3 From Strategies to Solutions – A Planning Methodology

3.1 Introduction

Companies around the globe are grappling with the wide range of business and technical issues that are encountered when you try to transition from strategies to working solutions. Figure 3-1 shows the main choices – in reality, a mixture of these choices is used. This chapter presents a methodology "pattern" that can be possibly used to bring order to chaos. The methodology can be customized to build a large number of solutions by concentrating on engineering of new applications and re-engineering of existing (including legacy) applications at an enterprise level. There is no "one size fits all" approach but certain generic activities take place when you try to tie your e-business strategies to developing CRM, ERP, data warehouse, online purchasing, and/or Web-based collaborative applications. In particular, you should not have to rethink the entire process for the aforementioned situations. Our goal is to expand on the activities outlined in Chapter 1 of this Module and answer the following questions:

- What are the key activities of a generalized solutions methodology (see Section 3.3)?
- How does IS planning, in particular at application level, fit into this methodology (see Section 3.4)?
- How can solution architectures be developed to put the pieces together that can be delivered to the users (see Section 3.4.4)?

- What are the examples and case studies to illustrate how the solutions are actually build (see Section 3.6)?

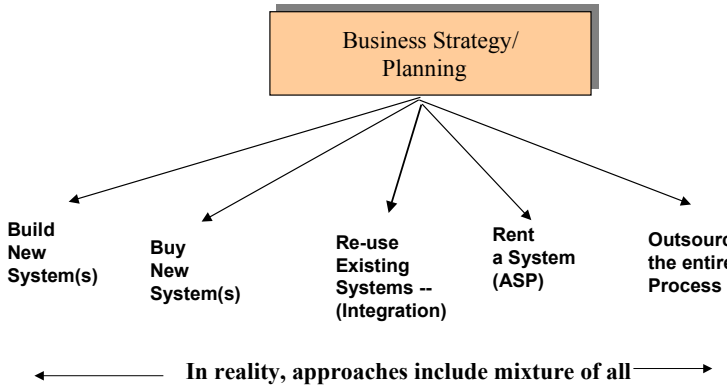


Figure 3-1: Options to Translate Strategies to Working Solutions

Key Points

- An IS engineering/re-engineering methodology pattern consists of successive iterations, refinements and expansions of four broad activities: analysis, solution architectures, implementations, and deployment/support activities.
- Overall planning should always be the first iteration of the process and should concentrate on business opportunity analysis and risk analysis to determine the best strategy.
- Strategic Analysis is one of the first activities in IS planning and can be driven by several initiatives such as business process re-engineering, mergers/acquisitions, new services, and paradigm shifts.
- Applications that support the strategic analysis must be conducted at enetrprise level.
- Decisions of building, buying, renting, re-use, and outsourcing need to be visited several times in the planning process.
- Solution architectures, based on business components, can handle most IS engineering/re-engineering issues and should follow a thorough enterprise application analysis. The solution may involve outsourcing through ASPs.
- Appropriate IT infrastructure should be selected carefully to support the solution architectures. If ASPs are used, then the infrastructure must support access to the ASP.
- Cost/Benefit Analysis should play a key role in making various decisions in the planning process.

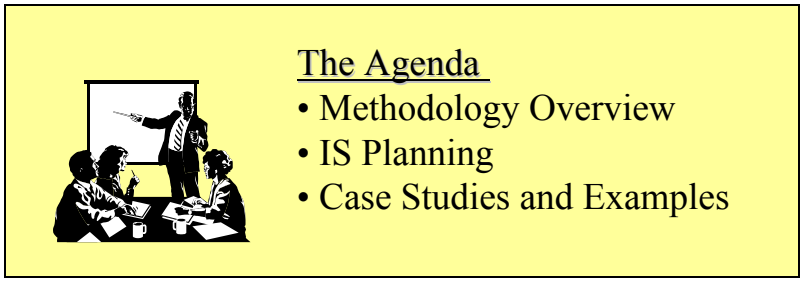
3.2 Case Study: XYZCorp Embarks on Building Solutions

Based on the previous projects, several new applications will be developed and many of the existing applications will be re-engineered at XYZCorp. At present, each application engineering/re-engineering is considered as a unique case. It is felt that an overall methodology is needed for engineering of new applications and re-engineering of existing (mostly legacy) applications. In other words, how should the corporation approach the challenges involved in building new Web-based applications and establish suitable strategies for dealing with its embedded base of mainframe-based legacy applications?

As we know, XYZCorp has embarked on a major initiative to extend and integrate the applications that support the business processes (payroll, accounts receivable/accounts payable, order processing, marketing information systems, and computerized checkout system), engineering processes (computer-aided design,

computer-aided engineering, computer-aided process planning), and manufacturing processes (material requirement planning, production scheduling and flexible manufacturing systems). As much as possible, the company wants to use "just in time" assembly process to minimize on-hand inventory. The company needs to develop enterprise applications that will support the company goals. The emphasis of enterprise application planning is not to develop internal designs of applications but to identify the key applications that fit into an overall solution architecture for business success (i.e., the delays between applications must be minimized to assure just in time processes).

The company is in a time crunch and has authorized a two week initial planning project. The management has been talking to a consultant (a good guy who writes very good books!) and has agreed to use the methodology template discussed in this chapter.



3.3 Building Working Solutions – An Overall Approach

3.3.1 Overview

A working IS solution may involve a mixture of: a) development of new applications, b) strategies to deal with existing (in many cases, legacy) applications, c) enhancement of the enabling IT infrastructure, and d) outsourcing decisions. The goal is to translate the business strategies to engineering/re-engineering of applications and IT services that are delivered to the users. Figure 3-2 shows a high level view of IS solutions development that was first introduced in the Overview Module and later in Chapter 1 of this Module. It can be seen that this is a non-trivial activity that involves changing business needs, high user expectations, a very large number of enabling technology options (i.e., large number of middleware and networking options), an ever increasing list of tools, a significant embedded base of existing infrastructure and applications that may partially satisfy your needs, and an almost unlimited number of off-the-shelf packages and applications.

A methodology is needed to systematically walk through various decisions. Before proceeding with details, we should acknowledge that formal methodologies have had mixed results [Inmon 1993, Mowbray 1995]. The appeal of a methodology is that it directs the developers down a reasonable path with pointers for what to do, in what order to do it, what to produce, and what to expect as inputs. However, many methodologies fail because of their linear flow of activities, rigidity in prescribed set of activities, and emphasis on diagramming tools. Figure 3-3 illustrates a high level methodology that refines Figure 3-2 and views IS engineering/re-engineering for e-business in terms of analysis, architectures, and deployment/support[1]. Our interest is in the following broad activities:

- **Enterprise IS planning**. What applications will be needed to support the business needs? If applications are not needed, can an upgrade of the IT infrastructure (e.g., a faster and more reliable network) do the job. Given a set of applications, some of these applications will be outsourced, some will be built or bought, while others will be re-used (see Section 3.4).

[1] Although this methodology is oriented towards e-business, it can be used for any modern information systems.

- **Application engineering**. How the new applications can be architected and developed efficiently by using the latest technologies (see the Module "Architectures").
- **Application re-engineering**. How the existing applications can be dealt with (i.e., how new applications can be integrated with existing applications, and when/how/if the existing applications should be transitioned). See the Module "Integration".
- **Deployment (IT Infrastructure engineering/re-engineering).** What type of IT infrastructure, especially the network and the middleware services, will be needed to build and run the new and existing applications? The needed IT infrastructure may need to be upgraded or completely overhauled. These issues are discussed in other modules (""Networks", "Middleware", and "Platforms").
- **Management and support.** How these applications and the needed IT infrastructure can be managed and supported. These issues are discussed in the Management Module.

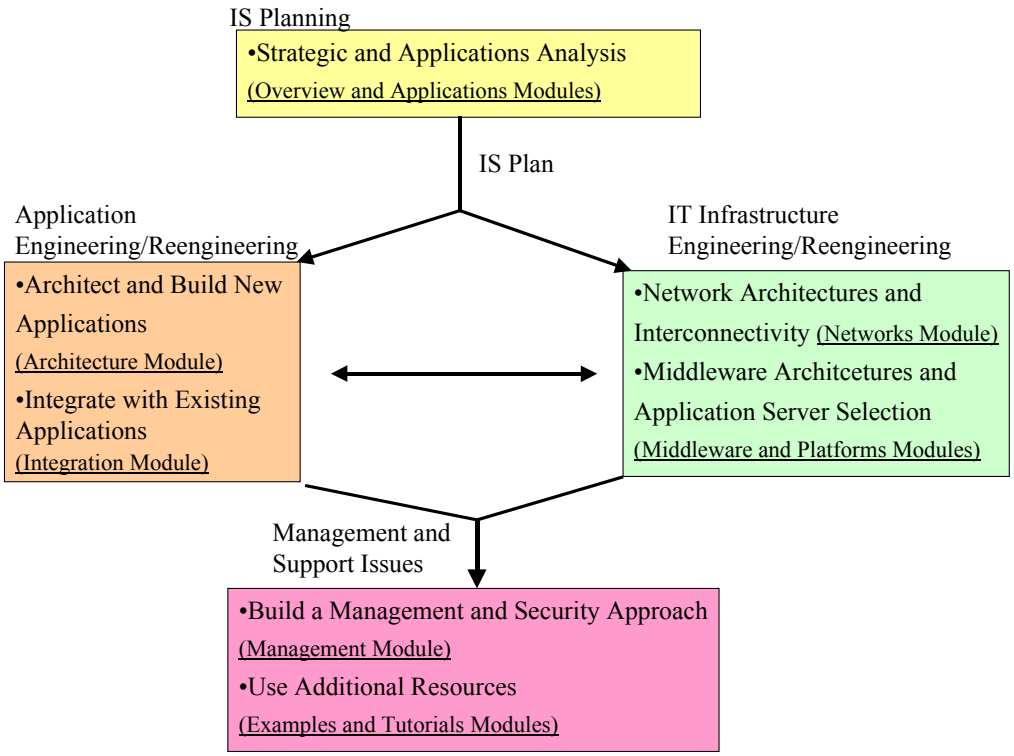


Figure 3-2: How to Build Working Solutions

The underlying idea of this methodology is that after planning, existing and new applications are consolidated into a solution architecture to meet business needs. For example, suppose you want to provide online-purchasing over the web. Then you may need to develop or purchase a new web application that displays catalog information on the web and allows a user to place orders through the web. However, this application needs to interface with order processing, inventory control, shopping carts, and payment systems that typically exist in an organization. In that case, you will need to re-engineer and integrate the existing inventory control and order processing applications with the new web application. A common example of application re-engineering at present is the melding of the latest Web, distributed objects and mobile computing technologies with the existing applications and databases.

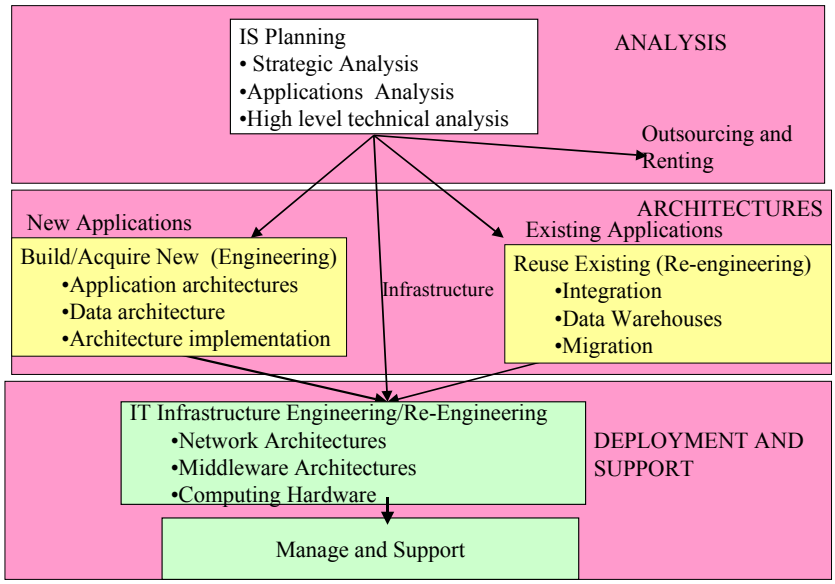


Figure 3-3: General Approach Building Solutions

3.3.2 An Iterative View of Solution Methodology

Although building solutions can be viewed as a sequence of activities, in reality this effort is a highly iterative process. Figure 3-4 shows a non-sequential "methodology pattern" that can be customized for specific cases. This pattern, briefly discussed in Chapter 1 of the Overview Module, represents a template that can be customized for specific cases. The key points of this methodology pattern are that translation of strategies to working solutions is an intensely iterative process and all iterations are based on the refinement and expansion of the following core activities:

- Analysis,
- Solution architectures,
- Implementations, and
- Deployment/support.

The methodology pattern suggests that planning, architectures, first release and subsequent releases are in fact iterations (not separate phases) in which each one of these activities is performed at a different level of detail. An important observation is that some activities are more extensive than others in each iteration (represented by the width of activity traversed in each iteration in Figure 3-4). For example, the first iteration requires extensive analysis and some high level architectures – the implementation and deployment/support activities are minimal. This is because the first iteration emphasizes business opportunity analysis and assessment of technical feasibility through architectural evaluations. In my own experience of working on large scale systems, I have seen first iteration to consist of 1 to 2 months of analysis, 1 to 2 weeks of architectures, and 1 to 2 days of discussion on implementation and deployment issues. As we proceed to later iterations, the time spent in analysis is reduced but increases in the architecture, implementation and deployment/support activities. Naturally, later releases of a system are heavily implementation and deployment/support centric. For example, most of the time on release 22 of a product is spent on implementation and deployment activities, unless major enhancements are planned. For a major release or enhancement of an application, you may re-start the entire process with first iteration.

Let us briefly review these iterations before getting into details:

First Iteration (Planning Focus). This iteration, discussed in Section 3.4, essentially concentrates on overall planning with quick analysis of architectural and implementation issues. It identifies the business

drivers, key stakeholders/funding sources, high level requirements, and costs/benefits based on a high level review of proposed solution architectures, implementation considerations, and deployment/support issues raised. The main purpose of this iteration, in our case, is to answer the following question: Given the high level requirements and business drivers, should new application(s) be built (i.e., engineered) from scratch, should existing application(s) be re-engineered, or should a mixture of engineering/re-engineering approaches (e.g., build a small portion and interface it with existing) be used?

Second Iteration (Solution Architecture Focus). In the second iteration, discussed in Modules "Architectures" and "Integration", you typically establish the blueprint that ties the applications (new plus existing) with the enabling infrastructure. The architectures, implementations, and deployment/support aspects of enterprise applications are studied and prototyped to gain insights into feasibility and effort sizing. In particular, the application engineering/re-engineering decision made in planning is revised, if needed. Although the specific steps and tools in this iteration depend on the type of applications, we suggest that a business component approach should be used as an overall framework.

Production (Third and later) Iterations. In the next iterations, the first and future releases of the application are built and deployed. The specific steps and tools in these iterations depend on the type of applications and on the type of application engineering/re-engineering approaches. Other modules of this book (e.g., "Management", "Networks", "Middleware") will explore these iterations in more detail.

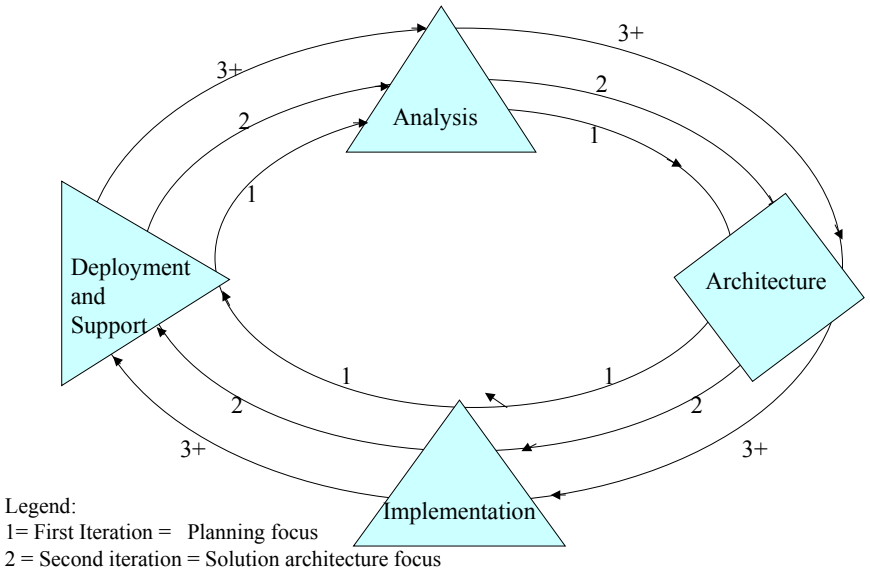


Figure 3-4: An Iterative Methodology Pattern

Table 3-1 illustrates the analysis, architectures, deployment, and support activities in the planning, architectures, and implementation iterations.

This general methodology pattern is based on my own experience in three different areas: a) more than a decade of real life development an management experience in a wide range of application engineering/re-engineering projects, b) development and teaching of numerous information system analysis and design courses, and c) extensive examination of the extant literature in methodologies such as the following:

- IS planning methodologies [Blodijk 1987, Zacman 1982, IBM 1978, Rockart 1982, Keen 1991, Luftman 1993].
- Object oriented and component-based analysis and design methodologies [Herzum 2000, Booch 1994, Rumbaugh 1994, Jacobson 1992, Nerson 1995].

- Special design methodologies for application areas such as data warehouses [Inmon 1994, Inmon 1995], real-time systems [Fuhrt 1995, Gemmel 1995], hypermedia applications [Isakowitz 1995], and rapid prototyping [Gordon 1995].
- Methodologies for re-engineering of legacy systems [Brodie 1995, Sneed 1995, Desfray 1996].
- Rapid prototyping approaches, such as the ones reported by [Gordon 1995] that are based on review of almost 40 case studies.

Table 3-1: Iterative Methodology

	Analysis	**Architecture**	**Deployment**	**Management and support**
Planning Iteration	Detailed Business analysis	High level solution architecture	High level IT infrastructure	High level staffing considerations
Architecture Iteration	Detailed requirement definition	Detailed solution architecture (application architecture)	Detailed IT infrastructure analysis Build a prototype	Initiate pilot projects
Implementation iteration(s)	Detailed requirement definitions	Build a run-time architecture and detailed design	Build and test the system	Assign staff, provide customer support, initiate change management, etc

Guidelines For Successful Solutions Building

1. Be clear about what you are trying to accomplish and what are the business drivers
2. Involve business and IT groups early (planning stage) in the process
3. Take every opportunity to improve the business processes and keep technology innovation as a lower level objective
4. Keep projects short so that you can measure success/failure
5. Base transition strategies on investments that are already necessary
6. Get sponsorship/buy-in from senior management
7. Do not forget/underestimate the people issues
8. Over-invest in end user support
9. Use information technologies as enablers and not as drivers of change
10. Pay special attention to reuse of existing, including legacy, resources

Time To Take a Break

- ✓ Methodology Overview
- IS Planning
- Case Studies and Examples

Suggested Review Questions Before Proceeding

- What is a methodology and why is it needed? Why the existing methodologies have had mixed results?
- What are the main characteristics of the methodology presented in this section? What appear to be the strengths and weaknesses of this methodology?
- List some examples of methodologies that you are familiar with

3.4 Enterprise IS Planning – The First Iteration

3.4.1 Overview – An Enterprise View

To illustrate the key ideas of enterprise IS planning, we will use the XYZCorp Case Study described in Section 3.2 in these discussions.

In general, the purpose of IS planning is to align the engineering/re-engineering of applications and the IT infrastructure with business goals. Several methodologies have been developed for this planning process based on the "classical" information systems planning methodologies such as the following:

- IBM's Business Systems Planning [IBM 1978, Zachman 1982]
- Rockart's Critical Success Factors [Rockart 1982]
- Nolan's Stage Model [Nolan 1973]

Extensions of these methodologies have been reported such as the BSP extensions by Blodgik and Blodgik [Blodijk 1988], and homegrown methodologies [Meyer 1996, Hulfnagel 1987]. Additional planning methodologies are discussed by [McNurlin 2002, Boar 2001]. Of particular interest are the business process re-engineering (BPR) approaches that leverage IT for business [Hammer 2001, Harrington 1997, Ward 1996, Davenport 1998, Davidson 1993, Henderson 1990, Keen 1993, Luftman 1996, Venkatraman 1984] and the Internet related strategies [Cronin 1996]. Extensive discussion of BPR and planning methodologies is beyond the scope of this chapter. We will revisit this topic in the Management Module. However, the following key BPR principle will guide our discussion:

> **Guideline**: use information technology to improve the performance of a business and cut costs by redesigning work and business processes from the ground up instead of simply automating existing tasks.

This principle should be the driver as you go through the activities of the planning iteration (i.e., analysis, architectural trade-offs, implementation issues, and deployment/support considerations). The sidebar "Application Planning Checklist" casts these activities into planning steps and Figure 3-5 shows the enterprise application planning steps (strategic analysis, application analysis, quick solution architecture analysis, IT infrastructure assessment, and cost/risk analysis) as a procedure. We will discuss these steps in the following subsections.

It is appropriate to note here that the enterprise application planning process presented in Figure 3-5 represents a comprehensive approach. In reality, only parts of these steps may be conducted by a company under the planning heading (in many cases, the application planning is conducted in one meeting that lasts

only about two hours). Our objective is to present the key decisions that need to be made early (of course, many of these decisions are reviewed and modified in later stages).

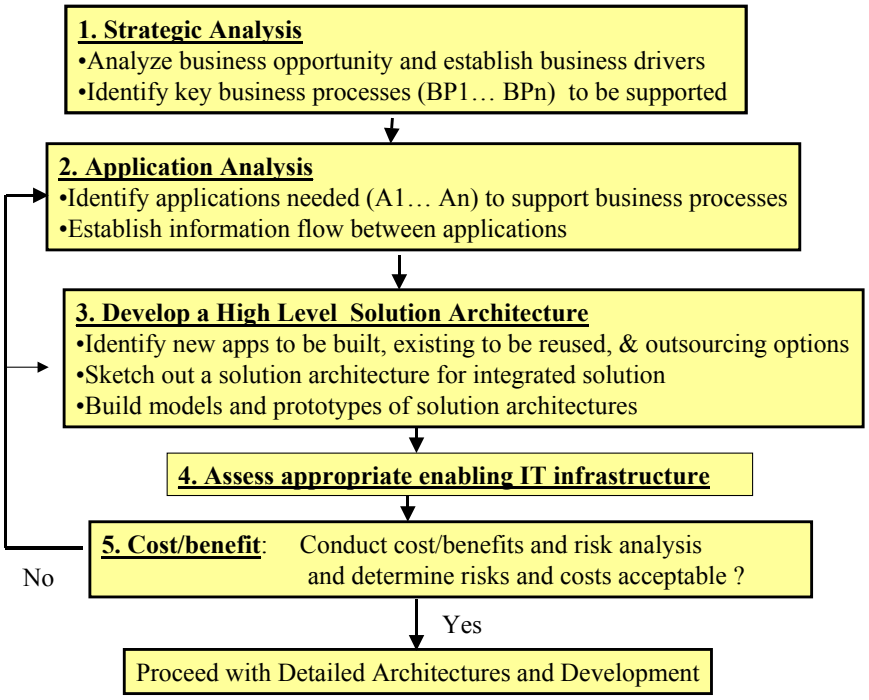


Figure 3-5: Overall Planning Procedure

Based on the results of the planning iteration, and any other relevant information, an initial important decision is made: given a business opportunity, should a new application be developed (application engineering), should existing applications be re-engineered, or should a mixture of engineering/re-engineering be used? This decision may be revised in later iterations as we develop a better understanding of the problem and the various trade-offs. However, this decision will help us to outline a rough project plan that specifies:

- Goals and objectives of the effort
- Expected benefits and risks
- Main steps, the deliverables, and time lines
- Key staff assignments, i.e., the roles and responsibilities of internal IS staff as well as external consultants and systems integrators

This plan is used to define the next iterations and to decide how much effort will be spent in engineering versus re-engineering. Keep in mind that planning does not need to take a long time, it just needs to be done. In my own experience, I have found that preparation of a response to each RFQ (request for quotation) essentially goes through the planning steps.

Application Planning Checklist

Analysis

Step 1. Strategic Analysis:

a) Establish business drivers and analyze business opportunity

b) Identify key business processes

Step 2. Application Analysis:

a) Identify applications needed to support business processes

b) Establish information flow between applications

Architecture

Step 3. High Level Solution Architecture

a) Identify new apps to be built, existing to be reused, & outsourcing options

b) Sketch out a solution architecture for integrated solution

c) Build models and prototypes of solution architectures

Implementation

Step 4. IT Infrastructure

a) Assess IT infrastructure needed

b) Choose the most appropriate IT infrastructure

Deployment and support

Step 5. Cost and Risks

a) Investigate implementation issues (i.e., skill, resources, time and money needed to implement)

b) Estimate effort needed to deploy and support

c) Outline costs (pitfalls) and benefits (promises)

d) Evaluate if costs are worth the benefits

3.4.2 Step 1: Strategic Analysis

3.4.2.1 Identify Business Drivers and Establish High Level Requirements

Application engineering/re-engineering at an enterprise level is an expensive and risky undertaking. The purpose of this step is to understand and analyze the basis and business motivation for this effort before fully launching it. The following broad categories represent many business drivers:

- Business process re-engineering (for example, BPR of healthcare industries has initiated many new applications to be developed and many legacy applications to be re-engineered). See Section 3.4.2.2 for a brief discussion of BPR.
- New services or business opportunities (for example, a large number of companies are building new Web-based applications to take advantage of the new opportunities created due to the explosion of Internet).
- Gain and maintain competitive edge (for example, many companies have investigated e-business to gain competitive edge in this rapidly growing area [Adam 1996].
- Align IT with business (for example, many companies are reevaluating their IT services and aligning them with the business [Luftman 1996]).

It is a good idea to take a strategic view (see the discussion on strategies in Chapter 1 of this module). In particular, you should conduct business strategy analysis based on the products (existing, new) and customers (existing, new). The basic idea, presented in Figure 3-6., was discussed in Chapter 1.

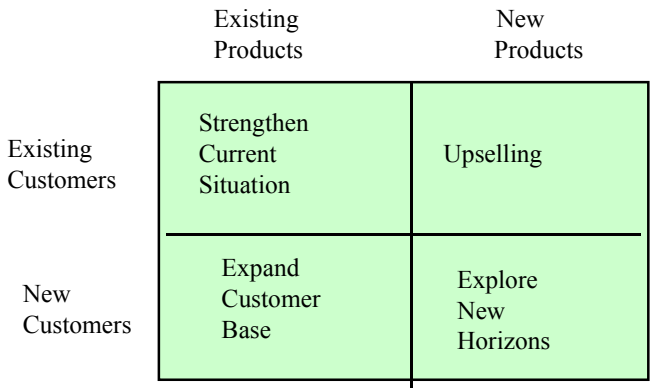


Figure 3-6: Establishing Strategies Based on Products and Customers

In addition to business drivers, you need to develop an understanding of the business requirements that drive the technical requirements (Figure 3-7). In particular, you need to identify the organization and the processes that will be served by the applications and develop an "end-user" profile that shows the characteristics of the user community (e.g., decision support versus operational support users, novices versus "power users"). You also need to identify the main customers and stakeholders who will benefit from the proposed applications. In addition, these requirements should capture:

- Business goals in terms of short-term and long-term goals of the FMO (Future Mode of Operation)
- The main stakeholders who will benefit from the effort
- The business problems with PMO (Present Mode of Operation)
- Funding sources and limits
- Time frame

Keep in mind that these requirements are very high level (also known as "thin" requirements) and are meant to help you assess readiness for building solutions. These requirements should be between 20 to 30 statements and must not include screen layouts and other programming details (all that comes later).

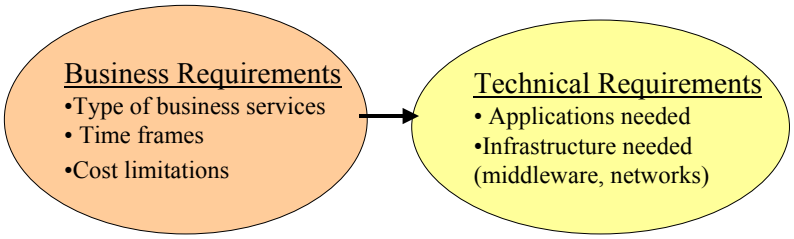


Figure 3-7: Business Requirements Drive The Technical Requirements

After identifying and clearly understanding the business drivers and high level requirements, you should analyze the opportunity by posing questions such as the following:

- Are we doing the right things (i.e., can the information technologies help us to reshape our future, are our information services providing us competitive edge, will they help us to administer our business objectives, will they enable us to adapt to unexpected changes)?
- Are we doing things right (i.e., are we minimizing the unit costs for each information service, is the value of information service worth the cost)?
- Are we heading in the right direction (i.e., are we listening to our customers and paying attention to their information services needs, are we aware of the changing market and government conditions)?
- What are the services we are good at and what are the services we need to improve/ discontinue (are we sinking too much money into older technologies, how confident are we about the new technologies)?

As a result of these questions, an overall strategy can be established which defines a short range and long range vision for the applications and services under consideration. It is the responsibility of management to develop and present a clearly defined and doable vision. Unfounded visions and too many visions ("vision of the day") cause serious problems. Announcing a vision is not enough; continued and visible support from management is essential for the success of a strategy [Henderson 1990, Luftman 1996]. One possible approach to achieve these objectives is to get the information technology experts involved early in the planning process. These experts can help in making the vision realistic and then work as agents and supporters during the implementation stages.

3.4.2.2 Business Process Reengineering and Business Workflows – A Quick Overview

Many IS planning processes are triggered by the business process re-engineering (BPR) efforts. A very brief overview is presented here for completeness because a great deal of literature is available on BPR (see the sidebar "Business Process Re-engineering Sources of Information"). The main idea of BPR is to determine what business processes (BPs) are needed to conduct a business activity, why they are needed, and to redesign the BPs from the ground up instead of simply automating existing tasks. The technology (IT) plays a crucial role in BPR to improve the performance of a business and cut costs. For example, while waiting in line at the grocery store, you can appreciate the need for the check-out process improvement so that you can go home without having to wait in line and browse through the gossip magazines (I do that!).

Business process is "a set of logically related tasks performed to achieve a defined business outcome" [Davenport & Short 1990]. The business processes in this simple case are the activities that you and the store personnel do to complete the transaction. Examples of other business processes include purchasing books online, ordering clothes from mail order companies, requesting new services from your ISP, developing new products and services, administering the delivery of furniture to customers, initiating a new line of software products, constructing a new office building, etc. The business processes are typically pictured as a set of squares with some feedbacks as shown below.

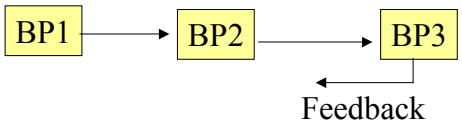


Figure 3-8: Simple Business Process (BP) Flow

Improving business processes is crucial for businesses to stay competitive in today's marketplace. Companies are continually forced to improve their business processes because the customers demand better products and services. While processes can be improved continuously (called business process improvement), BPR relies on a radical change by assuming that the current process is irrelevant and does not work. BPR forces designers to start over with a clean slate. This perspective enables the designers to disassociate themselves from today's process and focus on a new process. The main idea is to create a vision for the future and design new business processes for the company to compete in the future. Given a the definition of the FMO (future method of operation), you can then create a plan of action based on the gap between FMO and PMO (present method of operation). Based on this gap, you determine the technologies and business activities to fill the gap and then implement your solution approach.

Here are some examples of BPRs:

- Improving services. Several companies have significantly reduced product development time through BPRs. For example, GM has cut down the time to build new cars from 4 years in the mid 1980s to a few months in 2002. Similarly, several companies have reduced waiting time for checkouts (reducing drive through time by McDonalds by having two windows is an example). Reducing customer complaints (elevators)
- Mergers/acquisitions. Several BPR opportunities arise due to mergers and acquisitions. These may involve consolidation of support centers and elimination of redundancies in the merged organization. In addition, different purchasing approval procedures due to different approval rules and different order processing procedures with or without back-order support may need to be consolidated.

- New organizational goals. Companies decide to double sales, reach new customers, and improve customer services. For example, Guess Jeans decided to double its sales in 3 years by using the Internet and the Canadian Children Hospital Network decided to improve services through a new network that tied the participating hospitals together.

Workflows between business processes is the main approach used in BPR. Simply stated, workflow is concerned with automatically routing the work from one process to the next --- it defines the operations that must be performed between the origin of work to its completion. Workflow, in essence, represents the "business choreography" between business entities, i.e., the steps to complete a business process and the rules which decide what steps should be performed when. Consider, for example, the process of getting a purchase order (PO) approved for an equipment that costs more than $100K. In this case, the PO will be shuffled between various managers for different levels of approvals. Figure 3-9 shows a simplified view of PO approval process. In some cases, the approvers may be on vacation and might have delegated the approving authority to someone else.

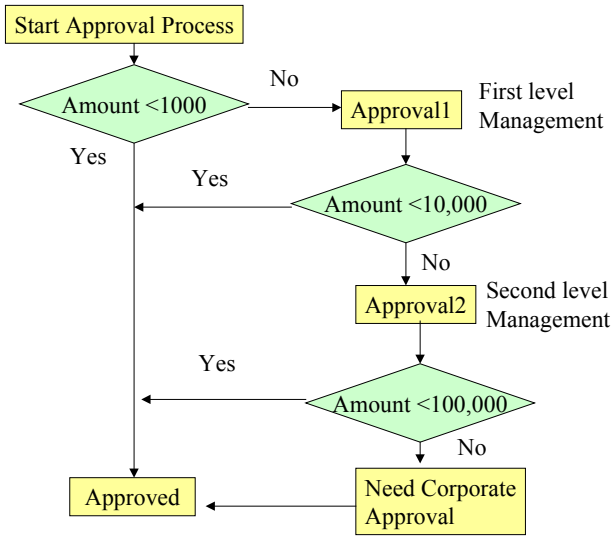


Figure 3-9: A Simplified View of Workflow for Loan Processing

Workflow management system is the middleware that enables improvements in business processes through automated workflow. Specifically, the workflow middleware performs the following tasks:

- It facilitates modeling of workflow process and its constituent activities
- It routes work in the sequence as specified in the process model
- It provides access to the data and documents required by the individual work performers
- It tracks all aspects of process execution

These functions, shown in Figure 3-10 are the foundation of workflow management. Many workflow managers are commercially available from suppliers such as IBM, HP, and others. See Chapter 3 of the Platform Module of this book for a detailed discussion of workflow management systems. For additional information, see the sidebar “Workflow Management Sources of Information“.

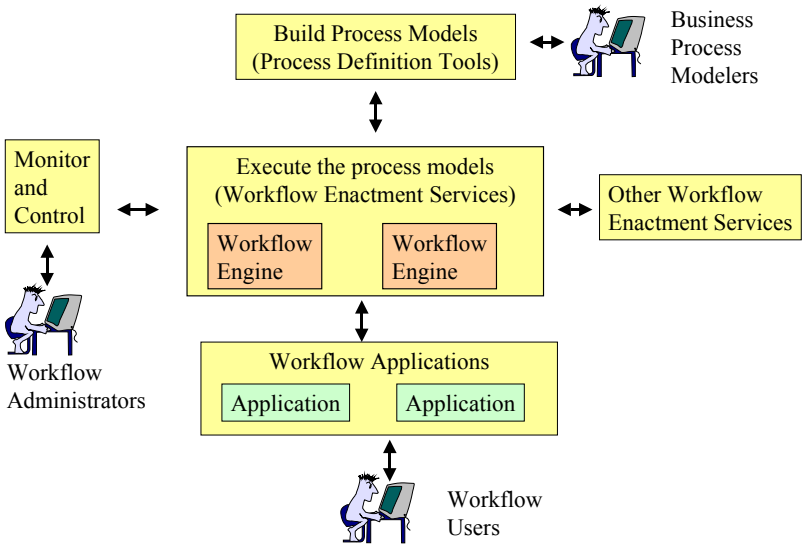


Figure 3-10: Conceptual Model of Workflow Management

Business Process Re-engineering Sources of Information

Websites

- http://www.brint.com/papers/bpr.htm -- BPR (redesign) overview paper
- http://www.prosci.com/ -- BPR online learning center – has reports, tutorials, books (cost)
- http://www.brint.com/BPR.htm -- many articles and tutorials
- http://www.waria.com/ -- WARIA -- Workflow And Reengineering International Association (publish handbook)

Books

- Hammer, M. and Champy, J., "Reengineering the Corporation: A Manifesto for Business Revolution", Harper Collins Publishers, 1993.
- Jacka, I. and Keller, J: "Business Process Mapping: Improving Customer Satisfaction", 2001
- Carr, D. and Johansson, H., "What Works And What Doesn't In The Reengineering Process", McGraw-Hill, 1995

Workflow Management Sources of Information

- Workflow Management Coalition (WfMC) web site (www.wfmc.org)
- Workflow And Reengineering International Association (WARIA) Web Site (http://www.waria.com)
- The Workflow Handbook 2001, Published by WARIA in association with WfMC, Edited by Layna Fischer, Published OCTOBER 2000
- Workflow Comparative Study, 2001 Edition, published by WARIA
- Information Systems Frontiers Journal, Special Issue on Workflow Automation, Vol. 3, No. 3, Sept 2001, Kluwer Publications

3.4.3 Step 2: Enterprise Application Analysis

3.4.3.1 Identify applications needed to support business processes

The main idea is to identify the applications A1, A2… An that are needed to support the business processes BP1… BPn identified in strategic analysis. How to identify the complete set of business processes (BPs)? Here is a suggested approach:

- List all BPs that support the B2C, B2D, B2E, and other business interactions
- To make sure that you don't get thousands of BPs that cover minute daily activities, keep your focus at enterprise level, activities that are vital to your business. According to Rockart's Critical Success Factors [Rockart 1982] methodology, the focus should be on those processes that are *critical to the success of your business*.
- Reduce duplication by clustering similar BPs into one. For example, the same BP is used for customers as well as business partners , then it is better not to have different BPs.
- It is a good idea to question, eliminate, and restructure business processes to improve organizational efficiency. This is the whole idea of business process re-engineering.

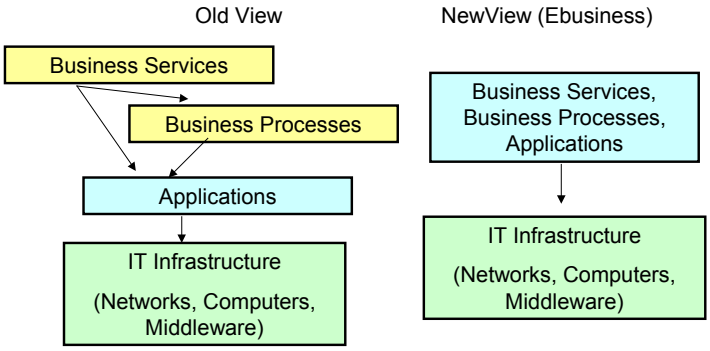
In reality, one or many applications may be needed to support a given business process. and a given business process may need multiple applications. For example, a customer information system may support many business processes such as purchasing, marketing, and payment. Similarly, purchasing business process needs support of many applications such as order processing, inventory management, shipping/receiving, and payment. The result of this step is a table that may resemble Table 3-2. Tables of these nature can be extremely revealing. For example, the following table indicates the following:

- Application 2 does not support any business processes. This may mean that an application was developed without any business reason or it supports an outdated business processes
- Business process 2 is not supported by any application. This may indicate that this business process can be directly supported by the IT infrastructure or that this BP is being ignored.
- Application 5 supports 3 BPs. Thus replacement/enhancement of this application should be done very carefully.

Table 3-2: Applications to Support Business Processes

	Business Process1	Business Process2	Business Process3	Business Process4
Application 1	x			
Application 2				
Application 3			x	
Application 4			x	x
Application 5		x	x	x

As discussed in Chapter 1, recall that applications are used to provide automated support for business services and processes. Although true technically, the differences are disappearing due to the increased use of automated services (e.g., self serve customer support) and automation of business processes (e.g., automated time reporting and approval systems). Consider, for example, a company that supports 30 business processes. In the "old days" of 1980s and before, a small portion, say 5 of these processes were automated (typically only payroll, accounts payable/receivable). But at present, due to the increased use of automation, most if not all of the 30 BPs could be automated. In that case, what exactly is the difference and why should we bother with the difference? The answer, is support -- applications support business services and processes and thus play the role of enablers (see Figure 3-11). Although these are beginning to look like a big "blob" due to interdependencies and the rise of completely digital companies such as Napster, it is a good idea to keep conceptual separation as much as possible. It also helps to keep business focus in mind.



Applications used to provide automated support for business services and processes. Difference disappearing due to automation. At present, the IT infrastructure supports applications, business processes, and services.

Figure 3-11: Business Services, Business Processes, and Applications (Old versus New View)

Business Services, Business Processes, and Applications

- Business Services: Directly offered to the customers (e.g., delivering food, selling books)
- Business Processes: procedures and methods needed to support the business services and other activities of an organization (e.g., printing menus)
- Applications: support (automated) the business services and processes. These are typically business aware. Examples are airline reservation systems, order processing systems, payrol, etc.
- IT Infrastructure: Support the applications. These are business unaware components (examples are networks, computers, middleware, etc).

3.4.3.2 Establish a Logical Application Architecture (Interdependencies and flows between applications)

As a result of the previous step, you will get a list of applications that will be needed to support your business. This list, graphically shown in Figure 3-12, could be refined further before proceeding. This list could, with minor modifications, represent the XYZCorp applications identified in the previous chapter. We should quickly establish interdependencies and flows between candidate applications. This is important because before making the buy/outsource/develop/reuse decisions, you need to understand how these applications interact with each other -- the cost of integration could be very high if systems from different suppliers that did not inter-operate with each other were thrown together. In fact, this is a reason for several failures.

The main questions to be considered in this analysis are: what information is exchanged between different applications, how frequently is the information exchanged, what is the volume of information exchanged, what are the performance, security, and availability requirements on information exchanges, etc.? The result of this analysis is a diagram that shows the information flows between various applications - known as ***logical application architecture.*** Figure 3-13 shows a possible logical application architecture for XYZCorp. This diagram is accompanied by a document that gives additional information (e.g., frequency, volume) about the information flow characteristics.

A variety of techniques are used to determine information flows. Sequence diagrams are one of the most effective means used. Figure 3-14 shows a sample sequence diagram that illustrates the information flow between a customer, a purchasing system and a payment system. For enterprise level modeling, sequence diagrams are developed for each business service that touches various applications. From these sequence diagrams, you can infer the interdependencies between various applications and also add traffic information (i.e., how many purchases are made each day, how many payments are made each day, etc.).

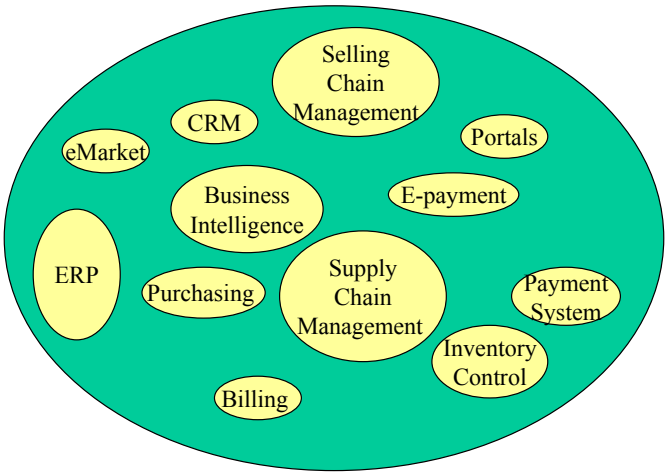


Figure 3-12: Example of Applications

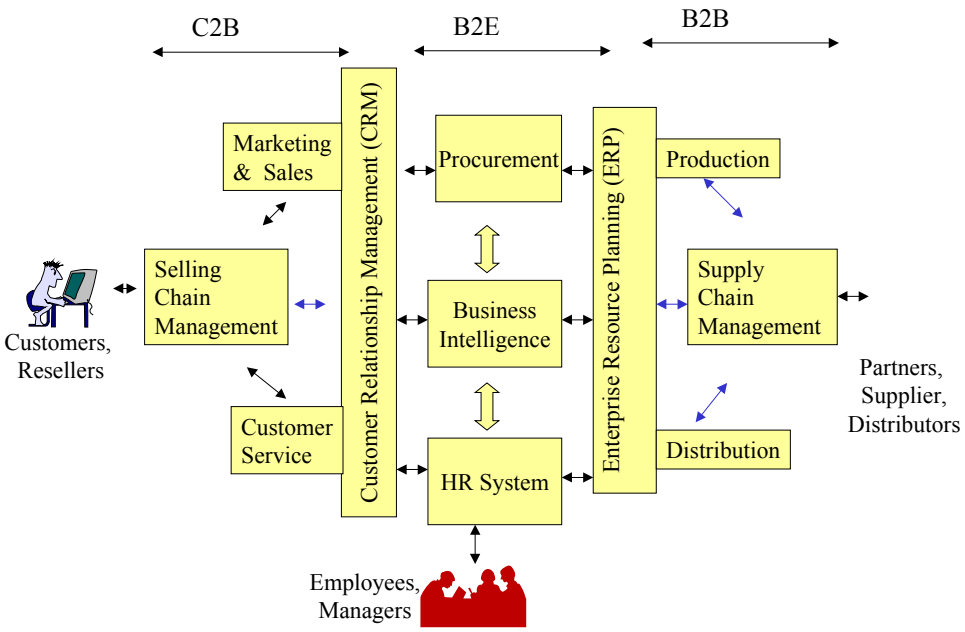


Figure 3-13: Example of High Level Information Flow (Logical Application Architecture) for XYZCorp

3.4.4 Step 3: Develop a Solution Architecture Sketch

Given an understanding of the business drivers and high level view of applications that satisfy the overall business strategy, the next challenge is to architect a solution approach that could satisfy these requirements. This step involves:

- Investigation of the outsource/buy/build/re-use options
- Development of a solution architecture sketch

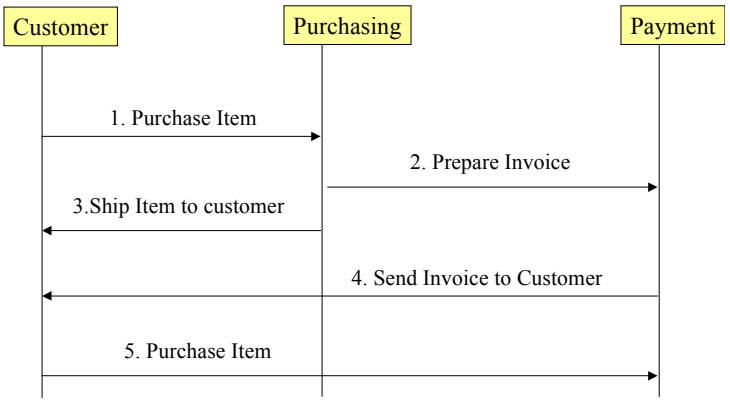


Figure 3-14: Sequence Diagram

Keep in mind that this step is just the first iteration into architectural considerations for planning purposes . The next iteration develops a detailed solution architecture that can be actually implemented and deployed. We will discuss the detailed solution architectures in the Architecture Module of this book.

3.4.4.1 Investigate Outsource/Buy/Build/Re-use Options

To develop a solution, you first need to answer questions such as the following:

- **Outsource**: Should I outsource part or all the applications I need?
- **Buy:** Should off-the-shelf packages be used to satisfy the requirements?
- **Build:** Should the needed applications be developed from scratch (i.e., new user interfaces, new application code, new databases)?
- **Reuse:** Should existing applications/databases be reused?

In reality, the overall solution is a mixture of these choices. In addition, lower level decisions, such as the following, may need to be made at the application software, user database, or user interface level:

- Should some of the outsourced applications access existing databases and applications (and how)?
- Should a new application software be developed/bought that uses existing databases?
- Should a new database be developed that is accessed from existing application software?

Should a new user interface (e.g., Web interface) be developed while utilizing the existing databases and application software? The answer depends on how new is the business undertaking, what already exists in your organization, what is available as a commercial-off-the-shelf package, what can be outsourced, and how flexible is the existing system.

Before discussing the process used in making the decisions, let us briefly show the *results* of the process. The results could be a refinement of the information flow model, also known as ***detailed logical application architecture.*** Figure 3-15 shows this architecture for XYZCorp. In this model, the ASPs have been identified. Additional information such as the applications that will be bought, developed and/or re-used can be also embedded in this diagram through color codes, etc. In general, we can analyze these decisions based on the following three factors:

- Databases. Are new databases needed or can the existing databases do the job? For example, several existing databases of legacy systems contain valuable information (e.g., customer information) that is needed by new applications.
- Business functions. Are new business functions needed or can the existing business functions be invoked and used to satisfy the requirements? For example, many existing applications have code that performs regularly used business functions such as generating bills, checking inventory levels and crediting/debiting accounts. The question is: can this code be reused?

User interfaces. Are new user interfaces needed or can the existing user interfaces do the job? The user interfaces is an area of considerable activity at present due to the popularity of Web browsers. In many cases, Web interfaces are being developed for existing applications even if the current user interfaces are quite

good . This is motivated by the desire of organizations to use Web browsers as the primary source for accessing all information.

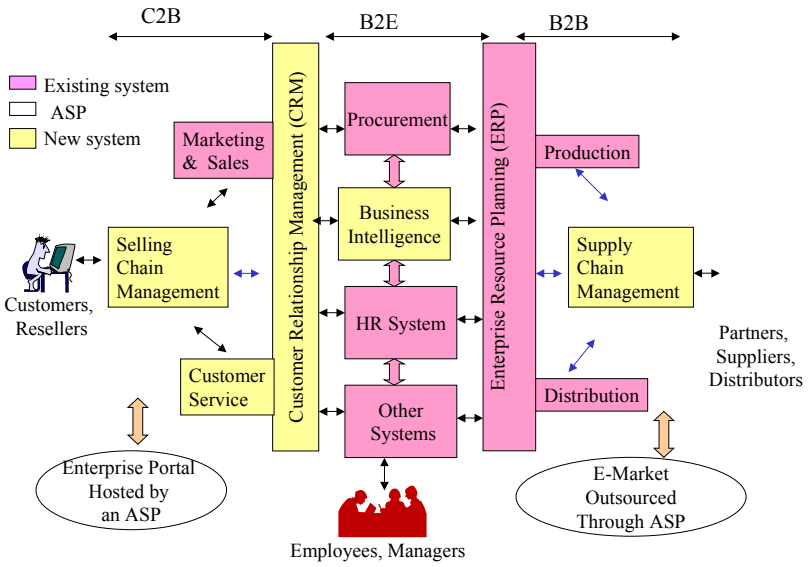


Figure 3-15: Refined Information Flow Model (Detailed Logical Application Architecture) for XYZCorp

Table 3-3 shows how these three factors can be used to systematically generate application engineering/re-engineering choices. In addition to this table, you need to consider the application interdependencies determined in the previous step. For example, you should not outsource an application that is highly interdependent on many in-house applications. Evaluation of these choices is a time consuming, yet an essential, aspect of contemporary application engineering/re-engineering practice. Broadly speaking, the choices are a mixture of outsourcing, purchasing off-the-shelf packages, developing new application components from scratch and interfacing of new components with the existing (including legacy) systems.

Table 3-3: Application Engineering/Re-engineering Approaches

Database Needed	Business Functions Needed	User Interface Needed	Potential Application Engineering/Re-engineering Choice
New	New	New	Outsource to service provider
New	New	New	Buy a new system
New	New	New	Develop a new system from scratch
Existing	New	New	Develop new application code and user interface (e.g., Web)
New	Existing	New	Develop a new user interface that invokes existing code
New	New	Existing	Develop new databases and application code
Existing	Existing	New	Develop new user interface (e.g. Web) to access existing (legacy) applications
Existing	New	Existing	Invoke new application functions from existing user interface
New	Existing	Existing	Develop a new database that is accessed from existing application code and user interfaces
Existing	Existing	Existing	Use existing application. May require some re-engineering if the existing application is too old.

Legend:

- Unshaded rows: need entirely new application (application engineering)
- Lightly shaded rows: need a combination of new and existing (application engineering and re-engineering)
- Dark shaded row: need to reuse existing applications only

Note that most real life situations require a combination of engineering/re-engineering.

Purchasing off-the-shelf applications is a viable solution when new databases and new functions are needed. This strategy, if feasible, is by far the most attractive. Consequently, it must be considered as the first choice. For example, if a human resource (HR) application is needed and the requirements seem to be satisfied by an off-the-shelf HR system from a vendor, then this package should be evaluated seriously. The key evaluation criteria are:

- Will this package satisfy the current as well as future requirements?
- Will this package scale well?
- Will this package interoperate with other systems?
- What is the current user experience base?
- Does it conform to the enterprise IT infrastructure standards (e.g., if the enterprise does not use Java, then a Java-based application should not be bought)?

Outsource to a Service Provider: Outsourcing, i.e., hiring someone else to do the job on your behalf, has been an attractive business practice for several years. As discussed in Chapter 2 of this Module, service providers (SPs) in the Internet economy are making it possible for companies to outsource many of its services. It is theoretically possible for a newly formed company to outsource *everything* by using a variety of service providers. In particular, the role of application service providers (ASPs) is of particular importance in enterprise application planning.

Cost and time saving are the two basic drivers for outsourcing through SPs. For example, a startup company may need a variety of applications such as payroll, inventory, purchasing, order processing, etc. to get started. Buying, installing, and running these applications and the underlying platforms would require an upfront cost even when there are none or very few customers. However, by outsourcing, i.e., renting these applications from an SP can be much more economical on a per usage basis. However, you should not outsource critical applications. For a more detailed discussion of outsourcing through ASPs, see Chapter 2 of this Module.

Develop from Scratch: In many cases, some or all portions of an application (databases, application code, user interfaces) need to be developed. For example, it is possible that the requirements can be satisfied by engineering (designing, developing, and deploying) a new database that is accessed by off-the shelf packages or end-user tools such as spreadsheets, data browsers, and report writers (e.g., data warehousing applications). If new application code needs to be developed, then we can assume that the new application code will be developed as an object-oriented C/S application. In a few cases, all components of new applications need to be developed. A later Module of this book ("Architectures") discusses these issues in detail.

Re-engineer Existing: Re-engineering (i.e., redesign, migration, interfacing) of existing, in many cases legacy, applications is an important aspect of IT practice at present. For example, it may be possible to satisfy the requirements by re-engineering (i.e., interfacing and migrating) the existing databases. Access to enterprise data, especially legacy data, from a diverse array of tools and applications, residing on a variety of platforms and interconnected through different network technologies is of key importance in most enterprises. In particular, many Web-based tools need to access enterprise data. In several cases, the requirements can only be satisfied by re-architecting and migrating the existing legacy applications that are old, unstructured, and monolithic. Choice of an appropriate approach depends on several factors such as business value of the legacy system and its technical value (see Figure 3-16). We will discuss the legacy application re-engineering issues in another Module of this book ("Integration").

3.4.4.2 Develop a Solution Architecture Sketch

As part of application planning, you should develop a sketch of a solution architecture that shows how the outsourced, bought, engineered, and re-engineered applications will work together at an enterprise level. Development of architectures for enterprise applications that span organizational units and enterprises is similar to establishing design of a city for an ever-changing and ever-evolving industrial and residential population. You have to worry more about how the individual parts of the city will be known to the city dwellers and how will they be interconnected (i.e., the infrastructure needed), instead of how the individual buildings will be designed internally. You only establish policies, rules and guidelines for the building

externals and focus more attention to the bigger issues of access and flows between the buildings (i.e., all buildings must be accessible). In a similar vein, architectures of enterprise applications is like designing many mini applications that need to interact with each other for corporate business goals. The emphasis is on identifying the interfaces of the applications, and the infrastructure needed to make this application operable as an enterprise-wide as well as, if needed, inter-enterprise application.

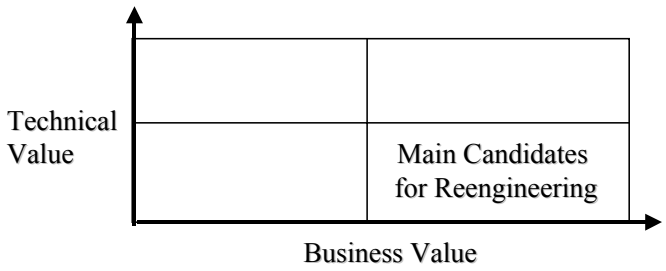


Figure 3-16: Legacy Application Analysis (Source: [Sneed 1995])

While developing the enterprise solutions, we need to keep in mind the structural issues in new versus old applications. Most new applications, such as discussed in the previous chapter, are being developed by combining three core technologies: object-orientation, client/server, and Internet (OCSI). Figure 3-17 shows an idealized view of OCSI applications. This view illustrates how web browsers are used to access hypertext documents, databases, and programs that may be located anywhere in the network.

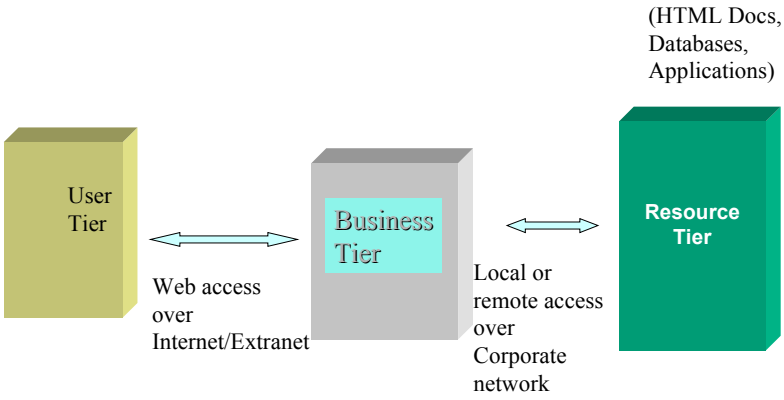


Figure 3-17: Ideal View of Object-Oriented Client/Server Internet Applications

The notion of "business components" is becoming popular in modern systems [Allen 2001, Herzum 2000]. Module "Architectures" of this book discusses the component-based application architectures in detail.

Including the existing, in particular, the legacy applications into a solution architecture is a non-trivial task. At present, the existing applications are commonly accessed/integrated with new applications through ***"Enterprise Application Integration (EAI)"*** platforms. This coexistence of old with new leverages the enterprise investment in existing systems and takes advantage of new technologies. Basically, the EAI platforms provide a common "bus" for application integration (see the following figure). EAI platforms, currently available from more than dozen suppliers, are designed to interconnect a large number of diverse applications through commercially available adapters. Data warehousing and application migration to newer platforms are other options in including legacy applications as players in an enterprise-wide solution architecture. The four chapters of Module "Integration" of this book discuss integration, warehousing, and migration options in great detail.

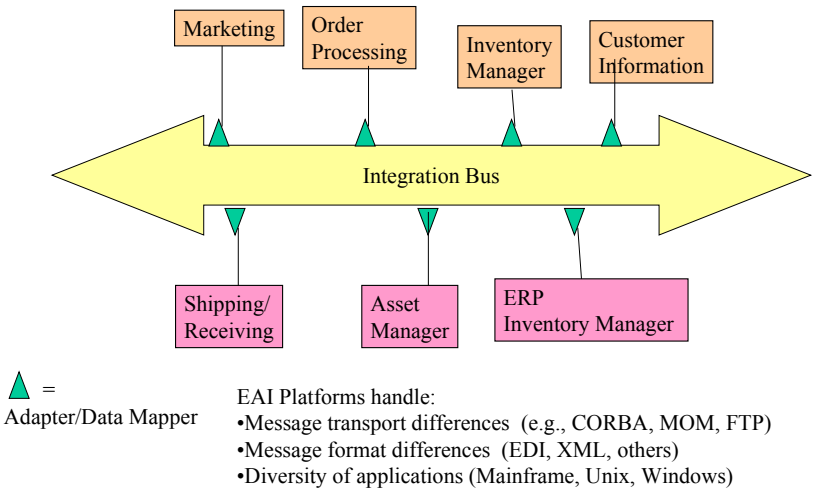


Figure 3-18: Integration through EAI (Enterprise Application Integration) Platforms

The overall solution architecture that combines applications (old plus new) with the enabling IT infrastructure may look something like a configuration shown in Figure 3-19. In this configuration for XYZCorp, the new applications reside in the middle tier that is accessed directly from the users. The existing back-end and external applications are accessed through some type of back-end integrator, in effect an EAI platform. We will discuss this configuration in the Module ("Architectures") of this book.

Strategy 4: Next Generation Enterprise Architecture (GENERIC)

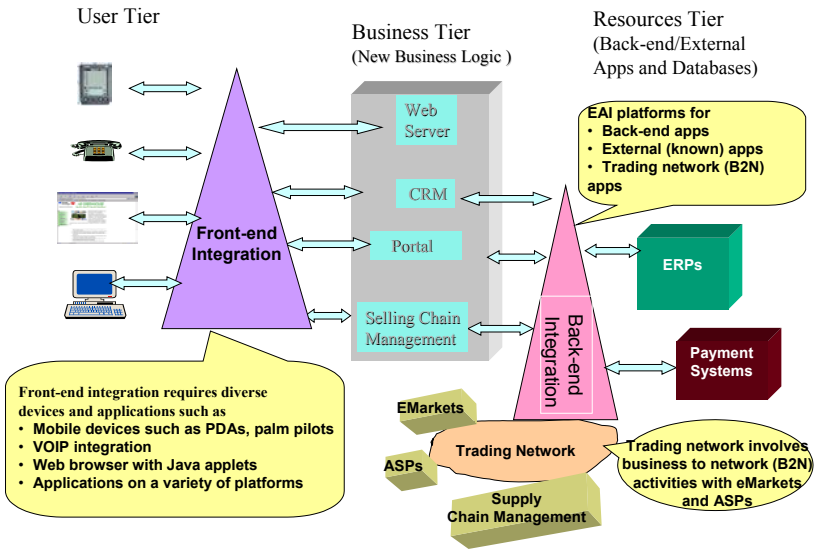


Figure 3-19: Solution Architecture that Combines New, Old, ASPs, for XYZCorp

It should be emphasized that the enterprise application architectures must satisfy the requirements of the Internet age because these applications operate in an environment where thousands of Internet users can possibly access your resources. Specific requirements may include:

- Internet Scale: Tens of thousands of users instead of hundreds, 24x7 not 9-to-5
- Internet connectivity: Unpredictable open Internet replaces the safety of the LAN
- Multiple customers: Security and load balancing between multiple customers

- Multiple configurations: Managing diverse user profiles and configurations
- High-volume infrastructure: Providing scalable services to diverse populations

3.4.4.3 Build Models and Prototypes of Solution Architecture

It may be important to build models and prototypes of solution architectures in application planning. This activity can be postponed till later, but let us discuss it here.

The objective is to demonstrate feasibility through prototypes and experiments. In particular, the application engineering/re-engineering decisions made (i.e., what needs to be built from scratch and what needs to be reused/restructered) is revised, if needed, based on hands-on discovery instead of end-less paper and pencil analysis. Although the details of this iteration depend on what is being engineered or re-engineered, we adopt an OO view as an overall framework for discussion. In particular, the object life cycle shown in Table 3-4 serves as a general approach that could be customized later. The basic idea is to develop and use an object model in this iteration and then refine and expand it in later iterations as production systems are built. This section gives a very high level overview of object modeling as described in the tutorial on OO concepts in Module ("Cases"). The sidebar "Key Object Oriented Concepts" should be consulted for a very quick review.

Table 3-4: Object Life Cycle Checklist

Activities	Steps
Analysis (Object Modeling)	▪ Capture key requirements ▪ Develop an Object Class Model ▪ Develop a system dynamic model
Architecture (Apply the Object Model to Design Application)	▪ Translate object and system dynamics model into modules ▪ Allocate modules to processors ▪ Develop inter-process communications ▪ Develop pseudo code and database design
Implementation (Build Application)	▪ Implement the user interfaces, databases, and program code ▪ Test the system
Deployment and support	▪ Install and support the system ▪ Enhance and maintain the system

The generic object life cycle shown in Table 3-4 is a good starting point. The objective of OO analysis is to capture the key requirements and then translate these requirements into an object model and a system dynamic model. Development of the object model involves identification of key objects in the system and their interrelationships. For example, Figure 3-20 shows an object model that represents a customer buying TVs and radios that belong to a product class. It is important in this stage to view systems in terms of business objects (also known as business components) because these objects represent business entities that can be re-used (see the sidebar "Business Objects and Business Components: Key to Reuse").

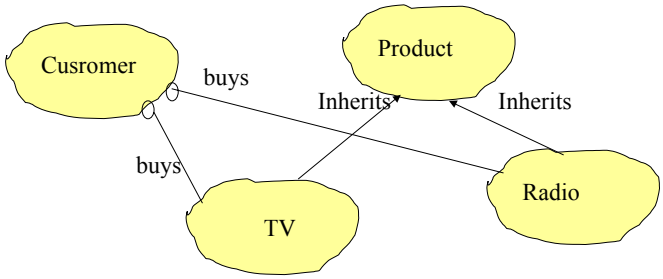


Figure 3-20: An Object Model

The system dynamic model captures the state transitions and other timing information. In particular, dynamic behavior of objects deals with issues such as synchronization of messages and event handling. Synchronization is essential for the server object that may need to manage multiple threads of control and event handling is a crucial aspect of most modern applications (most GUI applications are event-driven, i.e., they must respond to a large number of mouse clicks and key strokes). A wide variety of techniques can be used to specify and analyze dynamic models (e.g., directed graphs, message sequence charts, timing diagrams, Petri nets, discrete event simulations).

Object oriented architecture/design casts the results of the OO analysis into a specification that can be implemented. This activity consists of translating object and system dynamics model into modules, allocation of modules to processors, specification of inter-process communications, and pseudo code and database design. You exit this activity when you have enough details for implementations. Most of the current OO approaches assume development of software for a single machine (i.e., distributed objects are not considered).

Object oriented implementation involves developing and testing user interfaces, programming logic, and databases. Typically, OO programming languages are used to implement features such as classes, inheritance, polymorphism, methods and messages. The implemented system may use OO user interfaces (i.e., view system as objects, present operations on objects) and OO databases (i.e., store objects in databases, perform operations on objects in databases).

The object model and other information gathered during OO analysis is used to develop architectures and build prototypes of new applications and experiment with restructuring of existing applications. The knowledge and experience gained is used to refine the object model and build production systems.

Key Object Oriented Concepts

What are the Basic Object Concepts?

Object: An object is a piece of data surrounded by code (i.e., the data can only be accessed through code). This code is known as methods of an object. The data has certain attributes (e.g., name, address, age) and the methods are used to operate on these attributes.

Messages: Objects communicate with each other through messages. A message invokes a method of the target object (the object internals are known to the method and not to the object).

Classes: A collection of like objects makes up a class. A class acts as a template for similar objects (e.g., a class representing all Chevrolets). An object is an instance of a class.

Inheritance: Objects can inherit properties from other objects. Technically, inheritance allows you to create subclasses from parent classes. Subclasses inherit properties from parent classes. Subclasses can inherit multiple properties from multiple parent classes.

What is Object Orientation?

The following classic definition of object orientation is given by [Wegner 1987]:

Object oriented = objects + classes + inheritance

A more generalized definition, more suitable for distributed systems, is given by [Nicol 1993]:

Object oriented = encapsulation + abstraction + polymorphism

Encapsulation: The restriction of access to the object state via a well-defined interface (the operations). This involves a combination of two aspects: the grouping of object state and operation, and data hiding.

Abstraction: The ability to group associated entities according to common properties; for example, the set

of instances belonging to a class.

Polymorphism: The ability of abstractions to overlap and intersect. A popular form of polymorphism is inclusion polymorphism in which operations on a given type are also applicable to its subtype (for example, start a printer will start all types of printers). Inclusion polymorphism is often implemented via an inheritance mechanism.

Business Objects and Business Components: The Key to Reuse

The main business motivation for OO is increased reuse. The best way to achieve high reuse is to build objects that represent business entities (i.e., business objects). The basic idea of business objects, also known as business components, is that the users can construct objects that represent the real-world concepts of the business. Examples of business objects are customer, order, product, and regional office. If software could be structured around such objects and other business concepts, then organizations would be able to build software that simulates current business strategy. Moreover, businesses could reuse these objects to build new applications by using the OO paradigm.

You can represent business processes as objects (business things with policies as methods). For example, you can represent different business units as objects (i.e., purchasing department, shipping/receiving department). Since these objects represent business processes, these objects can change as businesses change. By using the OO concepts of encapsulation and abstraction, you can hide the internal details of business processes and build new business processes from these objects. After the business processes have been modeled as business objects, you can build applications by implementing business objects into lower level technical objects (e.g., GUIs, databases, etc.).

Business objects started appearing in the marketplace around the introduction of OLE 2.0, OpenDoc, and CORBA-based products in 1994. Since then, OO tools designed to support creation of business objects have emerged from vendors such as Easel and applications that employ business objects have appeared from vendors such as SAP and IMRS. The Object Management Group (OMG) has found the Business Object Management Special Interest Group (BOMSIG) to help the industry understand and utilize this technology effectively. System integrators such as Anderson Consulting are also committing to deliver software environments that can be used to build custom applications based on business objects.

Business objects are at the core of "Class-Based Re-engineering (CBR)" that integrates business process re-engineering with object technologies [Newman 1996]. This approach starts out by building a business model in terms of object classes and then uses these classes to architect and build systems. The object classes can represent different business entities such as customers, loans, accounts, etc. These classes can change to reflect business changes and tie into technology because these classes can be implemented by using OO technologies.

The books [Allen 2001, Herzum 2000] provide a comprehensive discussion of business objects and business components.

Build Prototypes of New Applications. There is no "one size fits all" approach for prototyping all new applications in the modern environments. For example, decision support applications mainly require a database design with almost no software development. These applications rely heavily on off-the-shelf query and browsel tools and use the object model to design the database (we will see later that the object model is very similar to the data model that is typically developed by the database designers.). There is virtually no reason to develop a system dynamics model for decision support applications (most activities are queries).

On the other hand, enterprise-wide mission critical operational support applications may require extensive software development and may require, in addition to the object model, a great deal of information about the system dynamics and flow. The real-time applications rely, for obvious reasons, on a very extensive system dynamics model.

Table 3-5 suggests the main activities in prototyping different classes of applications. The rows of this table show the four generic activities (i.e., analysis, architecture, implementation, deployment) and the columns represent the three broad application categories (i.e., decision support, operational support, and real-time). We introduced these application categories in Chapter 1 of this Module and discussed them in terms of in terms of span (i.e., departmental, enterprise-wide, inter-enterprise).

You can extend this discussion to other cases that require mixtures of the cases discussed here (e.g., applications that are operational support at corporate level but provide decision support capabilities at the departmental/group level) for these classes of applications. We will explain the entries in this table in Module "Architectures" (Chapter 1).

Table 3-5: New Application Prototyping Activities

	Decision Support Applications	Operational Support Applications	Real Time Applications
Analysis Stage (Refine the Object Model)	- Gather information requirements iteratively - Develop the object model for database design - No dynamic model (or lightweight)	- Gather and analyze detailed requirements - Develop the object model - Develop the dynamic model	- Gather the "hard" requirements - Develop the object model - Develop the dynamic model in great detail
Architecture Stage (Build architecture for prototyping)	Mainly data architecture (iteratively)	Data + application software architecture	IT infrastructure is of great interest because it can determine success/failure
Implementation Stage (Build the prototype)	Selection and testing of tools; population of databases	Application software development and testing (if cannot be purchased)	May involve middleware development
Deployment and support (Run pilots, if possible)	Installation of tools at user sites and quick response to changing needs	Installation of software at user sites and technical support/help desks	Installation of specialized hardware and softwares at user sites and technical support/help desks

Experiment with Re-engineering of Existing Applications. The object model and other information gathered during OO analysis can be used to experiment with re-engineering of a wide range of existing, including legacy, applications. The exact activities depend on the type of re-engineering strategy chosen. For example, access and integration of legacy applications requires selection of appropriate surround technologies that “wrap” the legacy functions and data (“object wrappers” are increasingly being used for this purpose at present). In this case, the object model can be used to represent various legacy functions and data as object classes. Data warehouses, a popular approach to deal with legacy systems, are decision support applications that require database design and data warehouse tools. Design of data warehouses uses the object model to represent the database design as discussed previously in decision support applications. For migration, the object model can be used to encapsulate the application components that will be transitioned, thus making the migration transparent to the users (see [Desfray 1996] for a case study).

Table 3-6 suggests the main activities for different application re-engineering “experiments”. The rows of this table show the four generic activities (i.e., analysis, architecture, implementation, deployment) and the columns represent the three main application re-engineering strategies (i.e., access/integration, data warehouses, migration). We will explain the entries in this table in Module "Application Integration and

Migration" of this book in the chapters on application integration, data warehouses, and migration, respectively.

Table 3-6: Application Re-engineering Activities

	Application Integration	Data Warehouses	Migration Analysis
(Refine the Object Model)	- Detailed requirement specification and analysis for accessing and integrating legacy system (e.g., types of legacy systems, type of access needed) - Use of object model to wrap the legacy system components	- Detailed requirement specification and analysis for data warehouses (e.g., informational requirements, data mining needs, interconnectivity requirements) - Use of object model to design data model	- Detailed requirement specification and analysis for migration (e.g., time allowed, level of migration, budget and policy restrictions) - Use of object model to encapsulate functions to be migrated
Architecture (Evaluate solution approaches)	Evaluate access/integration strategies, choose middleware for access/integration	Data warehouse architecture (e.g., what data is in the warehouse, synchronization policy)	Evaluate and determine a migration strategy (e.g., user interface migration, data migration)
Implementation (Install and Test the experiment)	Choose needed middleware and underlying networking technologies	Design the data warehouse data and select tools	Choose the most appropriate migration gateway
Deployment and support (use as a pilot)	Determine cost, time, and staffing considerations	Assess training and support considerations	Gradual migration support considerations

3.4.5 Step 4: Assess and Plan Appropriate IT Infrastructure

IT infrastructure (platform) planning is concerned with determining the most appropriate technology needed to develop and deploy the application systems. Examples of such infrastructures are the Intranets which provide Web services in corporate private networks; factory networks which connect many manufacturing devices to cell and area controllers, and "Extranets" which connect many business (e.g., healthcare industry participants). IT infrastructure planning is a much more challenging and crucial task than network planning because the diverse applications in business, engineering, and manufacturing written in different software/database formats which reside on heterogeneous computing devices need to be interconnected through different communication media by using a variety of communication software packages.

Specifically, the main infrastructure capabilities (middleware, network services, local computing services) needed to support applications must be carefully examined in the planning iteration (see Figure 3-21). The IT infrastructure becomes increasingly important as you iterate through the system life cycle (i.e., planning iteration only concentrates on high level issues that could be "show stoppers" while the first release must go through detailed considerations such as the exact version of middleware needed). The details depend on the type of applications being engineered/re-engineered. For example, legacy data access requires different type of infrastructure than a Web-based distributed object application.

The real IT infrastructure is a complex combination of middleware services that reside on different computers that are interconnected through a network. Figure 3-22 shows an example where a web browser accesses a purchasing system that in turn accesses a catalog. We will explain infrastructure in more detail in the next chapter. Extensive discussion of IT infrastructure can be found in other modules of this book ("Networks", "Middleware", and "Platforms").

Basically, the entire IT infrastructure should appear as a tightly integrated environment which provides a range of networking, database, transaction management, remote messaging, naming, directory, security, and other services needed by the applications. The key question is: how can we put all these pieces together into a functioning IT infrastructure that can support the variety of services and applications needed by the modern enterprises? The following iterative steps are suggested as a starting point:

- Analyze the requirements for infrastructure
- Architect the infrastructure services

- Implement the infrastructure services
- Deploy and support the infrastructure services

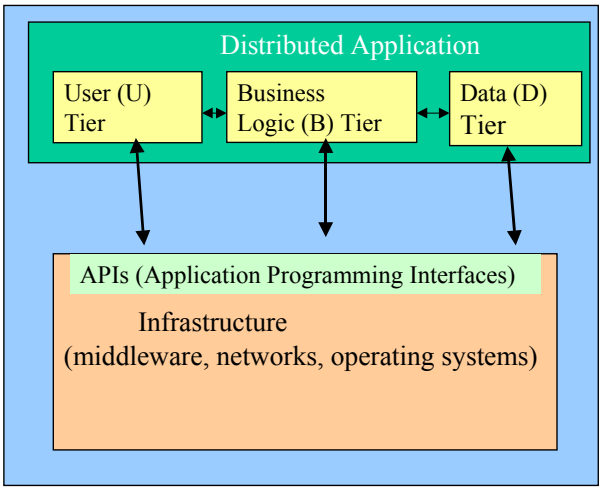


Figure 3-21: Conceptual View of Infrastructure Role

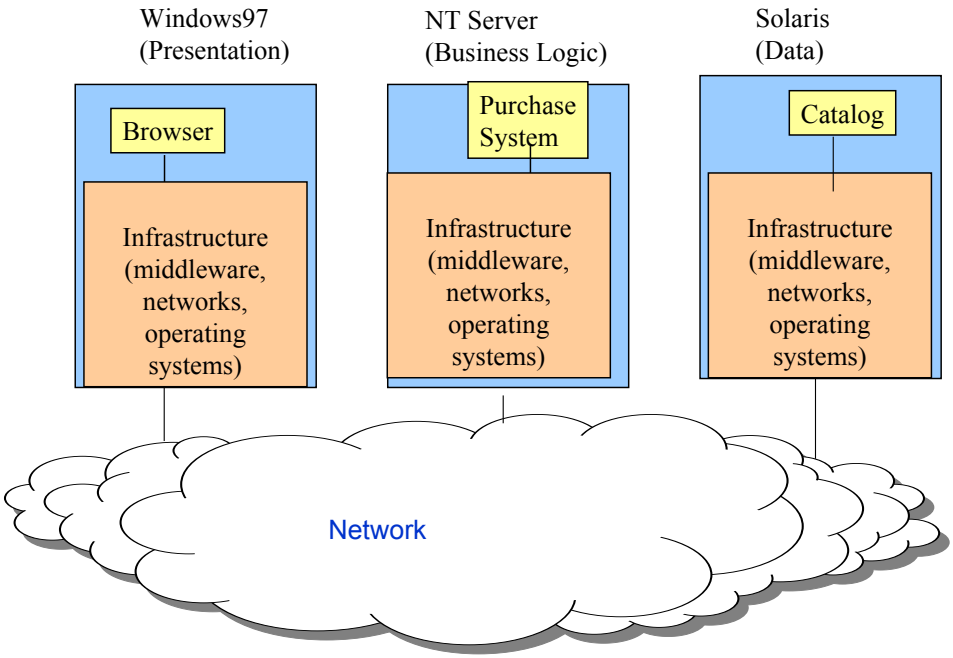


Figure 3-22: Example of Infrastructure

Table 3-7 summarizes the main activities in each stage of the methodology introduced in Section 3.3 for the infrastructure services. The rows of this table show the four generic activities (i.e., analysis, architecture, implementation, deployment) and the columns represent the three classes of infrastructure services (i.e., middleware, network, local).

Infrastructure issues are discussed extensively in other modules of this book ("Networks", "Middleware", and "Platforms").

Table 3-7: Summary of Infrastructure Considerations

Stage	Middleware	Network Services	Local Computing Services
Analysis	Specification and analysis of requirements for middleware services	Specification and analysis of requirements for network services	Specification and analysis of requirements for local computing services
Architecture	Select proper middleware (e.g., Web services, distributed objects, distributed data middleware)	Select proper network products (e.g., communication technologies, interconnectivity devices)	Select proper local computing products (e.g., database managers, operating systems)
Implementation	Select new vendor products and/or upgrade existing	Select new vendor products and/or upgrade existing	Select new vendor products and/or upgrade existing
Deployment and support	Examine and investigate deployment and support considerations	Examine and investigate deployment and support considerations	Examine and investigate deployment and support considerations

3.4.6 Step 4: Cost/Benefit Analysis

3.4.6.1 Overview

The main purpose of cost/benefit analysis is to compare the business benefits with the estimated time, money, computing and human resources needed for the proposed application engineering/re-engineering. The key idea is to determine if the proposed engineering/re-engineering is cost beneficial and if the risks are manageable. You must clearly understand the risks and payoffs before launching an effort because engineering of new and re-engineering of existing applications is not risk free. Although it is fashionable to jump on the latest technology bandwagon, many new technologies need to prove themselves in large scale mission critical situations and demonstrate business payoffs. Object-oriented, client/server, Internet-based technologies are no exception.

Costs and benefits of emerging technologies such as object-oriented, client/server, Internet (OCSI) environments is tricky. In general, most new technologies are accompanied by euphoric claims of benefits/promises with a deaf year (typically for 2 to 4 years) to costs/pitfalls. To some extent, this "positive wave" is natural because the real costs and pitfalls are revealed only after some hands-on experiences have been accumulated. This happened with client/server (C/S) systems in the mid 1990s. The results of C/S computing are very interesting in this context because they can be extrapolated to learn some lessons. Towards this goal, the costs and benefits of C/S computing are used in the following sections. Some cost benefit analysis and business justification for more recent issues such as enterprise application integration (EAI) and component-based software is beginning to appear [Acharya 2003, Bass 2001, Lim 2003].

The client/server technology wave hit us all around 1992 as a cure for all the evils of the old mainframe-based systems. Many C/S success stories were reported in the trade journals in the early 1990s. However, there have been numerous failures, albeit not well documented (see the sidebar "Failures -- Old Lessons from Client/Server"). Here is the main lesson: -- *there will be successes as well as failures in any technology implementations, but failures will be rarely publicized and will come to surface a few years after the initial wave (usually at the beginning of the next wave).*

Although most of the discussion on cost/benefit analysis is based on client/server, some specific cost benefit studies should be noted. For example, see the "Cost-Benefit Analysis Guide for the National Institute of Health (NIH) IT Projects" (http://irm.cit.nih.gov/itmra/cbaguide.html).

Failures -- Old Lessons From The Client/Server Wave

No new technology is risk free. What we are calling OCSI is no exception. Although, this technology offers many promises, there are several potential pitfalls (see the sidebar "Promises and Pitfalls of OCSI").

Unfortunately, many pitfalls are discovered after the fact and are usually not paid attention to initially.

The client/server technology wave hit us all around 1992 as a cure for all the evils of the old mainframe-based systems. There have been, admittedly, several well documented C/S success stories (see, for example, the Client/Server Journal, Computerworld, June 1995 Special Issue, which contains 20 C/S success stories that have been reviewed and selected by a panel of experts). However, there have been numerous failures, albeit not well documented. Here are some examples of failures intended as lessons to be more cautious:

- Information Week C/S backlash survey conducted at the end of 1995 indicated that 80% of the respondents had to reevaluate, cancel, or pull back from a C/S development project. The reasons include lack of management support, poor system performance, and costs higher than expected [DePompa 1995].
- Early examples of failures are given in "Dirty Downsizing", Computerworld, Special Report, August 10, 1992, These examples show how some companies sufferred major losses when they tried to replace their mainframes with Unix and/or PC-based C/S systems. Other examples of failures are presented by [DePompa 1996, McGee 1996, Jenkins 1996]. Reasons for these failures are repeatedly lack of management direction, difficulties of working with vendors, and immature technologies.
- A Gartner Group report [Dec 1995] includes a report card on C/S computing. Basically, the report card gives C/S computing failing grades on lowering costs (costs go up), higher availability (C/S systems tend to fail more often), and serviceability (too many components to service).
- The hidden costs of C/S have been much more than expected [Latch 1996]. According to a study conducted by the University of California at Berkley's School of Business, every $1000 spent on server hardware/software needs to be matched with $9000 of management and maintenance cost [Jenkins 1995].

Sources:

- Dec, K., "Client/Server - Reality Sets in", Gartner Group Briefing, San Diego, February 22-24, 1995.
- DePompa, B., "Corporate Migration: Harder Than It Looks", Information Week, Dec. 4, 1995, pp. 60-68.
- "Enterprise Client/Server: Can we get there from here", Gartner Group Briefing, July/August 1994.
- Jenkins, A., "Centraliation Strikes Again", Computerworld Client/Server Journal, August 1995, pp. 28-31.
- Latch, C., "The Hidden Costs of Client/Server Computing", Data Management Review, Nov. 1996, pp. 16-20.
- McGee, M. and Bartholomew, D., "Meltdown", Information Week, March 11, pp. 1996, pp. 14-15.

Cost/benefit analysis of IT, in general, is fuzzy and subjective. The main idea is to be realistic in your expectations about benefits and costs. Since costs (and risks) are facts of life, you need to determine the strategies which maximize business benefits and minimize/manage costs and risks. A survey of a large number of case studies has indicated that successful application engineering/re-engineering efforts exploit the promises of new technologies but carefully understand and manage the risks while the others do not (see Figure 3-23 and Figure 3-24). What precisely are the promises and pitfalls of OCSI applications from what we know at this point? We will visit these issues in more detail as we go along.

3.4.6.2 Estimating Costs

Surveys for the costs/benefits of C/S computing became available in the mid 1990s (see the sidebar "Sources of Information for Client/Server Costs/benefits"). For example, a Gartner Group [Dec 1995] report analyzed costs for C/S computing (see Table 3-8). The main findings of this analysis are:

- Hardware/software costs represent only 15 to 20% of total cost
- Labor costs represent the largest expense (70 to 75% of the total costs are labor)
- Total cost per client is about $50K for five years

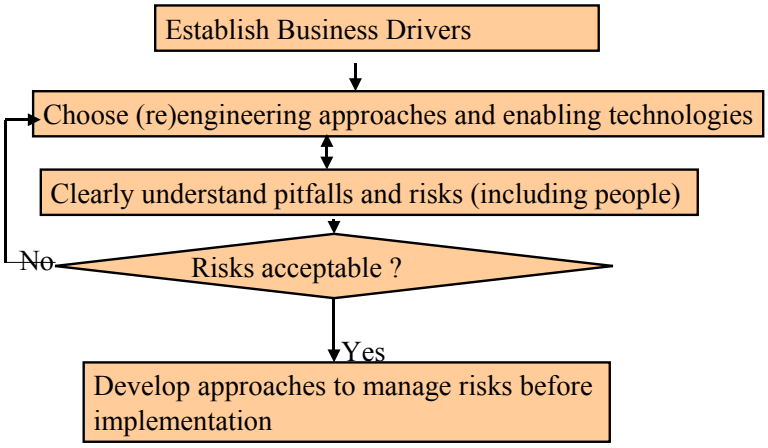


Figure 3-23: Recipe for Success

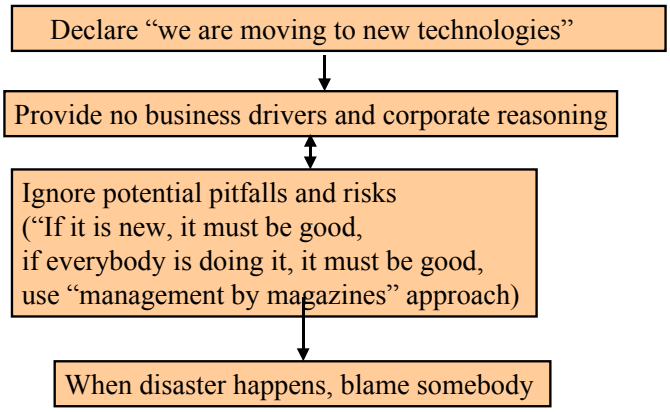


Figure 3-24: Recipe for Failure

[Dec 1995] also includes a report card on C/S computing. Basically, the report card gives C/S computing failing grades on lowering costs (costs go up), higher availability (C/S systems tend to fail more often), and serviceability (too many components to service). However, the grades for rapid deployment and better scalability are better (B and C, respectively).

Many costs are hidden and are not apparent initially in C/S projects. For example, middleware cost per mainframe is about $200K, and per PC cost is about $500). Installation of a large scale C/S system involving, for example, 1000 desktops could easily 0.5 million dollars. Distributed applications also require many software licenses at many computers, thus potentially increasing the software costs.

In addition, the initial axiom of one server machine per application has not resonated very well. According to a study conducted by the University of California at Berkley's School of Business, every $1000 spent on server hardware/software needs to be matched with $9000 of management and maintenance cost [Jenkins 1995]. Owing to this, many organizations have started using powerful servers that can house many applications (i.e., more centralization).

While estimating costs, several intangible factors need to be considered. Examples of these intangibles are increase in interdependencies and points of failure. Concerns for security and integrity control, and many management/support issues must be addressed. We will visit these issues in another module ("Management and Support Services").

Costs of migration to OCSI should be evaluated very carefully (we will visit this issue in Part IV of this book). Basically, potential of big savings for application migration to OCSI is there, but keep in mind that the mainframe costs are dropping. In addition, many hidden costs are hard to control/quantify because significant costs shift to consulting and training. Full cost savings can only be realized if the mainframes are retired completely and the expected payoffs do not happen right away (many efforts take 3-4 years with payoffs in the last year). In addition, many people issues arise because traditional IS staff and users may resist (new skills are usually needed) and responsibilities shift to end users.The bottom line: hardware cost savings alone are not enough for transitions and should not be used as the key business drivers.

In summary, it has been found that C/S increases total costs due to the following reasons:
- End user effort in learning and operating desktops is higher
- End user help desks require more support
- Many hidden costs at end user sites (e.g., middleware) emerge
- Distributed applications are more complex

Guidelines for Cost Estimation

Over the last 20 years, many cost estimation techniques for information systems have been suggested. Despite a great deal of work, most cost estimates in information systems are based on heuristics and guidelines. Lederer and Prasad [Lederer 1992] suggest the following 9 guidelines for better cost estimation, with numerous examples:
- Assign the initial estimating task to the final developers.
- Delay finalizing the initial estimate until the end of a thorough study.
- Anticipate and control user changes.
- Monitor the progress of the proposed project.
- Evaluate proposed project progress by using independent auditors.
- Use the estimate to evaluate project personnel.
- Computing management should carefully study and approve the cost estimate.
- Rely on documented facts, standards, and simple arithmetic formulas rather than guessing, intuition, personal memory, and complex formulas.
- Do not rely on cost estimating software for an accurate estimate.

3.4.6.3 Estimating Benefits

Most benefit surveys are not quantitative. Benefits typically occur in the end-user departments and not in the corporate IT departments (thus they are difficult to understand and quantify). Frequently mentioned benefits of C/S technology include:
- Better organizational fit (decentralization and local autonomy)
- Improves competitiveness (e.g., presence in new markets)
- Improves customer service quality, responsiveness
- Supports business restructuring
- Allows faster development of new applications
- Improves flexibility of services
- Easier access to corporate data
- Improves end user productivity
- Reduces training time due to GUI
- Allows exploitation of new technology

The introduction of C/S technologies have in general strengthened the position of IT departments because some of the functionality and budget dollars have shifted to the IT departments away from the central MIS departments. (A Gartner Group 1996 report indicates that the annual revenue for decentralized functions will

jump from 2.3% to 6.8% between 1994 to 1999, and will reduce from 2.0% to 1.6% for the centralized functions in the same time period). However, the central MIS departments are not extinct -- their role has changed to more enterprise wide consultation, integration and management [Grygo 1996].

In general, C/S benefits are scattered outside the IS department such as:
- Flexibility in handling business changes
- End user control increases

Table 3-8: How Much Do Client/Server Systems Cost? (Source: [Dec 1995])

	Small Organization	Large Organization
Configuration	Single site 20 clients One PC-based server One LAN administrator	250 remote sites with servers 5000 clients Legacy and new enterprise servers 83 end-user support staff 55 application developers Centralized management
Total Costs	Total five year C/S cost: $1,016,000 Cost per client = $50,000	Total five year cost: $241,800,000 Cost per client: $48,400
Cost Breakdown	72%: Labor cost 41%: End user labor 15%: End user support	72%: Labor cost 41%: End user labor 15%: End user support

Sources of Information for Client/Server Costs/Benefits

- Datamation supplement "Client/Server: The Business Advantage", June 15, 1994.
- Dec, K., "Client/Server - Reality Sets in", Gartner Group Briefing, San Diego, February 22-24, 1995.
- DePompa, B., "Corporate Migration: Harder Than It Looks", Information Week, Dec. 4, 1995, pp. 60-68.
- “Dirty Downsizing", Computerworld, Special Report, August 10, 1992.
- "Enterprise Client/Server: Can we get there from here", Gartner Group Briefing, July/August 1994.
- Grygo, E., “Upper Hand”, Client/Server Computing, August 1996, pp. 34-42.
- Jenkins, A., “Centraliation Strikes Again”, Computerworld Client/Server Journal, August 1995, pp. 28-31.
- Korzenioski, P., "Many Different Paths to Get to Downsizing", Software Magazine, January 1993, pp. 81-85.
- Liana, A., "Controlling costs for downsizing", IBM Internet Journal, June 1994, pp. 34-39.
- Moad, J., “Client/Server Costs: Don’t Get Taken For a Ride”, Datamation, Feb. 15, 1994, pp. 34-38.
- Standish Group conference on client/server failures.
- "Where Do Client/Server Apps Go Wrong", Datamation, Jan. 21, 1994, p. 30.

3.5 XYZCorp Case Study: Hints and Suggestions

We have worked through this case study in previous sections but let us recap the main points. The first two iterations of the methodology described in Sections 3.3 through 3.5 should be completed as much as possible within the two week time constraint. In the planning iteration, an understanding of application interrelationships is developed and very initial choices are made about what applications should be built, what should be purchased, and what should be based on a re-use of existing systems. A high level model is built in the second iteration to gain further insights and to scope out the work in more detail. We have

worked through the steps of the application planning described in this chapter to develop and refine the application architectures.

Time To Take a Break

- ✓ Methodology Overview
- ✓ IS Planning
- Case Studies and Examples

Suggested Review Questions Before Proceeding

- What is application planning and what are the results of application planning?
- What are the main steps of application planning? In each step, what are the two main activities and what are the main results from each step?
- Suppose you had to do application planning for an online purchasing system for an organization in a hurry (have to show results tomorrow morning). What steps will you skip, if any, and why (this actually happened to me!)
- List some examples of application planning that you are familiar with

3.6 Case Studies and Examples

The following case studies and examples illustrate how variants of the methodology pattern discussed in this chapter can be used to translate strategies to working solutions.

3.6.1 A Financial Marketplace

A large international bank, let us call it ***Xbank***, and an emarket provider, let us call it **Zmarket**, have formed a partnership to deliver financial services to B2B exchanges. This system, also discussed in Chapter 1 of the Cases Module, was developed by using an approach very similar to the methodology discussed in this chapter (strategic analysis, application identification, and architectural sketches).

3.6.1.1 Strategic Analysis

The basic strategy is that Xbank will provide a wide suite of financing services to B2B members through a link with Zmarket. By joining hands, Zmarket as well as Xbank intend to increase their customer base and offer broader services. Figure 3-30 shows a high level view of the intended B2B Service. This Service involves a B2B exchange, Zmarket, Xbank, and the ***Partner Bank Network (PBN).*** Zmarket will route financial transactions to the Xbank link. For those transactions that Xbank processes, Xbank will route the incoming messages from Zmarket to its respective systems and operations areas. For those transactions not processed by Xbank, Xbank will route the transactions to its correspondent banks, which will make up the PBN.

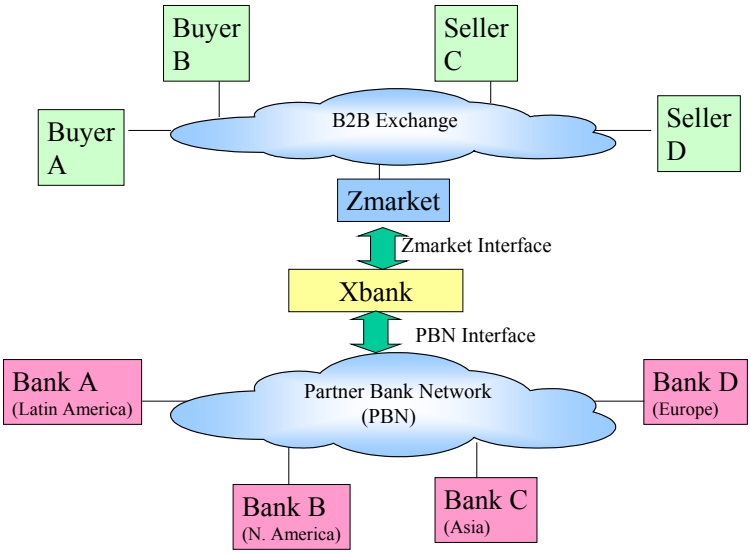


Figure 3-25: Conceptual View of a Financial Marketplace

B2B Exchange. This exchange provides members the ability to meet trading partners and arrange a deal between buyers and sellers. However, for the buyers and sellers to close a deal, credit and payment settlement needs to be made. Zmarket passes the financial transactions to Xbank for processing. For example, a seller will send the financial information about a buyer to Zmarket to verify. Zmarket will process the information and then send it to Xbank for financing and payment.

Zmarket. Zmarket will provide the authentication and certification that will underlie the transactions conducted on the exchange, as well as the transactions sent and received between the exchange and Xbank. The certificates are expected to enable trading partners to close their deals online as well as request transaction fulfillment via digital signatures.

Xbank. The bank will provide financial services to the B2B exchanges through Zmarket. Xbank may process the transactions itselfes, in which case the transaction request would be routed to the appropriate processing system and/or operations area. If Xbank *does not* process the transaction, then it will route the transaction to one of its partner banks for processing (depending on the transaction request and the underlying client).

Partner Bank Network. The B2B exchange can benefit from a network of financial institutions that are business partners of Xbank. Instead of financial institutions maintaining direct relationships with the members of the exchange, they can use Xbank's interface to the B2B exchange. Basically, Xbank receives the financial transaction (e.g., request for a letter of credit) and attempts to process it by itself. If not, it broadcasts the transaction to the partner network for bids. It thus serves as a clearinghouse for the partner banks.

3.6.1.2 EMarket Transactions and Transaction Flows (Application Analysis)

Many applications and transactions have been identified. Let us concentrate on two: Payment and Finance.

Payment System. Figure 3-26 shows the general payment flows that involves direct payment (without intermediary banks) as well as through intermediaries. The following describes a possible payment flow process in the B2B exchange environment.

- Buyer sends a payment request on the B2B exchange.
- Zmarket receives the transaction request, certifies the sender and the instruction and passes the authorized transaction to Xbank through the Xbank-Zmarket interface in the agreed format.
- Xbank receives the instruction and validates that the authentication conditions have been met. If the Buyer has an account with Xbank and there are available funds, then Xbank will debit the Buyer

account and make the payment to the Seller bank through electronic fund transfer or other means. If Xbank does not have an account for the Buyerr, then Xbank will determine if there is an account for the Buyer's Bank on the Partner Bank Network. .

Typically, both the Seller and the Buyer require some form of advice that the payment has been made/received. In addition, each instructing party in the chain usually requires some confirmation/advice that their instruction has been executed. Each bank has specific advising arrangements with each of their clients and set prices for the service accordingly. Advices may be sent via phone, fax, mail, or other means.

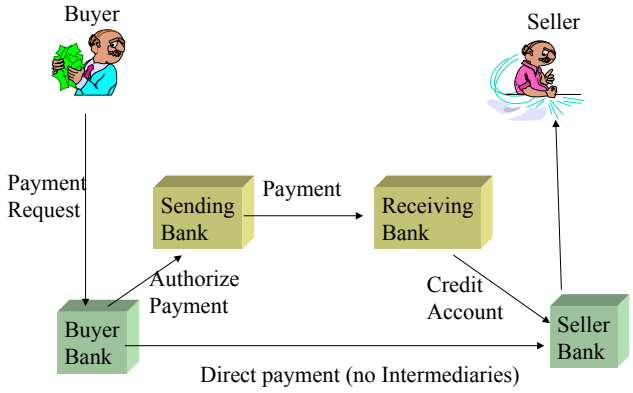


Figure 3-26: General Payment Flows

Let us work through a few scenarios of payment, depending on whether the Remitter has an account with Xbank or with a PBN bank.

The flows are represented as sequence diagrams that are used heavily in system modeling techniques such as UML (Unified Modeling Language). System models are typically built in UML by using tools such as Rational Rose. These tools not only perform consistency/completeness tests but also generate, if needed, skeleton code in Java and C++ to speed up the system building process. .

Figure 3-27 shows a sequence diagram for Scenario 1, i.e., when Xbank has an account for the Buyer. In this case, Xbank after authentication and verification of available funds, will debit the Buyer's account and make the payment. After the payment has been made the payment advice is sent back to the Buyer.

Figure 3-28 shows the sequence diagram for scenario 2, i.e., Xbank does not have an account for the Buyer's Bank, but the Buyer's Bank is a partner bank of Xbank. In this scenario, Xbank will forward the instruction to the Buyer's Bank requesting them to make the payment.

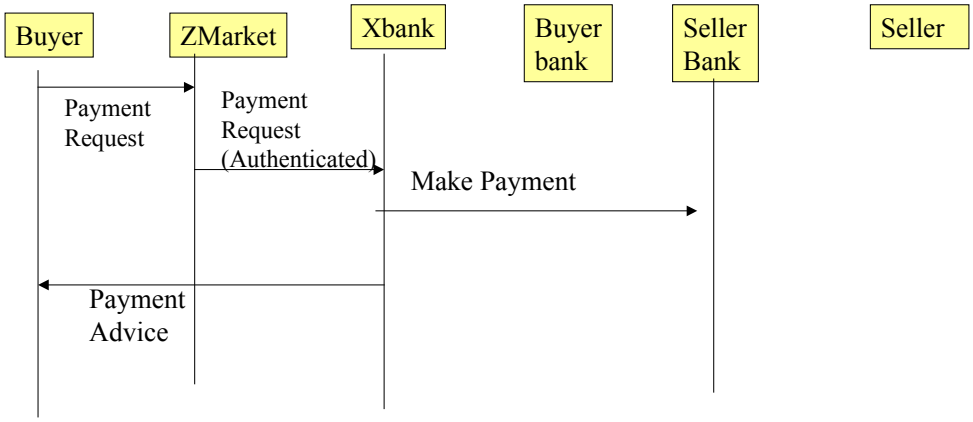


Figure 3-27: Payment Scenario 1 - Xbank Account

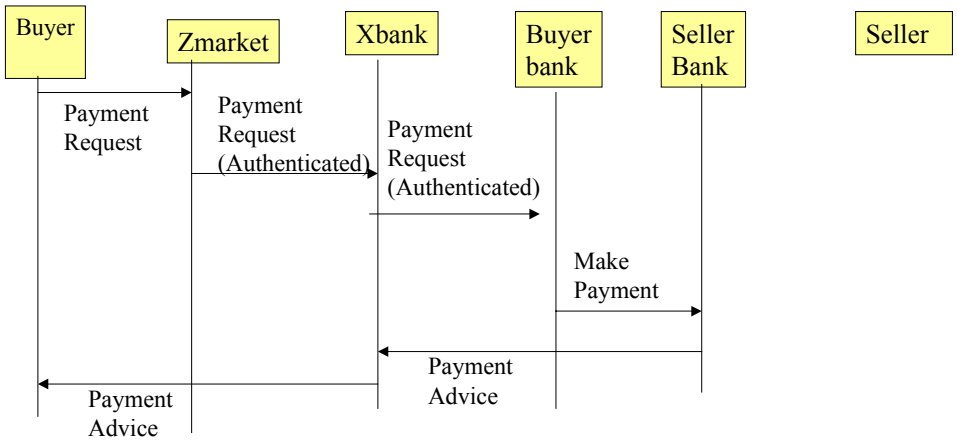


Figure 3-28: Payment Scenario 2 - PBN Xbank Account

Financing Flows. Figure 3-29 shows the general financing flows. There are several variations on the types of financing and the relationships between the various parties. The buyer, seller or both may require financing. Additionally, the banks may request financing and credit enhancements from other financial institutions in support of the payment assurance they provided to their clients. Various financing arrangements and variations are currently utilized in cross-border commerce. It is beyond the scope of this case study to discuss different financing arrangements. For the purpose of discussion, we will assume Buyer/Seller Financing. In buyer financing, also known as import financing, the buyer lender agrees to finance the buyer (importer), thereby taking the credit risk of the buyer. In seller financing, also known as exporter financing, the seller lender agrees to finance the seller (exporter), thereby taking the credit risk of the seller.

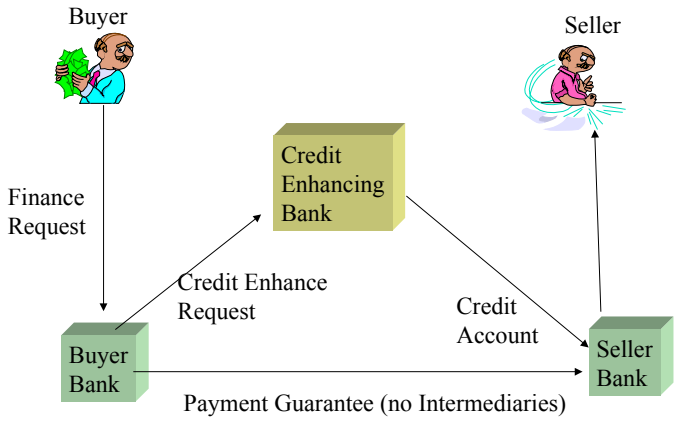


Figure 3-29: General Financing Flows

The following describes a possible financing process flows in the B2B exchange environment for Financing

- The financing may occur prior to, as a prerequisite of or after the deal closing between the seller and buyer.
- Buyer/seller requests financing for a deal on the B2B exchange. The request goes through the Zmarket certification process and is forwarded to Xbank. The financing request includes the name, address, account and financial institution of the requesting party, amount and tenor of financing, payment trigger, along with the details of the underlying transaction (i.e. goods involved, etc.).

- Xbank receives the financing requests and determines if it will provide the financing or if it will pass the transaction on to the Partner Bank Network. If Xbank provides the financing, it will confirm back to the exchange (via Zmarket), the terms of the financing (i.e. rate, maturity, terms, etc.). If the transaction is passed on to the Partner Bank Network, then the PBN will be given notice of the financing request including the details mentioned above. The Partner Banks will bid on the request and then pass the financing information back to Zmarket through Xbank.

3.6.1.3 An Architectural Sketch

Figure 3-30 shows a high level architecture diagram of the pilot. This architecture is a more detailed view of the logical view presented in **Error! Reference source not found.**. This high level architectural diagram shows that:

- B2B Exchange is based on the public Interact (i.e., it uses HTTP messages)
- The traders on the exchange that use Zmarket services use the Zmarket client to communicate with a Zmarket server that resides at the Zmarket site. .
- Zmarket is connected to Xbank systems through a line that may be a dedicated T1 line. Xbank clients transfer the information between Zmarket and Xbank
- The message transported between Zmarket and Xbank will be in an XML variant such as FinXML.
- The Xbank systems receive the messages from Zmarket and send them over the Partner Banking Network (PBN).

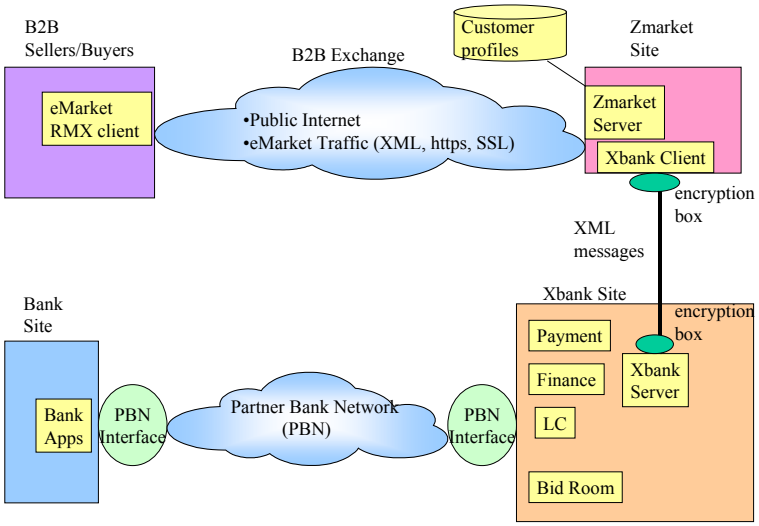


Figure 3-30: High Level Architecture of a Financial eMarket

Let us go through a few more details about this architecture.

B2B Exchange. B2B exchanges provide members the ability to meet trading partners and arrange a deal. The Exchange is based on the public Internet (i.e., the members use Web browsers, web servers, and HTTP for communications over the IP network).

Zmarket . Zmarket provides the authentication and certification through a server that will underlie the transactions conducted on the exchange, as well as the transactions sent and received between the exchange and Xbank. The certificates are expected to enable trading partners to close their deals online as well as request transaction fulfillment via digital signatures.

Xbank to Zmarket Link. Due to high security concerns, there will be a direct CPU-to-CPU link (a T1 line) between Zmarket and Xbank. Hardware encryptors will be installed on both ends of the T1 line for security.

The Xbank software will transfer information between Xbank and Zmarket. The interface between Zmarket and Xbank is expected to change over time.

XML Messages Exchanged Between Xbank and Zmarket The messages exchanged between Xbank and Zmarket will be based on XML. The following XML standards are worth consideration: FinXML, FpXML, and SWIFTML (a markup language used in the Swift network - www.swift.com) .

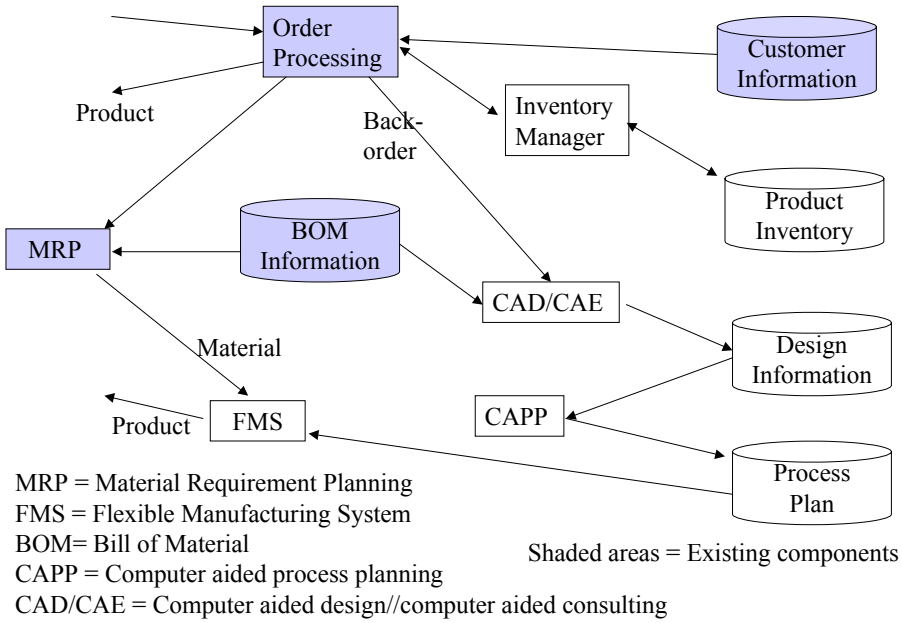


Figure 3-31: Advanced Integrated Control System (AICS)

Xbank Applications. Xbank software will receive the information from Zmarket and will invoke one of the following applications: Payment, Finance, and LC (Letter of Credit). These applications, as the names imply, will handle the payment, financing and letter of credit scenarios (some of these scenarios have been discussed previously). Let us discuss a new application (bid/auction room) that will need to be developed specifically for PBN.

Xbank wants to create an online mechanism, similar to an auction room for the financial institutions to participate in financial transactions originated through an exchange and routed through Xbank. This "auction room" will get the PBN banks involved in cases where Xbank does not execute the transaction, or in cases where the buyers/sellers may not have direct accounts with the PBN banks. The main idea is that a possible financial transaction can be broadcasted to the PBN for bidding. The auction room will obtain the bids, summarize them, and send them to the customer (buyer/seller). Based on customer selection, then a financial transaction is executed.

3.6.2 Integrating Manufacturing Processes – An Example

A manufacturing company is interested in integrating and automating the order processing, inventory control, CAD/CAM (computer-aided design/computer-aided manufacturing) and the "manufacturing" processes of the company products (IBM PC desktops, laptops, network computers, personal digital

assistants). This system, referred to as the ***AICS (Advanced Integrated Control System)***[2], will receive a customer order and assemble and pack a product for shipping within half an hour of order reception. AICS, known around the office as "aches" to represent the pains it is causing, is intended to contribute significantly to improving customer service and customer satisfaction.

Let us use the methodology suggested in this chapter to develop the AICS application plan.

The first two iterations of the methodology described in Sections 3.3 through 3.5 should be completed as much as possible within the two week time constraint. In the planning iteration, an understanding of AICS is developed and very initial choices are made about what portion of AICS should be built, what should be purchased, and what should be based on a re-use of existing systems. A high level model is built in the second iteration to gain further insights and to scope out the work in more detail.

Figure 3-31 shows a very high level view of AICS -- a result of the planning process. The first stage in this system is an order processing system which processes orders for a product. If the specified product is in stock and the customer credit is acceptable, the product is shipped to the customer from the finished product inventory. The product inventory is adjusted to show products shipped. For an out of stock product, a CAD/CAE system produces the design based on the customer specification. The design is then downloaded to a Computer Aided Process Planning (CAPP) system where the manufacturing program is automatically created which shows how the product will be assembled. The CAPP system uses the information about available assembly equipment to generate the process plans. An MRP (Material Requirement Planning) system determines the materials needed for the product. MRP systems use sophisticated algorithms to take into account quantity discounts, vendor preferences, various capacity constraints and factory status. The manufacturing program is downloaded to a flexible manufacturing system (FMS) which consists of an area controller, two cells and several manufacturing devices. FMS also receives a production schedule (how many products to manufacture) and needed raw materials. Because FMS is a real-time system, it must conform to the constraints of real-time control on factory floors.

As can be seen, this system has a combination of application types such as operational support (e.g., inventory manager, order processing), decision support (MRP) and real-time (FMS). At present, the MRP and order processing systems with associated databases exist (shaded areas in Figure 3-31) Other components either need to be built and/or purchased. This observation is enough for the time being. We will consider the details later as different parts of AICS in the case studies in the later chapters. You should work through the steps of the application planning described in this chapter to refine Figure 3-31. In addition, several such systems have been developed in the industry and should be reviewed for additional insights. A well known example is the Dell Computer build-to-order system that builds customized computers. This case study is very well written about (see [Kalakota 2000] for an overview).

3.6.3 Kinion Furniture

Kinion Furniture Company is a small manufacturer of handmade furniture based in Oregon. The company ships around 1,000 orders per year, with each order taking 12 – 16 weeks to produce. Before implementing an integrated information technology system, Kinion was entering all sales orders by hand in its Portland showroom, and then bringing them manually to its manufacturing site in McManneville, 40 miles away. Their existing computer system was limited to DBA Software's accounting and manufacturing application. However, it wasn't networked among the showroom, manufacturing shop, and Vice President Kerry Kinion's home computer (on which the software was installed), making it impossible for anyone but Kinion to conveniently access the information. This lead to a situation where the communication process was impeded and real-time information between the sales and manufacturing departments was non-existent. Ultimately, information was slowed down throughout the manufacturing process, beginning with the placement of the order.

Due to these setbacks and the company's desire to use information systems to further its growth, Kinion contacted the Washington State-based TechnoResources, Inc. In working with the firm Kinion focused on

[2] This is a totally contrived name and any resemblance to another product with this name is totally accidental.

expediting the information-handling process. Based on TechnoResources' suggestions, Kinion implemented a number of software applications and technology services, including Microsoft Office XP, Microsoft Windows 2000 Professional, Microsoft's SharePoint Team Services and a Microsoft Windows 2000 Server. The additional help of high-speed digital subscriber lines (DSL) allowed Kinion to use these programs to access and exchange information electronically at a more rapid pace.

The enhanced information system yielded several important results for Kinion. The Windows 2000-based network provided secure links from the showroom to the DBA system in Kerry Kinion's home as well as to one which was installed in the manufacturing shop. With Office XP (specifically Microsoft Access and Outlook), the company was able to establish an online order entry system also connected to the DBA system. This permitted order and customer contact information entered by sales representatives in the showroom to be immediately available to staff at the manufacturing shop. In addition, a Web-based intranet established using SharePoint Team Services provided an easy way to access information about company projects in real-time. Combined, the suite of technology solutions allowed Kinion Furniture to drastically reduce its order processing time, more easily access customer contact information and track orders, and share information among staff in different locations through a secure, integrated network.

Sources:

- Jasmine Davy, James Drew, Erika Lawrence, Jinxiao Li, Project Report, Fordham Graduate School of Business. edited by Amjad Umar
- Kinion Furniture Company: Office XP Streamlines Operations for Handcrafted Furniture Manufacturer. (2002, March 21) [On-line] Available: www.microsoft.com/office/evaluation/indepth/casestudies/kinion.asp

3.6.4 Garrett Engine Boosting Systems

Garrett Engine Boosting Systems (GABS) is a mid-size division of Honeywell, an industrial conglomerate, based in Torrance, CA. Founded in 1936 by Cliff Garrett, the company was originally called Aircraft Tool and Supply Company and was designed to act as a liaison between East Coast tool suppliers and the fast growing West Coast aviation industry. One year later the company had evolved into a production company providing generic and technologically advanced tools to the aviation industry. Throughout the 20th century, further diversification took place so that GABS now also produces intercooler systems and gas turbines. In the last few years both the high cost of oil and technological advances have helped to increase the GABS customer base substantially. As a response to market demands, the GABS product base has become more complex, offering an extensive variety of new products. Sales have rocketed from $575m in 1994 to $1.4bn in 2001, an annualized growth rate of 17%.

With this evolving complexity, GABS recognized that the existing systems technology would be incapable of growing to meet the customers' demands. "We were still using systems that were up when we were a $300m company… we literally outgrew our business processes," said Jerry Rockstroh, Vice President. The employees could not effectively coordinate with the 350 suppliers to ensure timely delivery of the 90,000 components used in the factories. Without these parts, production was stalled and customers' orders were not filled. Resulting in not only lost sales, but a potentially greater crisis – the shut down of a customer's factory. Because of the small margins in the automotive world, hiring more staff was neither a cost effective nor a long-term solution.

A two-year review of the company's business processes determined that it was vital to turbo-charge the GABS supply chain. With Apexon, a San Jose based technology firm, GABS developed and introduced a new supply chain management system with a simple goal – to give Garrett's suppliers more insight into the firm's internal systems. The system provides GABS and its suppliers with standardized web based systems management procurement. This provides an electronic linkage between the two throughout the process, from the request for quote to the technical drawing.

By linking with its suppliers electronically, GABS has increased the speed of business exponentially, decreased costs and mitigated communication errors. Changes that would have taken several days and required an overnight delivery service before, now takes seconds online. Having a single communication platform reduces costs, communication time and communication errors, as GABS employees are no longer relying on a mix of phone and fax messages. It also allows better visibility for both sides, enabling GABS to ensure the supplier can meet their needs and allowing the customer to check order status. The system also contains a best practices solutions matrix for any problems that may arise in the supply chain promoting better integration with the suppliers.

Garrett's suppliers love the system. Requiring only a web browser, the system is extremely flexible allowing information to be easily downloaded into a variety of systems, including MS Excel. The system is tied to the existing ERP system used by GABS for shop floor scheduling, planning and financial controls. Suppliers are automatically notified of inventory shortages or changes and can even check invoice status online. The system has revolutionized Garrett's business and is being fully integrated into the company worldwide. Management is also studying the possibility of extending the system to include live online design collaboration with its suppliers.

Sources:

- Fraser Owen-Smith, Dylan Schlott, Rusty Ray, Traci Scott, Project Report, Fordham Graduate School of Business, edited by Amjad Umar
- www.egarrett.com
- www.honeywell.com
- http://itmanagement.earthweb.com/ecom/article/0,,11952_906361,00.html

3.7 Chapter Summary

An application engineering/re-engineering methodology pattern is presented that can be customized for a wide range of application engineering/re-engineering situations. The pattern consists of successive iterations, refinements and expansions of four broad activities: analysis, solution architectures, implementations, and deployment/support.

Overall planning should always be the first iteration of the process and should concentrate on business opportunity analysis and risk analysis to determine the best strategy. It is also important to identify the applications needed to support the enterprise, and establish a sketch of an enterprise solution architecture. The IT infrastructure issues need to be considered in a systematic manner with iterations throughout the system life cycle.

3.8 Review Questions and Exercises

1) There is a common saying that "It seems there is never enough time to do a job right the first time, but there is always time to do it over." Does the methodology presented in this chapter address this tendency? How?

2) Show how this methodology can be used to develop applications such as electronic commerce, expert systems, and video conferencing.

3) Compare and contrast this methodology with other IS planning and development methodologies such as OO methodology [Booch 1994, Rumbaugh 1994] and the RMM methodology for hypermedia applications [Isakowitz 1995].

4) Figure 3-13 shows one view of logical application architectures. This view is based on [Kalakota 2000]. Present other views of logical application architectures that show information flows between various applications in an enterprise.

5) Expand Table 3-3 into a decision tool.

6) Expand the cost/benefit analysis to web cost/benefit analysis. What are the similarities? What are the differences?

7) Do a comprehensive survey of decision support and expert systems for e-business

3.9 Additional Information

Allen, P., "Realizing e-business with Components", Addison Wesley Professional; 2000.

Acharya, R., “EAI: A Business Perspective”, eAI Journal, April 16, 2003

Bass, C., “Building a Business Case for EAI”, eAI Journal, Jan. April 2001

Blokdijk, A. and Blokdijk, P., "Planning and Design of Information systems", Academic Press, 1987.

Boar, B., "The Art of Strategic Planning for Information Technology", Second Edition, John Wiley, 2001.

Boar, B., “Practical Steps for Aligning Information Technology With Business Strategies”, John Wiley, 1994.

Booch, G., "Object Oriented Design with Applications", Benjamin/Cummings, Second edition, 1994.

Brodie, M.L., and Stonebroker, M., " Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach", Morgan Kauffman, 1995.

Clemens, P. and Klein, K., "Evaluating Software Architectures", Addison Wesley, 2002.

COST-BENEFIT ANALYSIS GUIDE FOR NIH IT PROJECTS, http://irm.cit.nih.gov/itmra/cbaguide.html, Prepared by: Office of The Deputy Chief Information Officer October, 1998, Revised May, 1999.

Cronin, M., "The Internet Strategy Handbook", Harvard Business School Press, 1996.

Davenport, T.H. & Short, J.E. (1990 Summer). "The New Industrial Engineering: Information Technology and Business Process Redesign," Sloan Management Review, pp. 11-27.

Davidson, W., “Beyond Re-engineering: The Three Phases of Business Transformation”, IBM Systems Journal, Vol. 32, No. 1, 1993, pp. 65-79.

Davydov, M., "Corporate Portals and e-business Integration", McGraw-Hill Professional Publishing; 2001.

Dec, K., "Client/Server - Reality Sets in", Gartner Group Briefing, San Diego, February 22-24, 1995.

DePompa, B., "Corporate Migration: Harder Than It Looks", Information Week, Dec. 4, 1995, pp. 60-68.

Dewire, D., “Client/Server 101: Business process Re-Engineering”, Client/Server Computing, March 1996, pp. 102-103.

Gale,T. and Eldred, J., “The Abstract Business Process”, Object Magazine, January 1997, pp. 32-37.

Gemmel, D., et al, "Multimedia Storage Servers: A Tutorial", IEEE Computer, May 1995, pp. 40-51.

Gordon, V. and Bieman, J., “Rapid Prototyping: Lessons Learned”, IEEE Software, Jan. 1995, pp. 85-95.

Hammer, M., "The Agenda: What Every Business Must Do to Dominate the Decade", Crown Pub; 2001.

Harrington, H., "Business Process Improvement Workbook", McGraw-Hill Professional Publishing, 1997.

Herzum, P. and Sims, O., "Business Component Factory", John Wiley, 2000.

Henderson, J. and Venkatraman, “Strategic Alignment: Leveraging Information Technology for Transforming Organizations”, IBM Systems Journal, Vol. 32, No. 1, 1993, pp. 4-16.

IDC, "The Organizational Impact of e-business: Seven Real-Life Case Studies", 2001.

Inmon, W.H., "Developing Client/Server Applications", QED Publishing Group, Revised edition, 1993.

IBM Corporation, "Business Systems Planning", GE20-0527.

Isakowitz, T., et al, “RMM: A Methodology for Structured Hypermedia Design”, Commuincations of ACM, August 1995, pp. 34-44.

Jacobson, I., et al, "Object-Oriented Software Engineering", Addison-Wesley, Reading, MA, 1992.

Kalakota, R. and Robinson, M., "e-Business 2.0: Roadmaap for Success", Addison Wesley, 2000.

Keen, P., “Shaping the Future”, Harvard Business School Press, Boston, 1991.

Keen, P., “Information Technology and the Management Difference: A Fusion Map”, IBM Systems Journal, Vol. 32, No. 1, 1993, pp. 17-39.

Lederer, A., and Prasad, J., “Nine Management Guidelines for Better Cost Estimation”, Comm. of ACM, Feb. 1992, pp. 34-49.

Lee, J., Siau, K., and Hong,S., “Enterprise Integration with ERP and EAI”, Communications of the ACM Feb 2003

Gable, J., “Enterprise Application Integration**”,** Information Management Journal March-April 2002

Lim, W., "Cost-Benefit Analysis of component-Software", eAI Magazine, April 16, 2003

Linthicum, D., "Distributed Objects Get New Plumbing", Internet Systems, January 1997, pp. 4-5.

Luftman, j., Lewis, P., and Oldach, S., “Transforming the Enterprise: The Alignment of Business and Information Technology Strategies”, IBM Systems Journal, Vol. 32, No. 1, 1993, pp.198-221.

Luftman, J., “Competing in the Information Age: Strategic Alignment in Practice”, Oxford University Press, 1996.

McNurlin, B., and Sprague, H., "Information Systems Management in Practice", Prentice Hall, Fifth Edition, 2002.

Meyer, M. and Zack, M., “The Design and Development of Information Products”, Sloan Management Review, Spring 1996, pp. 43-59.

Mowbray, T. and Zahavi, R., "The Essential CORBA: Systems Integration Using Distributed Objects", John Wiley, 1995.

Nicol, J., et al, “Object Orientation in Heterogeneous Distributed Computing Systems, IEEE Computer, June 1993, pp. 57-67.

Ng, P. and Yeh, R., Editors, "Modern Software Engineering: Foundations and Current Perspectives", Van Nostrand Reinhold, 1990.

Nolan, R., "Managing the Computer Resource: A Stage Hypothesis",Communications of the ACM, Vol. 16, No. 7, pp. 399-405, July 1973.

Rosenberg, D., "Applying object-oriented methods to interactive multimedia projects", Object Magazine, June 1995, pp. 40-49.

Rumbaugh, J., et al, "Object-Oriented Modeling and Design", Prentice Hall, latest edition.

Rymer, J., "Business Objects", Distributed Computing Monitor, Patricia Seybold Group, January 1995.

Sawhney, M., et al, "The Seven Steps to Nirvana: Strategic Insights into e-business Transformation", McGraw-Hill Professional Publishing, 2001.

Sims, O., "Business Objects", McGraw Hill, 1994.

Sneed, H., "Planning the Re-engineering of Legacy Systems", IEEE Software, January 1995, pp. 24-34.

Turban, E., et al, "Electronic Commerce: A Managerial Perspective", Prentice-Hall, 2000.

Vasudeva, R., “Reusing Business Objects”, Object Magazine, January 1997, pp. 32-37.

Venkatraman, N. and Camillus, J., “Exploriing the Concept of “Fit” in Strategic Management”, Academy of Management Review, Vol. 9, 1984, pp. 513-525.

Ward, J., and Griffiths, P., "Strategic Planning for Information Systems", Wiley Information Systems Series, 1996.

Whyte, W., and Whyte; B., "Enabling e-business - Integrating Technologies Architectures & Applications", John Wiley & Sons; 2001.

Zachman, J.A., "Business Systems Planning and Business Information Control Study",IBM Systems Journal, Vol. 21, No. 1, 1982, pp.31-53.