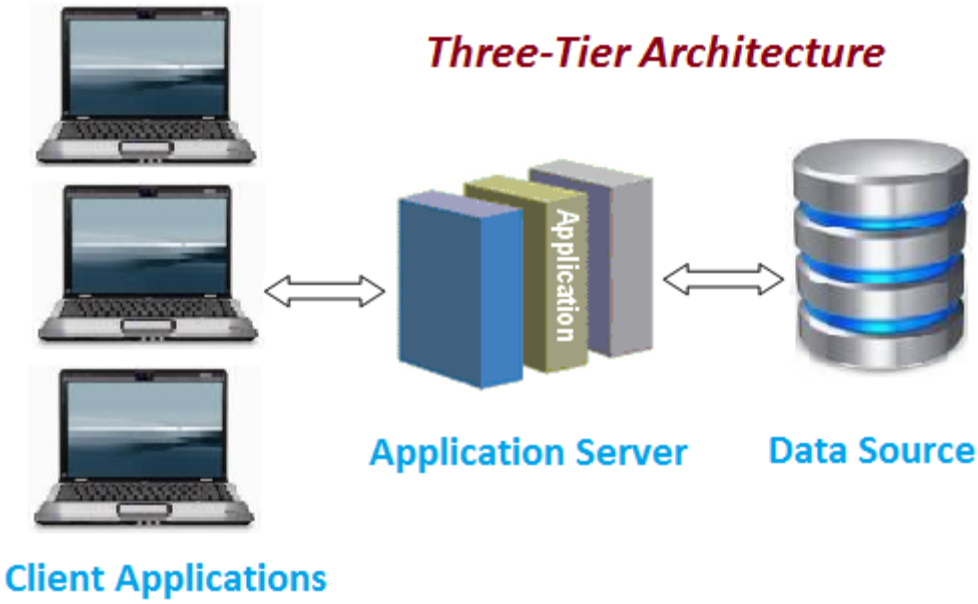


Figure 4-2: Three Tier Architecture
Image Source: https://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/

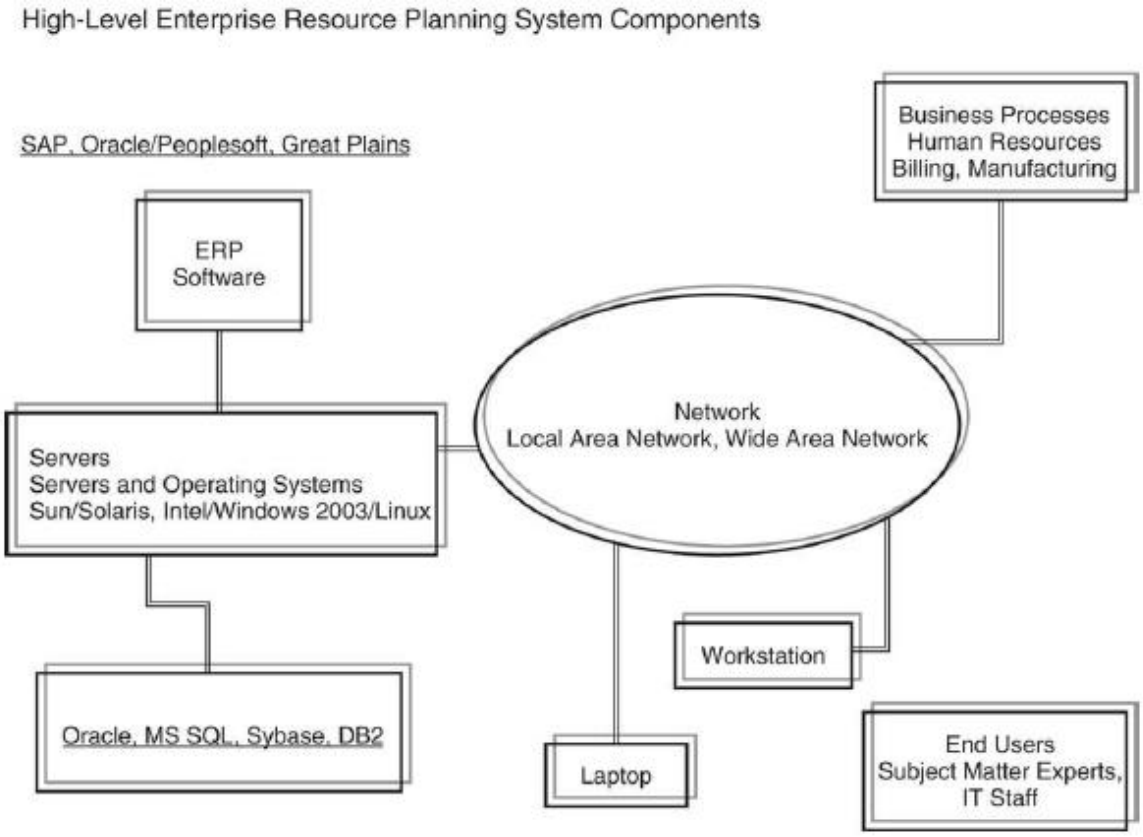
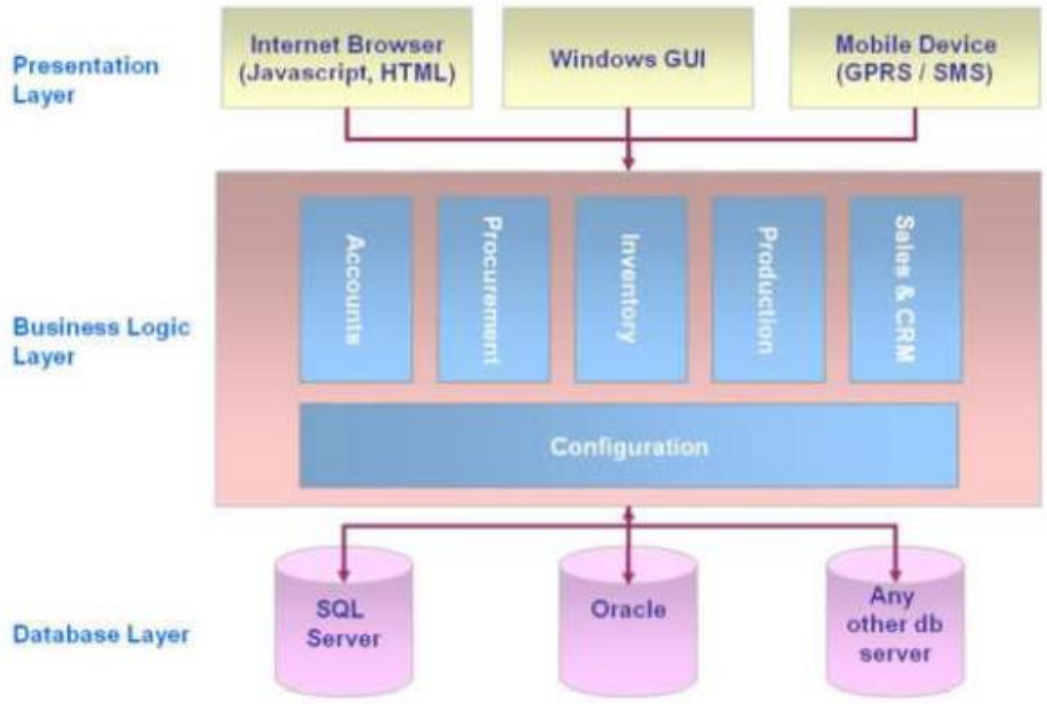


Figure 4-3: Example of Architecture of ERP at Large University



Source: http://www.weblineindia.com/erp_modules_solutions_tech_overview.htm

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson using different platforms like Google Forms, Schoology, and the like.

LESSON 5
INTEGRATION METHODOLOGIES

INTRODUCTION

In order to come up with a more feasible solution when dealing with business system or application integration projects, it is critical to follow appropriate processes and procedures. Students will apply their working knowledge of method-level, data-level, application interface-level, and user interface-level Enterprise Application Integration (EAI) in this course to propose a high-level approach, or methodology, that can be applied to most enterprise application integration projects.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to

1. Identify the 12 steps in Application Integration;
2. Construct understanding on how to different steps in application integration relates from one another; and
3. Construct realization on how to one must conduct a practical yet effective system integration activity.

DISCUSSION

Several techniques and actions must be completed and considered in EAI. Organizations have used many processes and procedures over time, and the following are the most typical actions, according to Linthicum (1999).

Step 1: Understanding the Enterprise and Problem Domain

This is the most difficult, time-consuming, and necessary step in the procedure. The issue domain must be examined both independently and within the context of the company. In order to fully comprehend the issues, organization chiefs/heads must be involved in order to have a complete knowledge of the context and structure of the different information systems, as well as the company's business requirements.

The fundamental requirements collecting techniques are used to determine the issue domain in this approach. It necessitates interacting with paper, people, and systems in order to gather the information needed to accurately describe the EAI problem so that it may be studied, modeled, and refined. After that, and only then, can the proper solution set be used. The quality of information in this stage influences the future steps and has a significant impact on the total output quality.

Step 2: Making Sense of the Data

Data must be the starting point for everything in EAI. Typically, EIA initiatives are simply data-driven. This fact alone warrants the effort to figure out what's in the many databases spread around an organization. Another point to consider is that databases must be understood even if the EAI project is focused on methods, application interfaces, and user interfaces.

Finally, identifying where data exists, obtaining information about the data (i.e., schema information), and using business principles to determine which data goes where and why is the key to implementing data-level EAI.

Three basic steps for implementing data-level EAI

a. Identifying the data
b. Cataloging the data
c. Building the enterprise metadata model

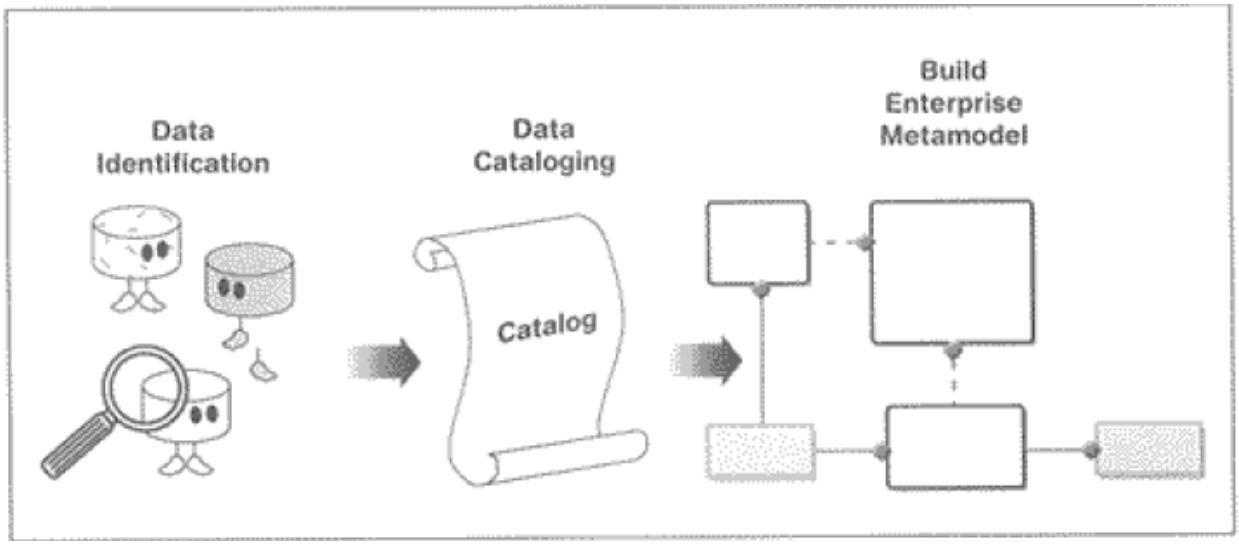


Figure 5-1: Implementing Data-Level EAI
Image Source: Enterprise Application Integration, David Linthicum, 1999

Step 3: Making Sense of the Processes

Following the understanding of enterprise data and the creation of baseline information such as the enterprise metadata model, a choice must be taken on how to approach the enterprise business model. This choice will be made based on how the specific EAI issue domain is addressed.

This is an enterprise perspective at the process, or method, level, comprehending and documenting all business processes and how they connect to one another, as well as the enterprise metadata model. Traditional process-modeling approaches, such as object modeling (e.g., Unified Modelling Language), should be used to design business processes, just as they should be used to create database analytical procedures.

Rather than starting from scratch with a set of application requirements, it is preferable to describe current business processes and methods in order to better

understand what they do and, as a result, how to combine them at the method level through a composite application.

Step 4: Identifying Application Interfaces

It's vital to identify the available application interfaces in support of application interface-level EAI, or integration with application interfaces and other EAI levels, in addition to looking for common methods and data to integrate. The user interfaces are peculiar. They differ significantly from one application to the next. Furthermore, many interfaces aren't truly interfacing at all, despite what program sellers or developers may assert. It's critical to spend time testing assumptions regarding interfaces.

The establishment of an application interface directory is the ideal place to start with interfaces. This is a repository for accumulated information on accessible interfaces, as well as documentation for each interface, similar to other folders. This directory is used to comprehend the sites of integration throughout all systems in the EAI problem domain, together with the shared business model and the enterprise metadata model.

Step 5: Identifying the Business Events

The next stage is to identify all important business events that take place within a company. This means that whenever anything happens—an event—there is a reaction. An "event" is something like a customer signing up for credit at an online Web store. It may be useful to record this occurrence and use it to trigger anything else, such as a credit check to ensure that the consumer is creditworthy. Of course, this can set off a chain of actions at the credit bureau, such as the credit bureau returning the client's credit status, which can set off even further events, such as alerting the customer by e-mail if the credit application is approved or refused. These events are usually asynchronous, although in rare cases they can be synchronous.

It's critical to comprehend what triggered a business event, what happens during the event, and any subsequent events that may occur as a result of the first event. The ultimate outcome is a network of interconnected occurrences, each of which is reliant on the previous.

Step 6: Identifying the Schema and Content Transformation Scenarios

It's a good idea to obtain a concept of how schema and content of data traveling across systems will be altered if you have a strong grasp of the data and application semantics that exist inside an EAI issue domain.

a. First, data from one system will not make sense in another unless the data schema and contents are reformatted to fit the target system's needs.
b. Second, it will ensure that application semantics are consistent from system to system.

Step 7: Mapping Information Movement

After the preceding stages have disclosed all of the information accessible, it's time to map the information movement from system to system, identifying which data element or interface the information is traveling from and where it will eventually go.

For example, the customer number from the sales databases must be sent to the credit-reporting system, where it will eventually be stored in the credit system's customer table. This knowledge allows the team to chart the flow of information from the source systems, sales systems, credit systems, and target systems. It's important to notice where the data is physically stored, what security measures are in place, what supporting technology is in place (e.g., relational tables), and how data is removed on one side and deposited on the other. It's also important to keep track of the event that's linked to the information flow. What additional criteria (such as data time, real time, or state changes) drives the flow of information from the source to the destination if no event is required?

Step 8: Applying Technology

Selecting the proper technology to solve the problem may be challenging yet fun. Application servers, distributed objects, and message brokers are just a few of the technologies accessible. The technology chosen will most likely be a mix of technologies and suppliers that, when combined, satisfy the EAI solution's requirements. It's extremely uncommon for a single seller to be able to address all problems—not that this has ever stopped merchants from claiming to be able to do so.

Technology selection is a challenging task that takes a significant amount of time and effort. Developing technology and product criteria, comprehending accessible solutions, and then matching the requirements to those goods is no easy task. This "marriage" of requirements and goods frequently necessitates a trial study to demonstrate that the technology will work. The time it takes to choose the proper technology might take as long as the EAI solution itself is created. While this may appear overwhelming, consider the alternative: selecting the incorrect technology for the task at hand. A poor decision almost guarantees the EAI project's failure.

Step 9: Testing, Testing, and Testing

Testing is time-consuming and costly. Testing is also thankless, which makes it an even more "appealing" task. Still, if an EAI solution isn't thoroughly tested, tragedy is a distinct possibility. Important data, for example, can be erased (and thus lost). Worse still, incorrect information may surface within apps. Even if none of these catastrophic scenarios occur, the solution must be scalable and capable of handling the additional rigors of day-to-day use. A test strategy will need to be put in place to ensure thorough testing. Because testing an EAI solution is

challenging, a test strategy is very necessary. The majority of source and target systems are mission-critical; hence, cannot be shut down. As a result, evaluating these systems might be difficult.

Step 10: Considering Performance

Too frequently, performance is overlooked until it is too late. One piece of advice: don't overlook performance! EAI systems that aren't up to par are doomed to fail. If, for example, completing a credit report for a telephone client takes 20 minutes during peak hours, the EAI solution isn't worth it. While most EAI solutions won't achieve zero latency with today's technology, the flow of data across systems or the execution of common business processes should achieve reaction times of less than a second. Furthermore, the EAI solution should maintain the same response time as the user and processing load grows. In a nutshell, the EAI solution will grow.

To build performance into a system, it is necessary to design for it and test for it before going live. Remember that after the EAI solution has been installed, performance cannot be improved. "From the ground up," performance must be planned. This means that the EAI solution's design must include both performance infrastructure and the chosen enabling technology. Traditional performance models, such as those established over the years in the field of distributed computing, can be used to make certain modifications before the solution is implemented. Finally, certain performance tests must be conducted to confirm that the system works properly under a range of situations. For example, how well does it perform at 100 users, 500 users, 1,000 users.

Step 11: Defining the Value

Two factors must be considered when determining value: soft and hard money. Simply said, hard money represents the value of an EAI solution as defined by its capacity to reduce costly operations, such as automating manual procedures, lowering mistake rates, or processing client orders more rapidly.

Soft money, on the other hand, is more difficult to describe. Improved productivity over time, increased retention rate owing to the capacity to make systems operate together for users, and increased customer satisfaction (based on ease of use) with an organization with integrated systems are all examples of cost savings.

Defining value in a system may, of course, be based on any criterion and will vary from issue domain to problem domain and from business to business.

Step 12: Creating Maintenance Procedures

It's important to think about how an EAI system will be maintained in the long run. Who is going to be in-charge of the message broker server? Who will be in-charge of security? Who will keep track of system performance and troubleshoot issues?

When dealing with the requirement for continuous maintenance, it's a good idea to list all of the tasks that must be completed and assign individuals to complete them.

Remember that an EAI solution is the beating heart of an organization, transporting crucial data between mission-critical systems. As a result, it is a potential point of failure that may bring the company to its knees. With that pleasant idea in mind, now could be a good time to think about disaster recovery problems like redundant servers and networks, as well as the possibility of moving the EAI system in the event of a disaster.

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson in different platforms like Google Forms, Schoology, and the like.

LESSON 6
DESIGNING SYSTEMS INTEGRATION SOLUTIONS AND ENTERPRISE INTEGRATION PATTERNS

INTRODUCTION

In the previous lessons, you were able to understand what middleware is. It is any type of software that facilitates communication between two or more software systems. It can be simple communication connection between applications. It can also be as sophisticated as information sharing and logic execution mechanisms. In this lesson, you will further understand how to design systems integration solutions and the different integration patterns.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to

1. Define tight coupling and loosely coupled;
2. Understand the process of exchanging pieces of data thru messages via a channel;
3. Understand the things that makes up middleware;
4. Develop integration solutions

DISCUSSION

Middleware is a technology that allows to move information between multiple systems that may reside within and outside an organization. Message-Oriented Middleware (MOM) is a queueing software that uses a message as a mechanism to move information from point to point. MOM uses the notion of messages to communicate between applications. MOM also resolves the problem of tight coupling between applications.

Tight Coupling

A great example of tight coupling is a local method invocation. Invoking a local method inside an application is based on a lot of assumptions between the called and the calling routing. Both methods have to run in the same process (e.g., a virtual machine) and be written in the same language (or at least use a common intermediate language or byte code). The calling method has to pass the exact number of expected parameters, each using the correct type.

The call is immediate (i.e., the called method starts processing immediately after the calling method makes the call. Meanwhile, the calling method will only resume processing when the called method completes (meaning, the invocation is synchronous). The communication between the methods is immediate and instantaneous, so neither the

caller nor the called method have to worry about security in the form of eavesdropping 3rd parties. All these assumptions make it very easy to write well-structured applications that break functionality into individual methods to be called by other methods.

Example Problem – Tight Coupled Solution

Assume the team is developing an online banking system that allows clients to deposit funds from another bank into their account. Only the person's name and the amount are required as inputs for the remote function to deposit money into their account. The front-end web application must be linked with the back-end financial system that handles fund transfers in order to fulfill this job. The TCP/IP protocol is the simplest technique to link the two computers. A TCP/IP stack can be found in an operating system or a programming library. TCP/IP is a widely used communication protocol that transmits data between the millions of computers that are linked to the internet and local networks.

Figure 6-1: Sample Scenario

This simple integration method is quick and inexpensive; however, it is brittle since the two parties involved make the following assumptions about each other:

a. Platform Technology – internal representations of numbers and objects (34-bit vs. 64-bit)
b. Location – Hard-coded machine addresses
c. Time – all components have to be available at the same time
d. Data format – the list of parameters and their types must match

When two people communicate, coupling is a measure of how many assumptions they make about each other. The participants must make a number of assumptions in order to follow the example. As a result, this solution is inextricably linked.

Loosely Coupled Solution

Instead of developing a tightly connected solution, the team can use a message-oriented middleware (MOM) architecture to create a loosely coupled solution. MOM mechanisms offer a variety of services, including:

a. A common data format and transformers – removes platform and data format dependencies as the sender no longer has to depend on the receiver's internal data format

b. Queueing channels – removes location and time dependencies as team do not have to pay attention to computer's identity and location or whether the other computer is ready to accept requests or not.

Removing these dependencies between the systems makes the overall solution more tolerant to change, the key benefit of loose coupling.

The Things That Make Up Middleware

Middleware are the things that sit between applications. In order to connect two systems via an integration solution, a number of things have to happen. Applications wants to exchange data via messages. Message are snippet of data that has an agreed-upon meaning to both applications that are to be integrated. This piece of data can be very small, such as the phone number of a single customer that has changed, or very large, such as the complete list of all customers and their associated addresses.

When two applications wish to exchange a piece of data, they do so by wrapping it in a message.

1. Message Intent
    a. Command Message – invoking a function or method on the receiver
    b. Document Message – enabling the sender to transmit one of its data structures
    c. Event Message – notifying the receiver of a change in the sender
2. Returning a response
    a. Request-Reply – the request is usually a command message, and the reply is a document message containing a result value or an exception
    b. Return Address – the requestor specifies what channel to use to transmit the reply
    c. Correlation Identified – responder specifies which request this reply corresponds to
3. Huge amounts of data
    a. Message sequence – break the data into more manageable chunks and send them as a sequence of messages, so that the receiver can reconstruct the original data structure
4. Slow messages
    a. Message expiration – the sender can specify an expiration date. If the messaging system cannot deliver a message by its expiration, it should discard the message. If a receiver gets a message after its expiration, it should discard the message.

Channel

Data needs to be transported, usually across a network. Communication channels are needed to move information from one application to the other. This channel could be a series of TCP/IP connections, a shared file, or a shared database. A channel is a logical address that both sender and receiver can agree on the same channel without being

aware of each other's identity. The application selects a particular channel to send the data knowing that the receiver will be one that is looking for that sort of data on that channel.

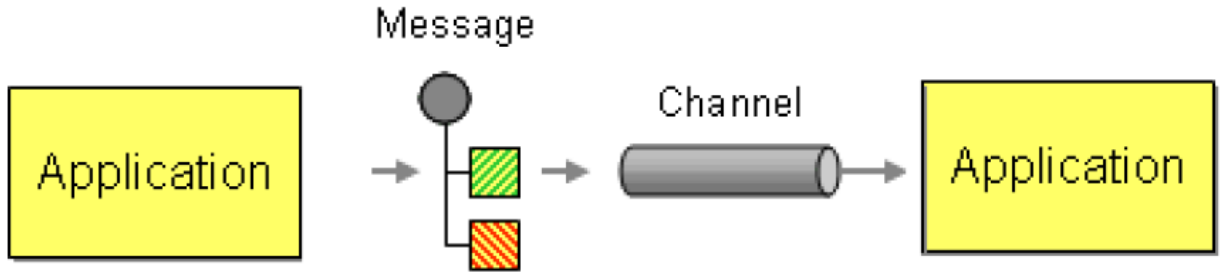


Figure 6-2: How Message are transported via channel

Messaging Channels

1. Point-to-point channel – To send the data to a single point (1-to-1 interaction)
2. Publish-Subscribe channel – In this channel, the channel effectively copies the data for each of the receivers (one to many interaction)
3. Datatype channel – all of the data on a channel has to be the same type (many to one interaction)
4. Invalid Message channel – when receiver receives a message that doesn't meet these expectations
5. Dead Letter channel – for messages which are successfully sent but ultimately cannot be successfully delivered
6. Guaranteed Delivery – makes channel persistent so that their messages are stored on disk
7. Channel Adapter – can be used to connect a channel (or set of channels) to the application without having to modify the application
8. Messaging Bridge – connecting two message systems, effectively connecting them into one composite messaging system
9. Message Bus – a backbone providing access to all the enterprise's various and ever-changing applications and functionality

Translation

Middleware needs to provide some mechanisms to convert one application's data format in the others. Internal data format of an application can often not be changed. For example, one data format may store the customers' name in two fields, called FIRST_NAME and LAST_NAME, while the other system may use a single field called CUSTOMER_NAME.

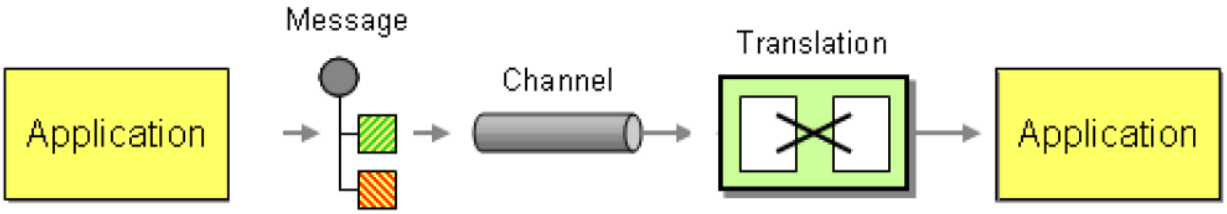


Figure 6-3: Translation

Message Transformation

1. Envelope Wrapper – wrap message payload data into an envelope that is compliant with the requirements of the messaging infrastructure
2. Content Enricher – when the target system requires data fields that the originating system cannot supply. It has the ability to look up missing information or compute it from the available data
3. Content Filter – removes unwanted data from a message
4. Claim Check – removes data from a message but stores it for later retrieval
5. Normalizer – translates messages arriving in many different formats into a common format
6. Canonical Data Model – design an independent data model from any specific application. Require each application to produce and consume messages in this common format.

Routing

Middleware needs to take care of sending messages to multiple systems. As the number of systems increases, it becomes very tedious and requires the sending system to have knowledge about all other systems. For example, if the customer's address changes in the customer care system, that system can be responsible for sending the data to all other systems that store copies of the customer address. It is expected that each application specific the target system/s for the data it is sending over the channel.

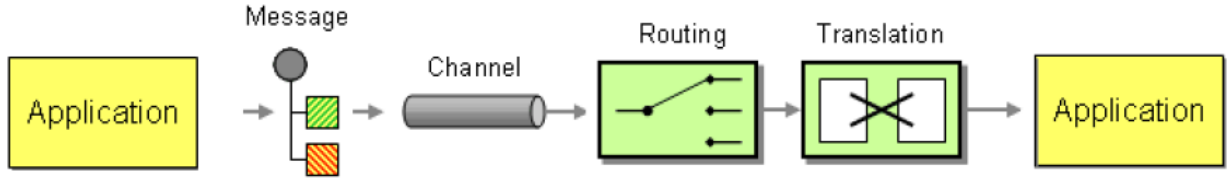


Figure 6-4: Routing

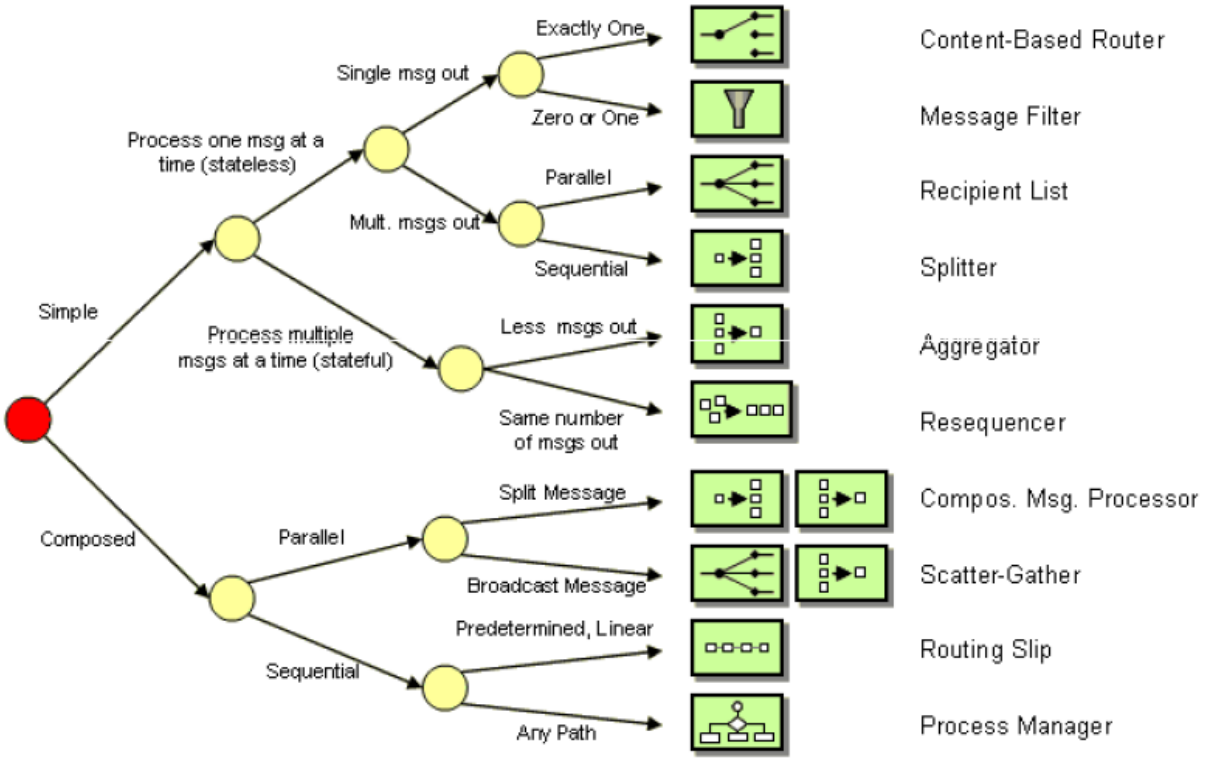


Figure 6-4: Message Routing

Systems Management

Integration solutions can quickly become complex because they deal with multiple applications, data formats, channels, routing, and transformation. All these elements may be spread across multiple operating platforms and geographic locations. In order to have any idea what is going on inside the system, it is essential that systems management function be recognized. This subsystem monitors the flow of data, makes sure that all applications and components are available, and reports error conditions to a central location.

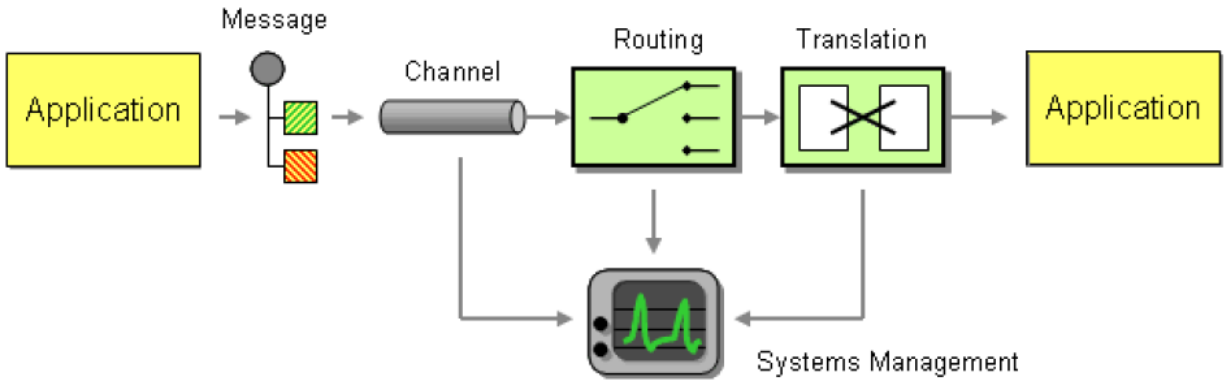


Figure 6-5: Systems Management

1. Control Bus – provides a single point of control to manage and monitor a distributed solution

2. Detour – route messages through additional steps, such as validation or logging – with ability to switch on or off these additional steps
3. Wire Tap – inspect the contents of a message without affecting the primary message flow
4. Message History – great aid to know where a specific message has been
5. Message Store – can provide a complete account of every message that traveled through the system
6. Smart Proxy – track messages sent to request-reply services
7. Test Message – actively verifying that the running messaging system is functioning properly
8. Channel Purger – remove all remaining messages from a channel so that the components under test do not receive 'leftover' messages.

Endpoint

Most packages and legacy applications and many custom applications are not prepared to participate in an integration solution. As they were designed to perform specific functionality with specific input/output. Message endpoint is needed to connect the system explicitly to the integration solution. The endpoint can be a special piece of code or an adapter provided by an integration software vendor.

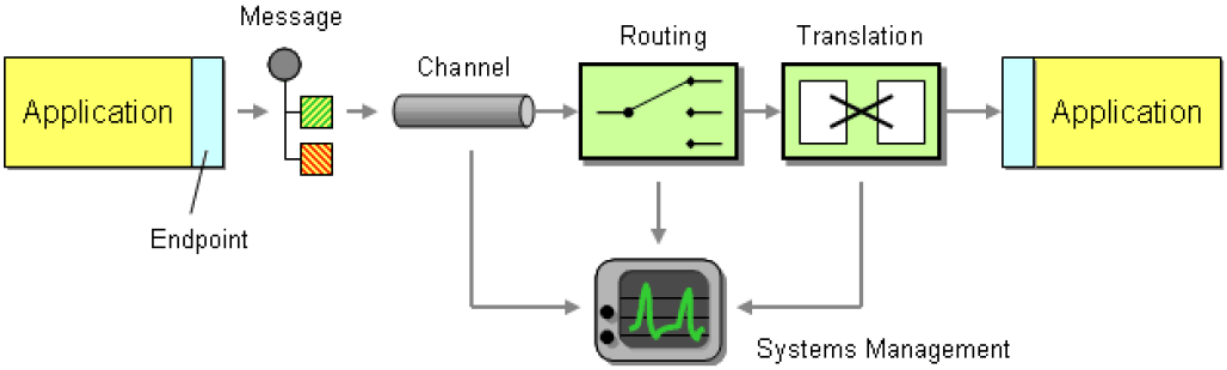


Figure 6-6: Endpoint

1. Messaging Gateway – a class that wraps messaging-specific method calls and exposes domain-specific methods to the application
2. Messaging Mapper – creates a mapping logic between the messaging infrastructure and the domain objects
3. Transactional Client – make the client's session with the messaging system transactional so that the client can specify transaction boundaries
4. Polling Consumer – explicitly makes a call when it wants to receive a message
5. Event-Driven Consumer – automatically handles messages as they're delivered on the channel
6. Competing Consumers – create multiple consumers on a single channel so that multiple messages can be processed concurrently

7. Message Dispatcher – consumer messages from a channel and distribute them to performers
8. Selective Consumer – filters messages delivered to a channel so that it only receives the ones that match its criteria
9. Durable Subscriber – to make the messaging system save messages published while the subscriber is disconnected
10. Idempotent Receiver – can safely receive the same message multiple times
11. Service Activator – connects the messages on the channel to the service being accessed.

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson in different platforms like Google Forms, Schoology, and the like.

LESSON 7
XML AND APPLICATION INTEGRATION

INTRODUCTION

Extensible Markup Language (XML) is a data transmission format that encapsulates both data and information. It was designed to store and transport data. XML are both machine and human readable. It encourages the creation of a self-contained message structure. On the internet, there is a common format for exchanging information. In the area of application integration, it's also an infrastructure for information sharing and administration.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to

1. Know the value of XML and XSLT mechanism; and
2. Understand the use of XSLT for B2B application integration

DISCUSSION

XML is a reliable, human-readable data interchange system. It facilitates the interchange of information content and application semantics. It provides a method for creating business data at the application level. It's crucial to keep in mind that apps don't have to know anything about each other. XML is also a mechanism for storing and transferring data that is device and program agnostic. XML, like HTML, is a mark-up language.

Unlike HTML, which was created with the goal of displaying data with a focus on how it appears, XML was created with the goal of carrying data with a focus on what data is. In addition, unlike HTML tags, XML tags are not predefined.

Middleware and XML

The term "middleware" refers to software that "carries the load." It transports XML-encapsulated or abstracted communications. It guarantees that every source or target program that requires the information understands the messages. The interfaces with the source or target application are managed by middleware. It transports data into and out of the application. To consume and create XML, it is essential to make modifications to the source and destination apps. XML necessitates a significant amount of overhead. A binary message requires less bytes than an XML document.

XML Elements

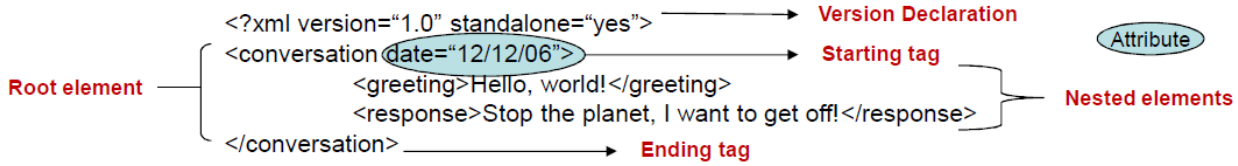


Figure 7-1: XML Elements
Image Source: Next Generation Application Integration Textbook

1. Elements – Basic Building blocks of an XML document
2. Defined by tags – Starting tag and an ending tag
3. Root Element – Outermost element in the XML document
4. Nested Elements – Elements within elements. This ability allows XML to support hierarchical structure
5. Element name – describes the content of the element
6. Structure describes the relationship between the elements
7. Attributes of an element – describe characteristics of the elements in the beginning tag of an element.

XML Schema

In the same way as database schema provide a structural model for the data they represent; XML Schema governs what can and cannot be done with XML data.

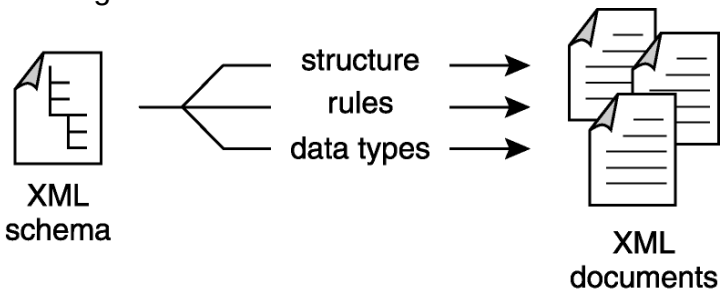


Figure 7-2: XML Schema
Image Source: Next Generation Application Integration Textbook

The vocabulary of elements is used to construct an XML document. Vocabularies can be explicitly specified using XML Schema, a schema definition language. It provides structure, validation rules, data type restrictions, and inter-element connections to ensure the integrity of XML document contents. As an enterprise data transfer standard, XML schema is critical since it substantially improves the quality of XML data. Authors can use schema to create logical domains to which any or all of the schema can be applied.

XML Schema and XML Document

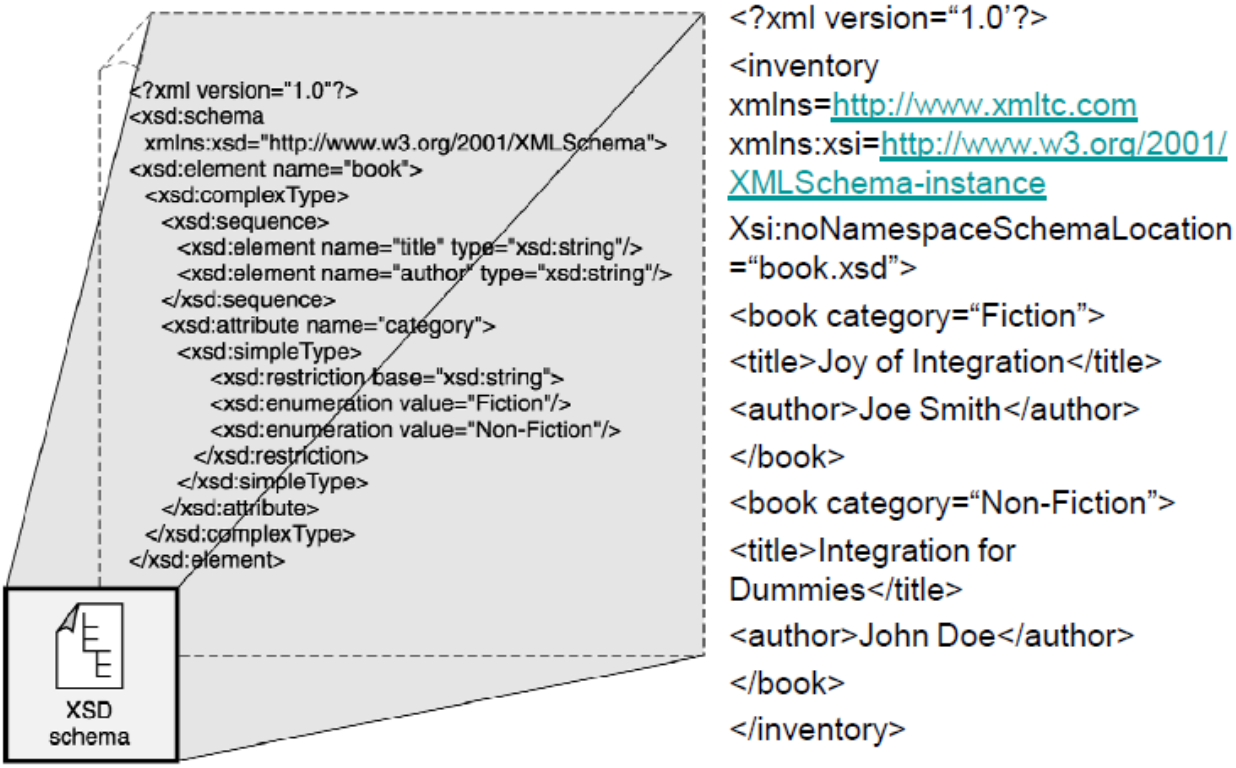


Figure 7-3: An XML Document
Image Source: Next Generation Application Integration Textbook

XML Parsers

XML parsers read XML documents and extract the data so that another software can access it. The Document Object Model (DOM) is a tree-based API, whereas SAX (Simple API for XML) is an event-based API.

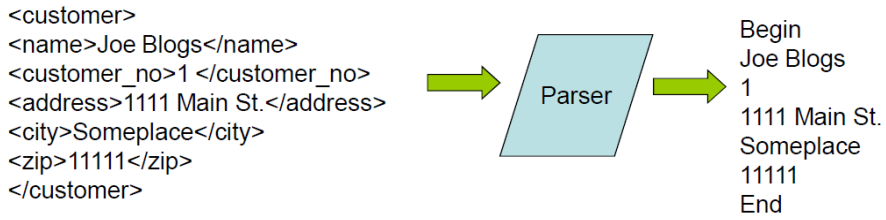


Figure 7-4: Example of XML Parser
Image Source: Next Generation Application Integration Textbook

XML Namespaces

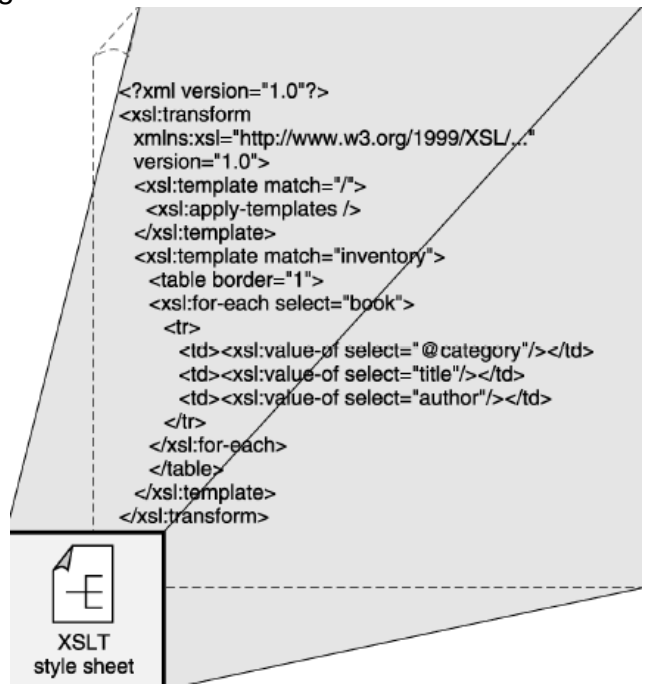
A namespace is a set of names that may be used as element or attribute names in an XML document. It's critical to associate names with a specific domain and minimize repetition. XML namespaces allow the same term to have many meanings. A Uniform Resource Indicator (URI) is used to identify namespaces, allowing each one to be distinct.

For example, there can be three elements known as “account” (e.g., frequent-flyer account, bank account, customer account at hotel). Each account name is associated with a particular domain (e.g., Airline URL https://www.airline.org).

The capacity of XML namespaces to create shared application semantics amongst trading partners is critical in the context of application integration.

What is XSLT?

Extensible Stylesheet Language Transformations (XSLT) is a programming language that allows you to change the structure and content of an XML document. It is a text processing system at its most basic level. Other standard markup languages can be generated. XSLT is an XML document transformation standard that uses a stylesheet as the processing engine.



Fiction	Joy of Integration	Joe Smith
Non-Fiction	Integration for Dummies	John Doe

Figure 7-5: Example of XSLT Document
Image Source: Next Generation Application Integration Textbook

XSLT Mechanism

Transforming an XML document using XSLT requires two main steps

1. Structural transformation

   Data is transformed from the input structure to the output structure. It involves selecting data, grouping it, sorting it, or aggregating it.
2. Formatting the text to new characteristics

   Information is placed in a particular type of text structure. Examples include XML, HTML, PDF, etc.

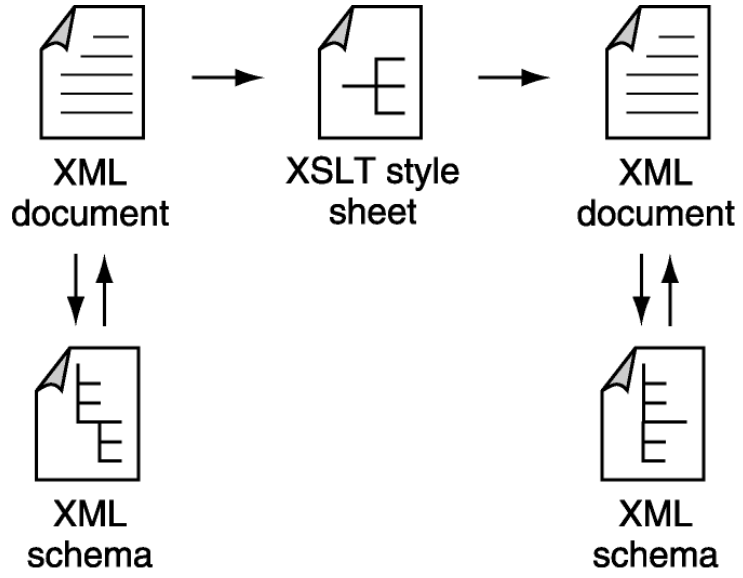


Figure 7-6: XSLT Mechanism
Image Source: Next Generation Application Integration Textbook

XSLT Processors

While staying consistent with the way processors handle XML through trees, XSLT processors apply an XSLT style sheet to an XML source document and generate a resultant document. Three trees must be processed by XSLT: the input tree, the stylesheet tree, and the output tree. The transformation to be performed is defined in the stylesheet document. The stylesheet tree is used by the XSLT processor to transform the input tree to the output tree.

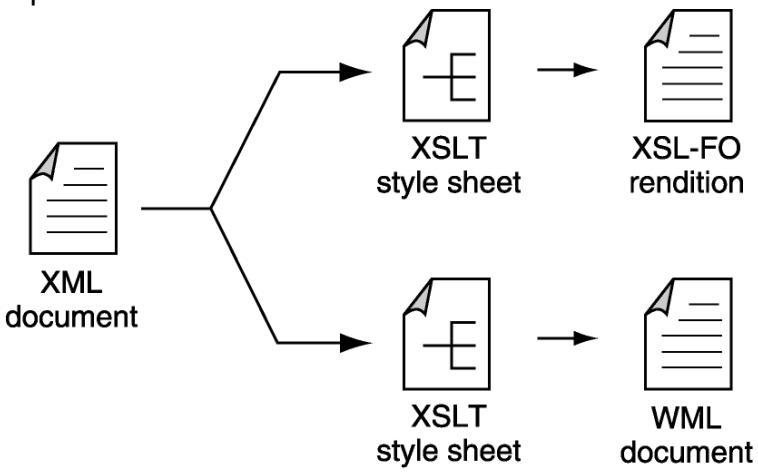


Figure 7-7: XSLT Processors
Image Source: Next Generation Application Integration Textbook

Using XSLT for B2B Application Integration

Both rules and transformation processes are standardized with XSLT. It is the chosen standard technique for changing content and application semantics when data travels from one application to the next and from one company to the next. Text-based formats can also be created with XSLT.

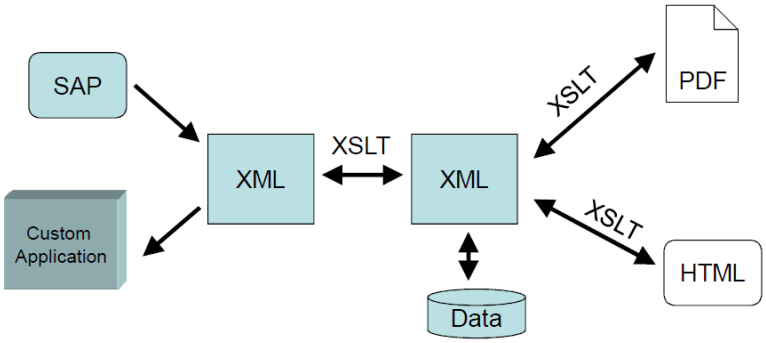


Figure 7-8: XSLT for B2B
Image Source: Next Generation Application Integration Textbook

XSLT can establish a standard data transport format within and between application tiers.

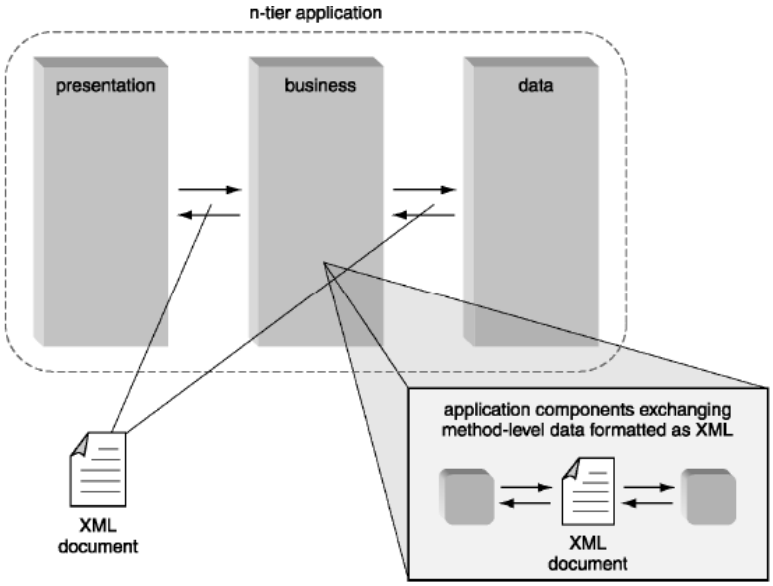


Figure 7-9: XSLT within and between application tier
Image Source: Next Generation Application Integration Textbook

XML documents can be used as the standard data transport throughout integrated environments.

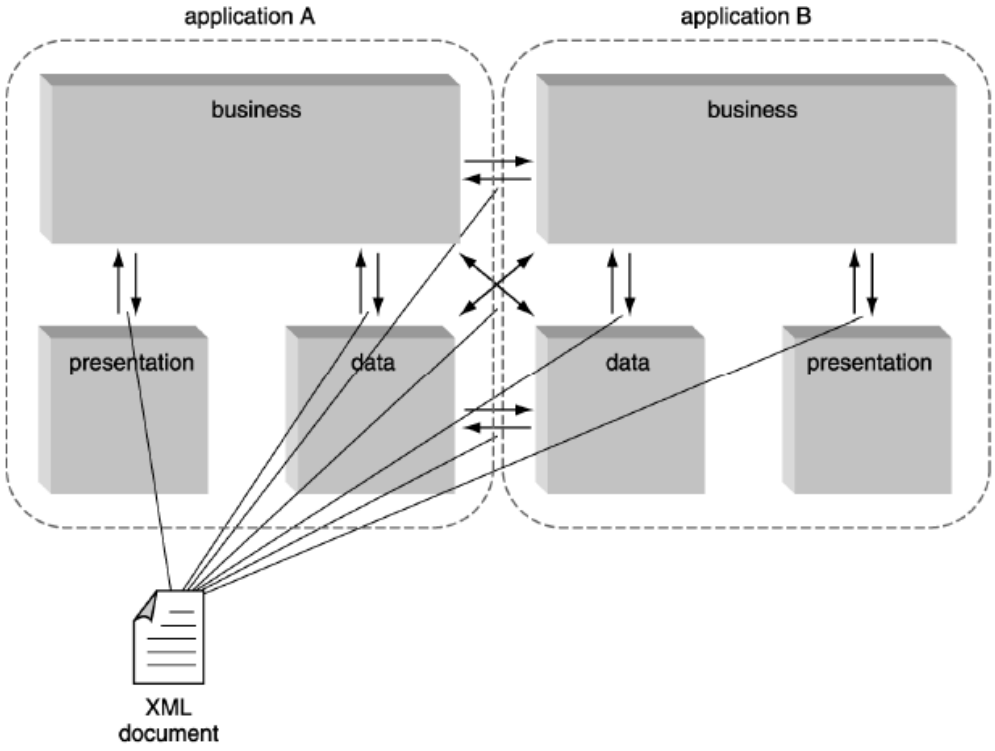


Figure 7-10: XML in integrated environments
Image Source: Next Generation Application Integration Textbook

XSD Schemas validating incoming data for each application endpoint

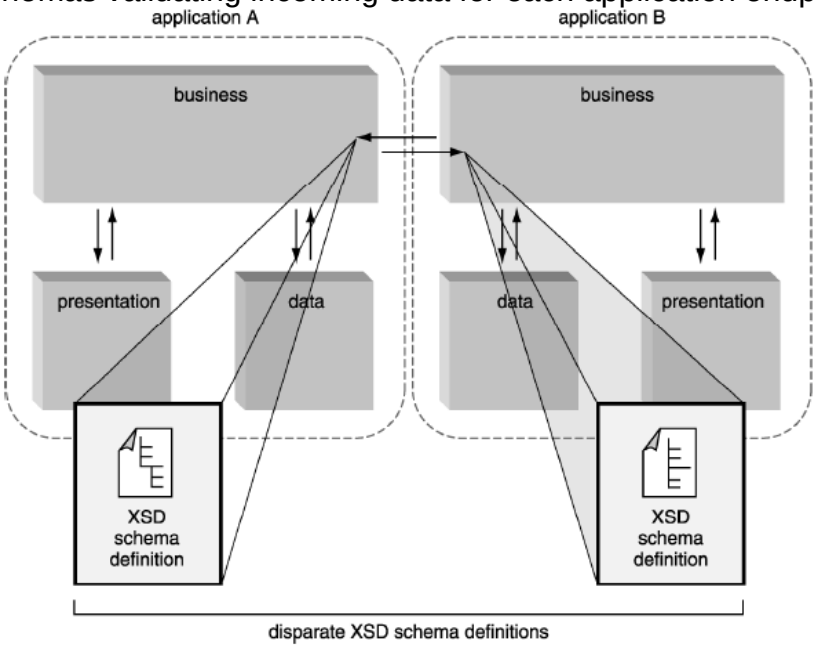


Figure 7-11: Validating incoming data for application endpoint
Image Source: Next Generation Application Integration Textbook

XSLT can perform dynamic structural transformations

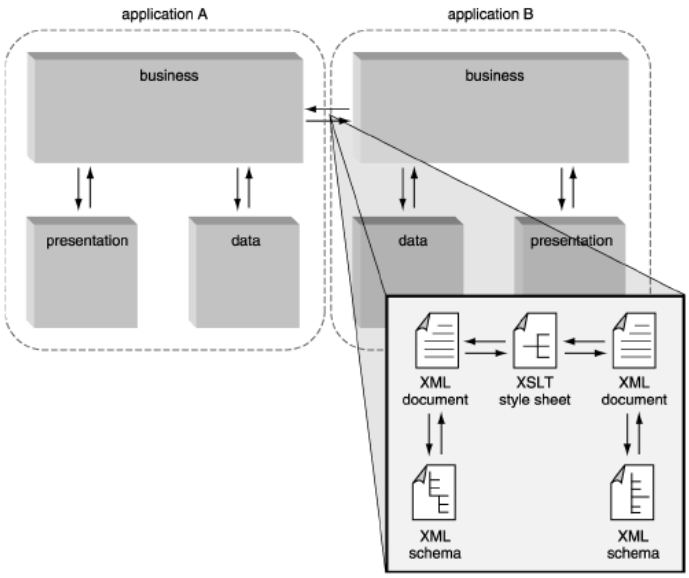


Figure 7-12: Dynamic Structural Transformation
Image Source: Next Generation Application Integration Textbook

Presentation centric transformations can be incorporated to output data for different mediums.

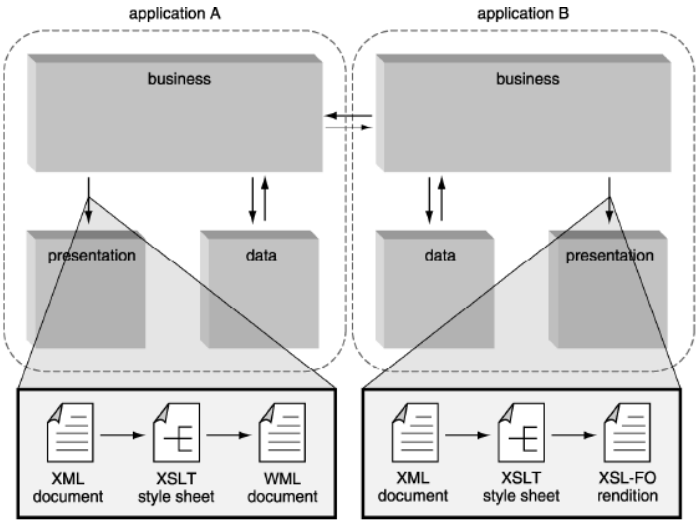


Figure 7-13: Presentation Centric Transformation
Image Source: Next Generation Application Integration Textbook

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson in different platforms like Google Forms, Schoology, and the like.

LESSON 8
SERVICE-ORIENTED ARCHITECTURE AND WEB SERVICES

INTRODUCTION

Service-Oriented Architecture (SOA) is a design pattern for creating software applications that make use of services offered through a network, such as the internet. It encourages software components to be loosely coupled so that they may be reused. SOA applications are made up of services. A service is a software implementation of a well-defined business function that may be used by customers in a variety of applications or business processes. (Mahmoud, 2005). In this lesson, you will be able to learn these concepts and apply such concepts for systems integration in the future.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to

1. Define Service-Oriented Architecture and Web services;
2. Know the different kinds of Web Services and Applications

DISCUSSION

Over the course of many years, most businesses have made significant investments in system resources. Because such businesses have a large quantity of data stored in outdated enterprise information systems (EIS), it is not viable to replace them. Evolving and improving EIS is more cost-effective. A cost-effective option is Service Oriented Architecture (SOA).

The notion of service-oriented architecture (SOA) is not new. Sun used the term SOA in the late 1990s to characterize Jini, a network-based environment for dynamic discovery and usage of services. Web services have taken Jini technology's notion of services and turned it into web-based services utilizing technologies like XML, Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery, and Integration (UDDI) (UDDI).

In today's complex and diverse computer environment, SOA is emerging as the primary integration and architectural framework. Previous initiatives depended on proprietary APIs and needed a high level of coordination between parties, rather than enabling open interoperable solutions. SOA enables the software as a service idea by helping businesses simplify operations so they can do business more effectively and react to changing requirements and competition. For example, eBay is making its web services API available for its online auction. The objective is to encourage developers to use the eBay platform to generate money. Developers may use the new APIs to create bespoke applications that connect to the online auction site and allow them to submit things for

sale. Because consumers must still go to ebay.com to bid on products, such apps are generally geared for sellers (Mahmoud, 2005).

Service Oriented Architecture

SOA is a design pattern for creating software applications that make use of services offered through a network, such as the internet. It encourages software components to be loosely coupled so that they may be reused. SOA applications are made up of services. A service is a software implementation of a well-defined business function that may be used by customers in a variety of applications or business processes.

SOA enables the reuse of existing assets, allowing for the creation of new services from an existing IT architecture of systems. In other words, it allows organizations to make the most of their existing investments by allowing them to reuse existing programs, as well as ensuring compatibility across disparate systems and technology.

Levels of Flexibility of SOA

1. Services are implementation-independent software components having well-defined interfaces. The separation of the service interface (the what) from its implementation is a key component of SOA (the how). Clients who are unconcerned about how these services will carry out their demands use these services.
2. Services are self-contained (they execute specific activities) and loosely linked (for independence)
3. Services can be found in real time.
4. Aggregates of different services can be used to create composite services.

Figure 8-1 depicts the find-bind-execute paradigm used by SOA. Service providers register their services in a public registry in this model. Consumers use this register to locate services that meet specific requirements. If the registry offers such a service, the customer is given a contract as well as the service's endpoint address.

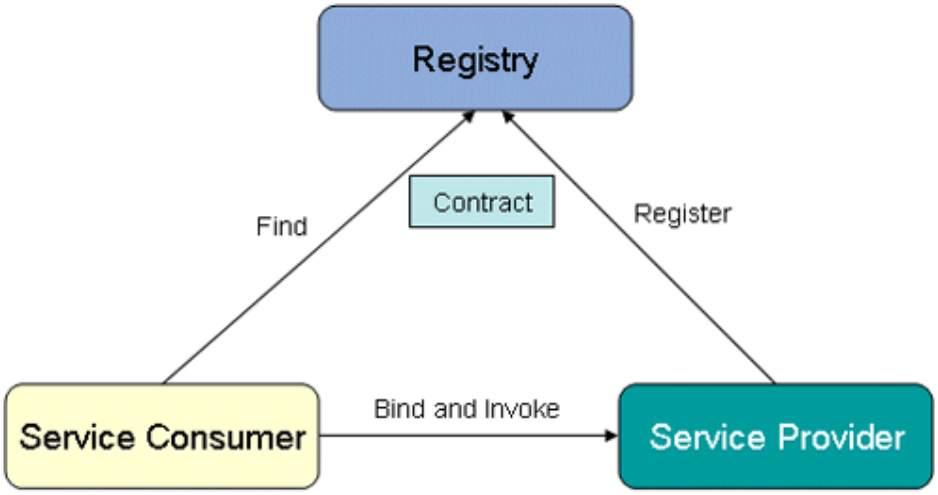


Figure 8-1: SOA's Find-Bind-Execute Paradigm
Image Source: https://www.oracle.com/technical-resources/articles/javase/soa.html

SOA-based applications contain presentation, business logic, and persistence layers and are distributed multi-tier applications. SOA applications are built on the foundation of services. While any functionality may be turned into a service, the issue is defining the proper level of abstraction for the service interface. The functionality of services should be coarse-grained.

Supporting Interactions

The purpose of systems integration is to support each of the following: Human to Human Interactions, Human to Machine (Applications) Interactions, and Machine (Application) to Machine Interactions.

Constructs Necessary to Support Interactions

Different things are necessary to support interactions such as:

1. Language (English, ASP.Net, Java)
2. Vocabulary (Meaning of words, meaning of code)
3. Context (Conversation context)
4. Medium (Communication channel, ability to speak, and ability to listen)
5. Situation awareness (Recognizing where you are, role of the person)

New Frontier: Service Computing

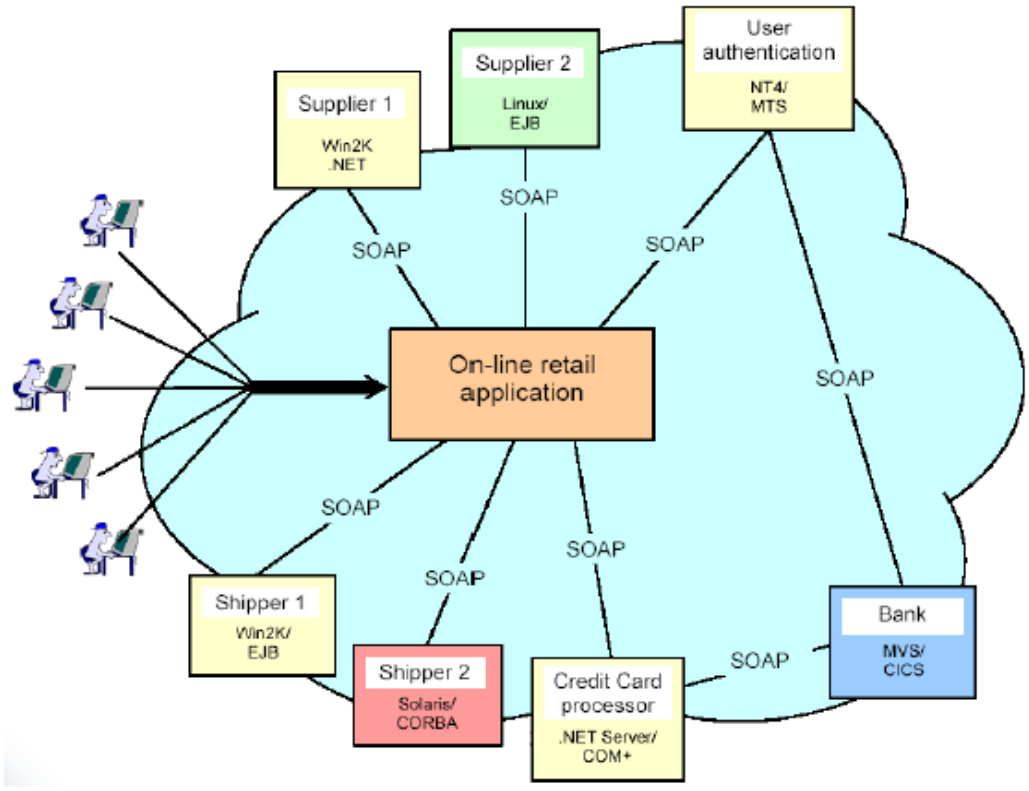


Figure 8-2: Web Service Vision
Image Source: Coyle, F.P, XML, Web Services, and the Data Revolution, 2002, ISBN: 0-201-84456-7

Web Services

Application-to-application communication is supported through web services. It allows for integration that is "loosely linked." It reduces the amount of time and effort needed to create interconnected apps. Web services allow applications or programmers to locate cooperating programs to complete a given job, allowing programming to quickly construct applications by just putting application modules together. Web services are intended to allow application modules (objects) to communicate with one another. Service apps provide transactional computing services once they are linked.

Three Main Parts of Web Services

1. Data is presented and shared using XML
2. Applications find services and share information and data across diverse systems environments using shared, open, emerging technology standards – SOAP, UDDI, WSDI, etc.
3. How communications take place over a common network (the Internet) using HTTP protocol as the transport.

Let's see this example!

Consider a scenario of finding a plumber to fix a plumbing problem

1. Find a plumber in Yellow Pages (Discovery)
2. Plumber should have advertised in the Yellow Pages (Publishing)
3. You call plumber to schedule appointment (Binding)

Three Basic Participants

1. Service provider – develops an application and converts into a service. They create a Web Service Description Language (WSDL) document describing the capabilities of the service and how to access the service. They publish the WSDL document in a service registry (UDDI)
2. Service registry
3. Service requestor – Needs a service for specific tasks/purpose. They search for services in the service registry meeting needs and selects a service that is satisfactory. The service requestor's application invokes the provider's service. Upon acceptance, application and service can exchange data using SOAP.

Three Basic Operations

1. Publishing services
2. Discovering services
3. Binding services

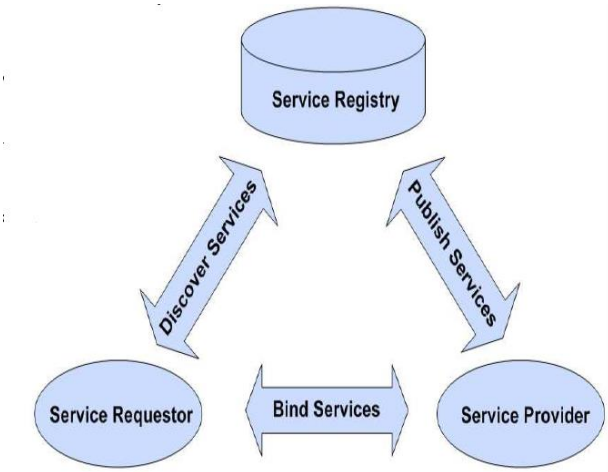


Figure 8-3: SOA
Image Source: Next Generation Application Integration Textbook

Key Web Service Standards

1. SOAP – Simple Object Access Protocol
    a. Submitted to W3C in 2000
    b. SOAP 1.1 became standard in July 2003
    c. SOAP 1.2 became standard in April 2007 (current)
    d. XML-based messaging format established a transmission framework for inter-application (or inter-service) communication via HTTP
    e. SOAP specification is vendor neutral technology; therefore, it was an attractive alternative to proprietary protocols such as COBRA and DCOM
2. WSDL – Web Service Description Language
    a. XML-based language for describing the interface of web services
    b. WSDL 1.1 became standard in March 2001
    c. WSDL 2.0 became standard in June 2007 (current)
3. UDDI – Universal Description, Discovery and Implementation (UDDI)
    a. Mechanism for the dynamic discovery of service descriptions
    b. UDDI 2 became standard in April 2003
    c. UDDI 3 became standard in February 2005 (current)

Relationship between Key Web Services Standards

1. Use UDDI to search for services. Use SOAP to access UDDI. Search using key words
2. Download WSDL document from UDDI for selected service
3. Use URI information from WSDL document to invoke the service using SOAP
4. After the services accepts the request, then the application and the service can exchange data using SOAP.

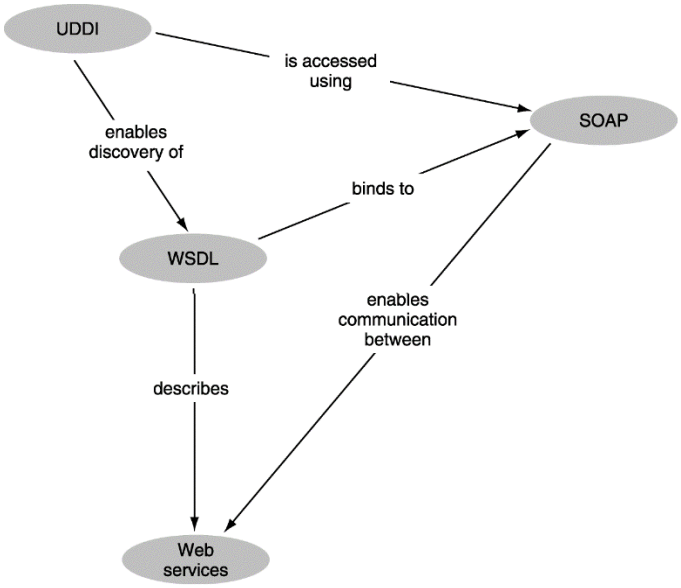


Figure 8-4: Relationship between key Web Services Standards

Client Server and Peer-to-Peer Architecture

Service Oriented Architecture is more like Peer-to-Peer Architecture

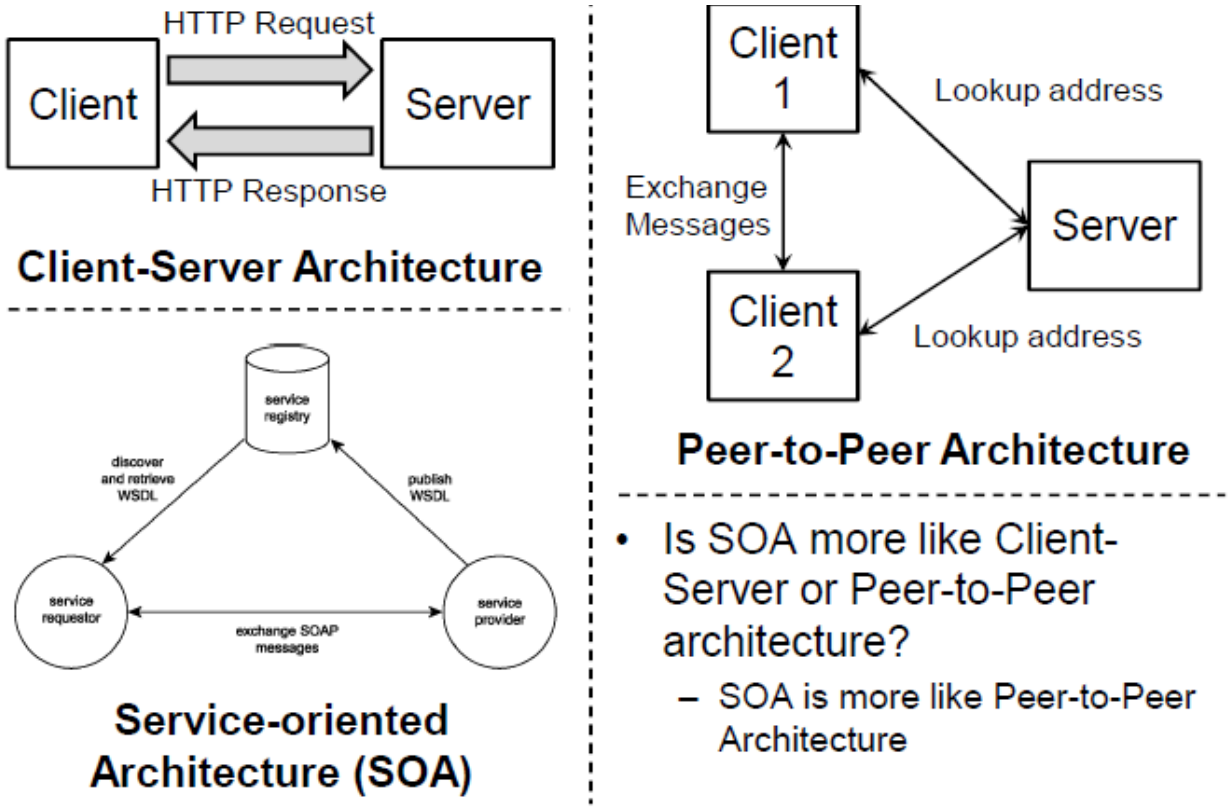


Figure 8-5: Client-Server, SOA, and Peer-to-Peer Architecture

Web-Based Application Development Architecture

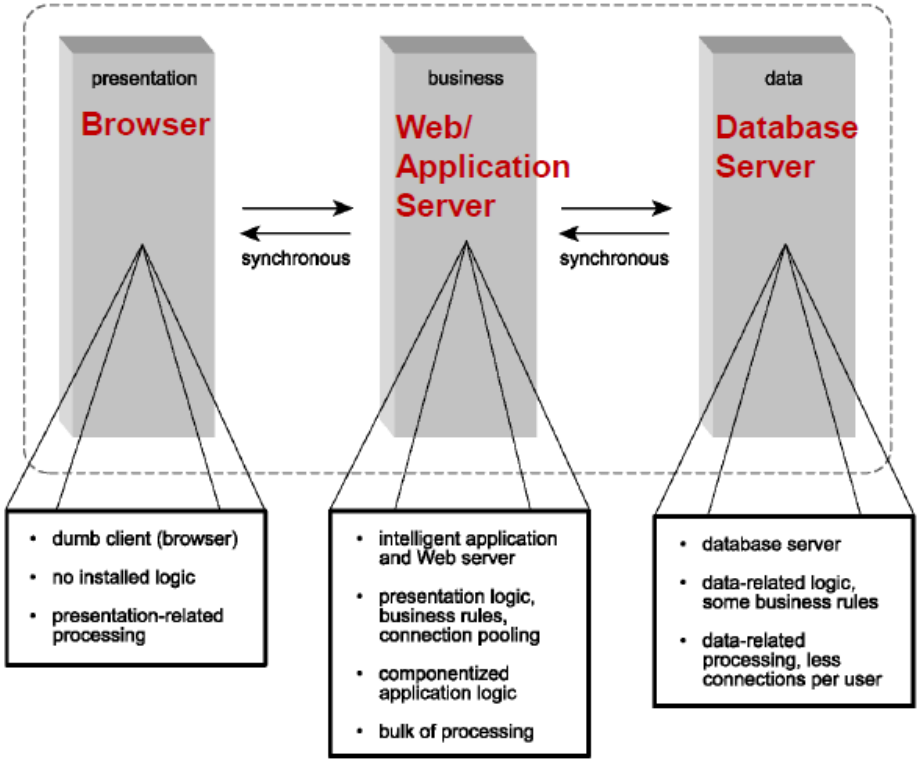


Figure 8-6: Mobile Application Development

Application Integration

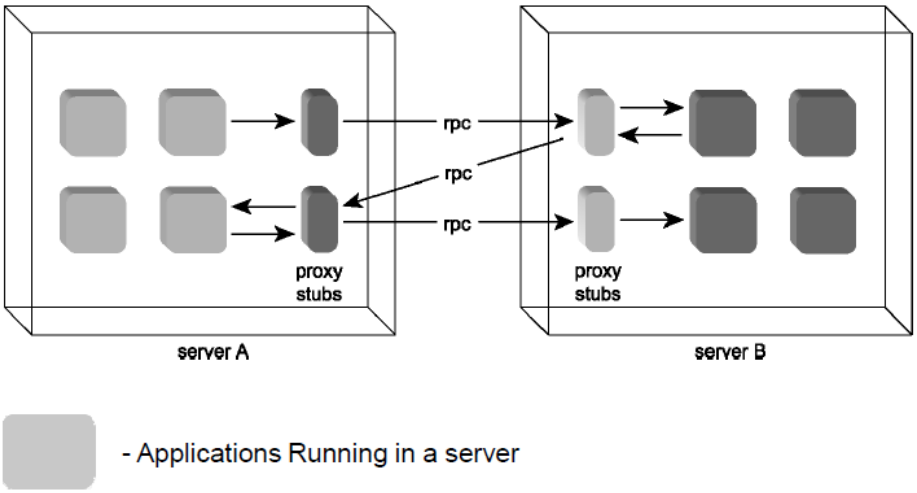


Figure 8-7: Application Integration Representation

Application Integration as Services

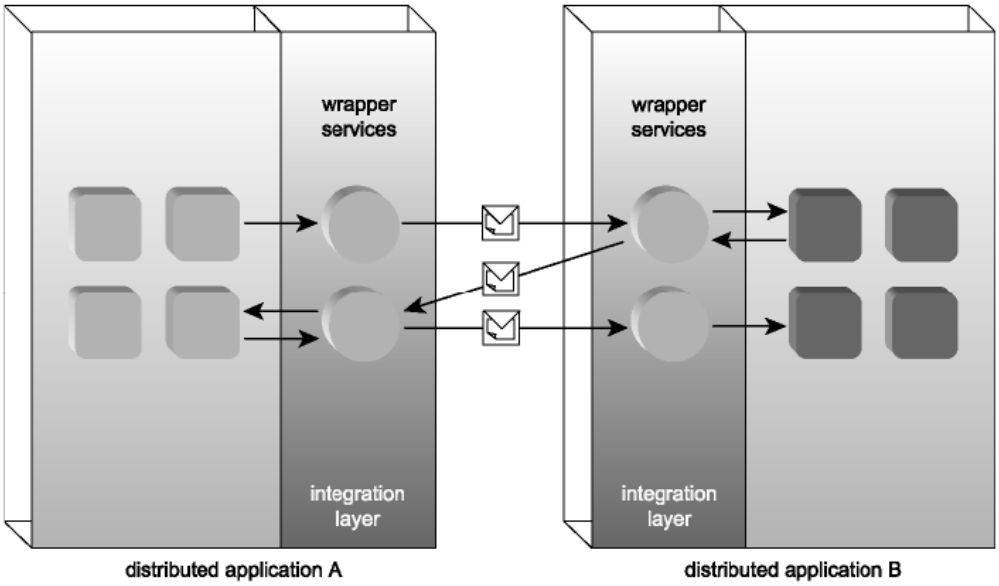


Figure 8-8: Application Integration as Services

Wrapper is a middleware. Middleware techniques and integration patterns can be applied to develop wrapper services.

A logical representation of a Web Service based on Integration

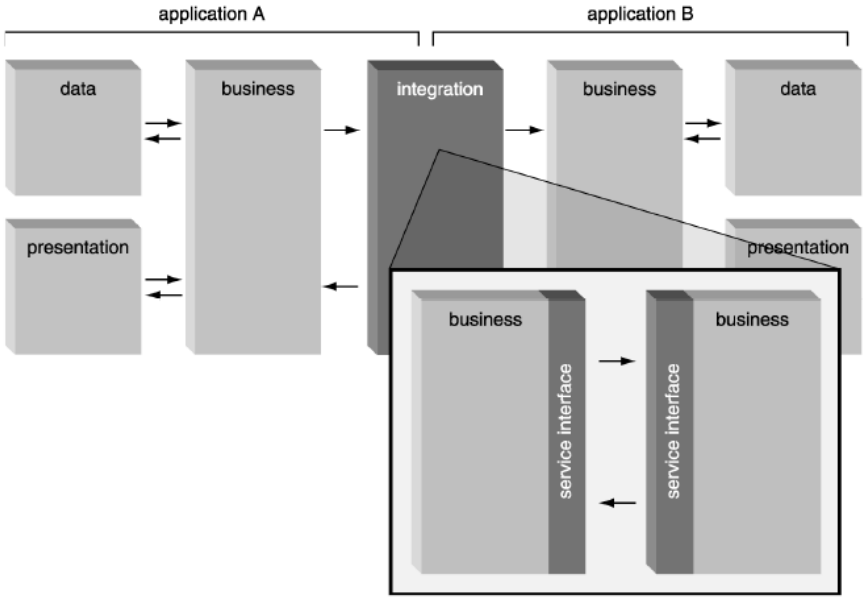


Figure 8-9: Web Service Based on Integration

WSDL documents representing Web Services to Applications

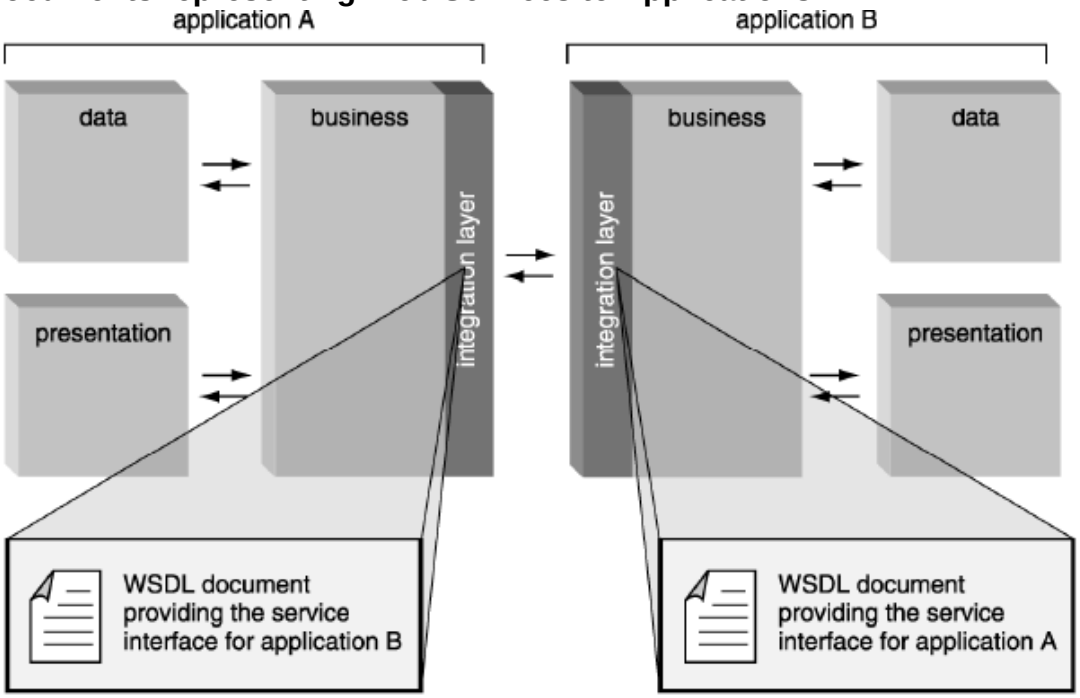


Figure 8-10: WSDL and Web Services Applications

The integration layer introduced by the Web services framework establishes a standard, universally, recognized and supported programmatic interface. WSDL enables communication between these layers by providing standardized endpoint descriptions.

Web Service Description Language (WSDL)

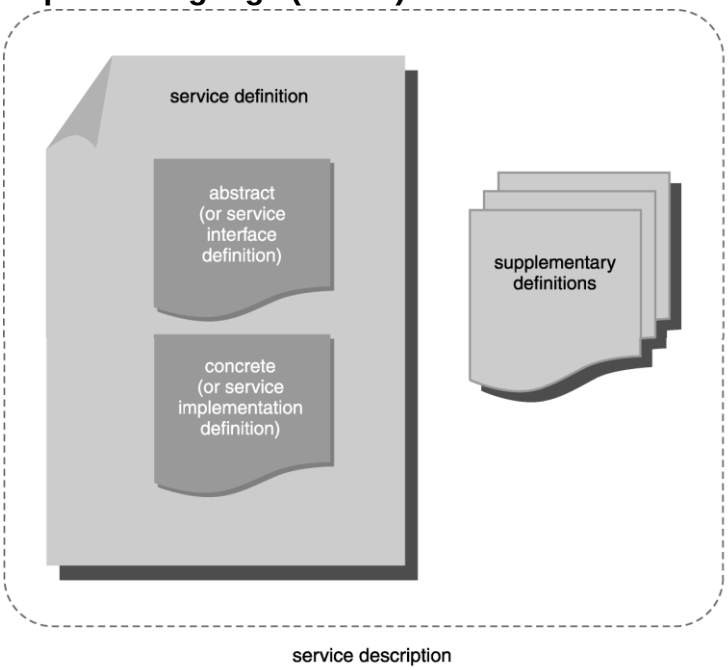


Figure 8-11: WDSL

1. Abstract – the description of a web service interface, independent of implementation details. Abstract interface definition is made up of interface and message elements.
2. Concrete – specifies location and implementation information about a web service. Concrete interface definition is made up of binding, endpoint, and service elements.

WSDL Definition Elements – Abstract Interface Definition

1. Interface – contain a group of logically related operations
2. Operations – represents a single action or function performed by an application (i.e., method in an application). Operations consist of group of related input and output messages. Message exchange required to support execution of the operation.
3. Message element can contain one or more input or output parameter that belong to an operation. Part elements are used to define parameters. Part element provides a name, value set, along with an associated data type.

WSDL Definition Elements – Concrete (Implementation) Definition

1. Service – represents one or more endpoints at which the web service can be accessed.
2. Endpoint – consist of location (URI) and protocol information
3. Binding – defines invocation requirements of each of its operation. Associate's protocol and message format information to operations. Operations construct within Binding block resembles its counterparts in the interface section. Each Endpoint can reference to a binding element, and therefore relates the endpoint information to the underlying operation.

WSDL Supplementary Constructs

a. Provide additional information about the service
b. Type element – provide data type support for web service definitions. XSD Schema information is provided
c. Documentation element allows supplementary annotations to be added.

Simple Object Access Protocol (SOAP)

SOAP specification establishes a standard message format that consists of an XML document capable of hosting RPC and document-centric data. Document-centric is most commonly used, due to standard endpoint description is provided in WSDL.

SOAP facilitates synchronous (request and response) as well as asynchronous (process-driven) data exchange models.

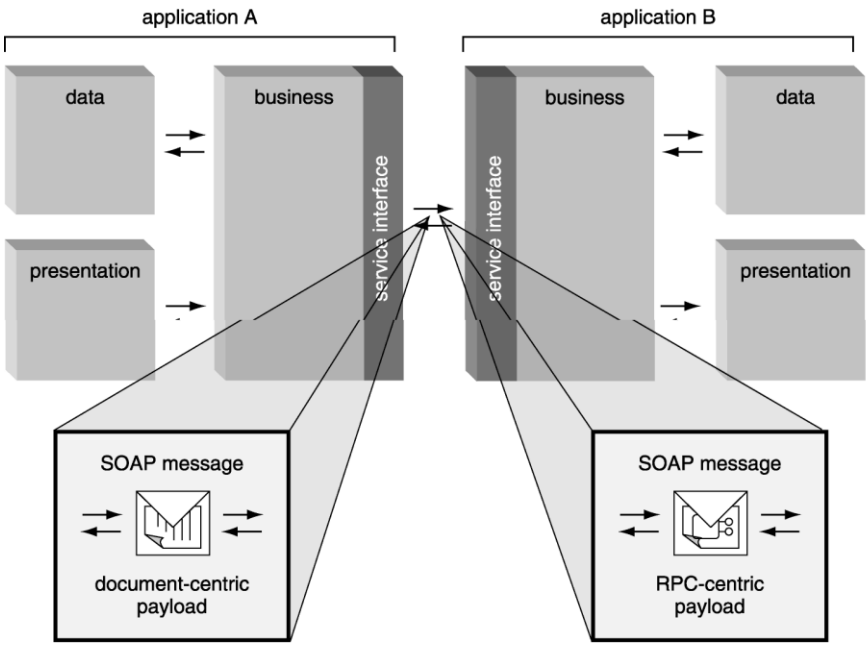


Figure 8-12: SOAP

SOAP Message Structure

The root Envelope element frames the message document consist of a mandatory Body element and an optional Header element.

Header element is used for

a. Including implementation of SOAP extensions (advanced web service standards)
b. Identification of target SOAP intermediaries
c. Processing information for SOAP intermediaries
d. Providing supplementary Meta information about the SOAP message.

Within the Body element, data being delivered by the SOAP message is included. Fault element can be used host exception information. Embedded within Body element.

Example SOAP Document

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope>
  <Header>                                                    Header Block
     <shipping>UPS</shipping>
  </Header>
  <Body>
     <Book xmlns=http://www.example.ws/>
        <Title>SOA A Field Guide to Integration XML and Web Services</Title>
     </Book>
     <Fault>
        <Code>                                                Body Block
           <Value>VersionMismatch</Value>
        </Code>
        <Reason>
           <Text>Version do not match</Text>
        </Reason>
     </Fault>
  </Body>
</Envelope>
```

Figure 8-13: Example SOAP document

Universal Description, Discovery, and Implementation (UDDI)

Fundamental of SOA is a mechanism for service descriptions to be discovered by potential requestors. UDDI is a central directory that hosts service descriptions (including WSDL). UDDI specifies a registry that stores service descriptions within a directory. UDDI registry can be public registry, a global directory of services or a private registry, a repository of services hosted within an organization.

UDDI Registry Elements

1. Business Entities (businessEntity element)

   Provides profile information about the registered business, including its name, a description, and a unique identifier.
2. Business Services (businessServices element)

   Records of the actual services offered by the registered business are nested within the businessEntity element
3. Specification Pointers (bindingTemplate element)

   Provides address linking the businessService to implementation information. Developer can learn how and where to physical bind to a web service.
4. Service Types (iModel element)

   Points to location of service interface definitions (WSDL), message formats, as well as message and security protocols.
5. Business Relationships (publisherAssertion element)

   Provides a means of establishing the relationship of the current businessEntity with another.
6. Subscriptions (subscription element)

   Allows subscribers to be notified when business entity profile information is updated.

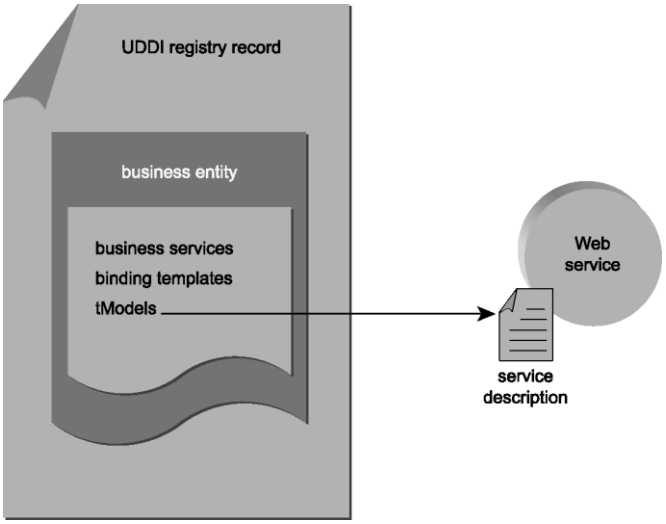


Figure 8-14: UDDI Registry Elements

Accessing UDDI Registry

UDDI provides inquiry and publishing APIs allowing applications to interface programmatically with a registry. Registries are expected to provide interface for humans as well.

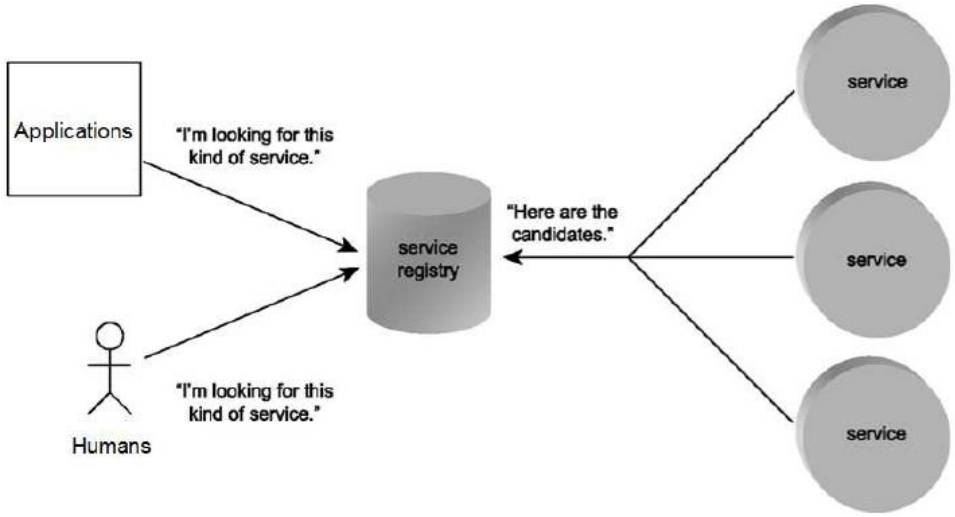


Figure 8-15: Accessing UDDI Registry

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson in different platforms like Google Forms, Schoology, and the like.

LESSON 9
SELECTING COMMERCIAL-OFF-THE-SHELF PRODUCTS

INTRODUCTION

Usually, vendors often blame for unsuccessful implementation of an ERP for System Integration. This lesson discusses how it is usually not the case and how, from the beginning, to best position of an organization to implement an ERP successfully. Once the decision or case has been made to replace the existing application system(s), the initial step involves the purchase of an ERP system. The selection of a vendor that best meets the needs and long-term direction of the company is a critical first step in its successful implementation.

LESSON OBJECTIVES

At the end of the lesson, the students must be able to

1. Understand the initial steps in the process for the successful purchase and implementation of an ERP system for Integration.
2. Gain knowledge on how to properly implement ERP.

DISCUSSION

In selecting a vendor, a well-understood selection process needs to be utilized. Depending on a given company's experience in purchasing an ERP system, that company may want to bring in a specialized consulting firm to assist in the selection process. In any event the steps involved in selecting a vendor generally are based on best fit of an ERP to business functions and the overall ERP vendor's product performance in the market.

High-Level ERP Purchase Process

1. Vendor Research and Information Gathering
2. High-Level vendor demonstration and evaluation
3. Needs and requirements assessment using current legacy systems, business process reengineering analysis, or both
4. Development of request for bid or proposal (if needed or desired)
5. Research request for bid to vendors
6. Analysis and selection
    a. Evaluation of bids
    b. Functional evaluation
    c. Technical evaluation

d. Vendor-detailed demonstration
e. Contact references
f. Develop a total cost of ownership

7. Vendor/s negotiation
    a. Contract review and change
    b. Pricing – software, maintenance, and consulting support
    c. Purchase system

Vendor Research

The first step in selecting an ERP system is generally to research vendor ERP systems in the market to identify a short list of vendors who will help to shape business requirements. This process is especially helpful for companies moving from aging legacy systems and technology to current and state-of-the-art technology. A state-of-the-art ERP system purchase will likely mean the replacement of the current hardware and software infrastructure. Identifying and researching all aspects of a vendor package and the platform that the hardware and software runs on will assist companies in determining the total cost of ownership (TCO).

In general, identifying vendors today is not overly difficult. Using current web search engines is a good starting point. It is also help in determining what packages the competition uses. An exhaustive list of vendors, even if you do not research them completely is important for a successful implementation. Another strategy to identify vendors is to ask department managers and subject matter experts if they know of vendors that should be considered. It will be said many times, “the process is important”, so including end users will help with change management issues later in the project. It will also help to gain and secure trust for later in the implementation.

Things to consider when researching vendors

a. Other businesses using the vendor
b. The vendor’s financial position
c. The vendor’s implementation philosophy and support issues
d. The hardware and software infrastructure used to support the ERP
e. The vendor’s direction and currency of software
f. The vendor’s release and upgrade strategies
g. The vendor’s user-based involvement in defining future functional changes
h. The vendor’s development and maintenance resources

Some ERP vendor systems (e.g., SAP) are designed for and can scale to a large number of users, whereas other systems (e.g., Great Plains) are geared for a small number of users. Many ERP vendors have similarly geared their application for a specific industry. For example, PeopleSoft has historically focused on government and educational organizations, whereas SAP has focused on the manufacturing industry. Oracle PeopleSoft is also known for human resource (HR) application, whereas SAP is well known for production and supply chain management (SCM) applications.

In the recent years these large vendors have tried to diversify their systems by expanding their application modules through acquisition of other software companies. Oracle now owns the PeopleSoft ERP along with its own e-Business Suite. Nonetheless, they still focus on certain industries and are known for applications in certain functional areas of business. It is therefore important for business evaluating ERPs to pay close attention to these criteria before selecting the software.

IT Infrastructure Criterion

The IT infrastructure criterion is important because a company not having the resources to invest in new infrastructure may want to acquire an ERP application that will work on an existing platform. Some ERP vendors have structured their applications to work on specific platforms with specific database and third-party software. In that case, having the vendor install a "sandbox" application for demonstration purposes on the company's existing infrastructure can be very helpful.

Resource

The resource question is the most important issue that has to be resolved before moving to the next phase of ERP implementation. That is, what organization resources will be needed to implement and support the product? Senior management must be involved in making this decision. If they are not committed to the project for the long run, both in terms of resources and time commitment, the implementation is doomed for disaster. The majority of ERP vendor relationships with organizations are long term. Vendors are constantly upgrading or releasing new application requiring a business to remain close to what is happening with the system. This requires long-term resource commitment from the organization.

ERP System Research Table

Item	Description
User Base	How many companies are using the system and for what purposes. Identify competitors using the system
Financial Position	If publicly traded, this information is fairly straightforward. If not, the ERP vendor information may not be available until such time later in the purchase process. It should include sales revenues, profits, growth, research and development, and any outstanding debt.
Implementation Issues and Support	This can be accomplished through calls to companies using the software or IT researching companies that survey and collect this type of information on a regular basis. Be sure information is current. What may have happened three or four years ago may not be accurate
HW/SW Infrastructure Fit and Scalability	ERP vendors usually support more than one platform. Identifying and documenting the platforms will provide a better understanding of the scalability of the system and ultimate fit with the company's direction.

Vendor Direction	This information should be tracked through the vendor history of change and upgrades to the system along with a statement of direction from each vendor. The ability to implement a stated direction is important; hence, knowing the history is critical to understanding the hype versus the reality
Currency of Technology	Like legacy systems, a vendor's software is often written in older technology. In some sense every vendor goes through this because technology is changing rapidly, but the ability to migrate to new technology must be understood and documented during the research process.
Release Strategy	How often these releases to the system, and what are is included in the releases? Are fixes timely and minor upgrades included periodically? What is the timing for major releases, and re the defined upgrade paths? Are there vendor costs related to upgrades?
Development and Maintenance Staff	This is an area that sometimes requires some exploration. It is difficult to compare apples to apples because the "size" of the ERP vendor system will have an effect on the development and maintenance staff. One should really look for a disproportionate number within an ERP vendor and across ERP vendors.
System Update Process	This is a key are. It helps to understand how much your company's direction and long-term needs will be met by the vendor. A company's involvement in further defining the functional direction of an ERP system is important to understand and document in the vendor research process.

ASSESSMENT

The faculty may conduct their assessment based on the contents of the lesson in different platforms like Google Forms, Schoology, and the like.

REFERENCES

1. Erl, T. (2004). Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services (1st Edition). Prentice Hall. ISBN-13: 9780131428980
2. Gold-Bernstein, B., and Ruh, W. (2004). Enterprise Integration: The Essential Guide to Integration Solutions (1st Edition). Addison-Wesley Professional. ISBN-19: 9780321223906
3. Hohpe, G., and Woolf, B. (2003). Enterprise Integration Patterns: Designing Building and Deploying Messaging Solutions (1st Edition). Addison-Wesley Professional. ISBN-13: 9780321200686
4. https://monday.com/blog/teamwork/break-functional-silos-organization/
5. Lehtonen, Karri (2018). What is System Integration?. YourEdi. Retrieved from https://www.youredi.com/blog/what-is-system-integration
6. Markgaf, B., (2019). Importance of Information Systems in an Organization. Small Business. Retrieved from https://smallbusiness.chron.com/importance-information-systems-organization-69529.html
7. Motiwalla, L., and Thompson, J (2012). Enterprise Systems for Management. Prentice Hall. ISBN-13. 9780132145763
8. Silva, A. C., & Loureiro, G. (2011). *System integration issues - Causes, consequences & mitigations. 2011 IEEE International Conference on Industrial Engineering and Engineering Management.* doi:10.1109/ieem.2011.6118134
9. Wagner, B., and Monk, E. (2009). Concepts in Enterprise Resource Planning (2nd Edition). Cengage Learning. ISBN-13: 9780201844566