

Chapter I

Engineering Issues in Internet Commerce

Xue Li

University of New South Wales, Australia

ABSTRACT

Engineering Internet Commerce is about building web-enabled enterprise information systems to carry out business transactions over the Internet. This engineering task is related to three aspects: the requirement specification, the Internet technology, and the development methodology. In the requirement specification, the business analysis and design is conducted to create a semantic business model that will reflect both the business and the system requirement. With the Internet technology, the modern information technology infrastructure is investigated in order to transform a business model into an implementation model. The system analysis and design will be performed and the architecture issues should be discussed. With respect to the development methodology, an efficient way to build enterprise information systems is addressed. This chapter is to provide an overview of the problems, concerns, and the background in an effort to rationalize the Internet Commerce Engineering.

INTRODUCTION

From an engineering viewpoint, we are dealing with three worlds, the **real world**, the **perceptual world** and the computerized **virtual world**. The real world is every thing existing in the physical world. The perceptual world exists in human brains. And the virtual world is existing in the Internet. The engineering activity is to transform ideas in the perceptual world into the real world (e.g., electronic engineering) or into the virtual world (e.g., information engineering).

The real world is objective. It changes and evolves. The perceptual world is subjective. It is individual and cognitive. It is configured in the best interests of human desire and survival. The perceptual world is intangible and reflects human understanding of and interactions with the real world. To an enterprise, the perceptual world is an asset that will

control and guide business planing and strategic decisions. On the other hand the virtual world is reflective. It is an implementation of our perceptual world. In the virtual world, the digital signals are interchanged as an efficient way of information exchange.

The concept of the three-world is to help the understanding of the relationships between the objects that are considered in the Internet Commerce (IC) Engineering. The success of business is becoming more dependent on the successful applications of information technology. An unrealistic perception of the business world may result in unfruitful business systems on the Internet, and consequently cause business failure. In general, the perceptual world should be proactive, that is to interpret the real world in the best way to satisfy business goals.

This chapter is to address the high level issues of the IC Engineering for building IC systems. The IC Engineering is considered in three aspects: the requirement specification, the Internet technology, and the development methodology. We will discuss the problems, concerns, and background related to these three aspects in an effort to rationalize the IC Engineering. Figure 1 illustrates this idea.

The **requirement specification** is a process that generates business requirement specifications and other necessary documents such as the explanatory files. The outcome is regarded as a business model. The process is for the business analysis and design that maps business information needs to the Internet technology.

The **Internet technology** is characterized by the Object-oriented technology, Internetworking, and the Client-server architecture (Umar, 1997). The system analysis and design techniques are used to derive an implementation model that is a mapping between the business model and the technological details with regard to the system components and the architecture.

The **development methodology** is applied to give a roadmap that shows the path of the system development. It provides answers on how to apply what technology on which business applications. There is a phenomenon that both business and the Internet may experience rapid changes. This requires the growth management to be built into the system development. Currently many development methodologies rely on the underlying software tools supplied by the major software market players.

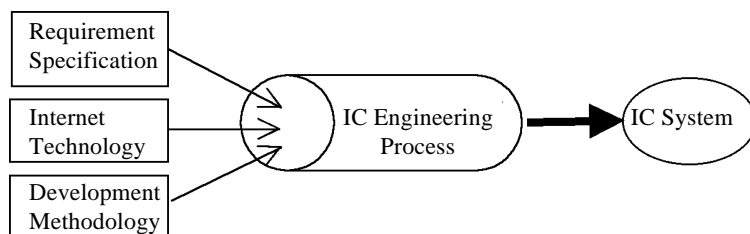
An **IC system** is an information system that provides web-enabled services including:

- the operational business transactions carried out over the web,
- the ability to maintain system security and data integrity in a web environment, and
- the strategic business planning and decision support in advanced web-applications such as data warehousing, on-line analytical processing, data mining, and enterprise knowledge management.

The growth of the Internet has been exponential in every aspect, including its size and the capacity. Many businesses are now engineering their information systems on to the Internet. Despite

the Y2K problem in the legacy systems, the information system evolution towards the Internet is not a trivial problem. The **IC Engi-**

Figure 1. The Engineering Process of Internet Commerce Systems



neering is about how to build a web-enabled enterprise information system for businesses. The engineering may also need to deal with the legacy systems that were developed without regards to the Internet. The trend has already begun that many businesses are now starting using web sites for their information needs. More and more software engineers are now employed to bring legacy systems on to the Internet.

The understanding of business is fundamental. However, it sometimes is not simple because the understanding of the applicability of the technology is a progressive process. It is like a marriage. It requires the mutual understanding between the business requirement and the technology applicability. It may require a recursive refinement process during the system development. It is evident that the new innovative technology may impact on the business and change the business process. Therefore, the requirement specification in the IC Engineering can be affected by the Internet technology. If the requirement specification is subject to a recursive refinement process, the development methodology should also play a role in the requirement specification. In fact, these three aspects of IC Engineering are overlapped. For example, a prototyping methodology may be applicable to the system development to accommodate the recursive refinement of the system requirement specification. In this case, the prototyping is a reflective process that incorporates the system requirement dynamically.

The IC Engineering should integrate the requirement specification, the Internet technology and the development methodology into a progressive and reflective process. This is mainly because that we are now in an unprecedented fast-changing world. The consistency between the real world and the virtual world means a successful engineering that is very much dependent on the reflectiveness of our perceptual world. Hence in building the IC systems, we need a tight connection between the fast-changing world and the system built. Comparing to the classical software engineering approaches, the IC Engineering is a process incorporated with the ability of dealing with the fast-changing environment.

The requirement specifications will be discussed in next section. Then the benchmarking will be discussed as business-critical specifications. We will give some emphasise on the current IC technologies. During the discussion, we will compare and contrast some different IC system architectures. The methodology issues are then discussed. We will give an overview on the implementation techniques towards the end of this chapter. The conclusions are presented at the end of this chapter.

IC REQUIREMENT SPECIFICATION

One of the challenges in the Information Age is that consumers are now accessing same information via web. This gives consumers a larger verity of options and thus businesses are facing a bigger and more competitive market.

The IC requirement specification is a process of the business analysis and design that is to create logistic models, process models, or semantic business models for the business-trading environment. This section will not be discussing on how to create those models (see Davis 1993 and Kotonya 1998) but rather on the explanation of the problems and concerns in doing so.

In terms of business transactions, there may be mainly two types of IC systems on the Internet, **business-to-business** and **business-to-customer**. In a business-to-business system, most transactions are automated by using EDI (Electronic Data Interchange) or EDI/XML (Plapante, 1998). The network traffic is predictable and stable. The transactions are mostly in batch mode. In a business-to-customer system, most transactions are manual. The

network traffic is unpredictable and dynamic. Also the most transactions are in on-line mode.

We identify three major concerns in the IC requirement specification, the **business vision**, the **assessment criteria**, and the **system viability**. A successful business must have a vision. We will analyze two cases to demonstrate the importance of the business vision. The perceptions that developers may have in their minds may be business-centric or customer-centric. Then, we discuss the assessment criteria for benchmarking the web-enabled enterprise information systems. We identify four factors for the business-critical applications, which are availability, reliability, security, and performance of the systems. Finally, we discuss the system viability of a business application in terms of the growth management as the ability of coping with changes.

Having a Vision: Business-Centric versus Customer-Centric Applications

A business vision is about the perception that we possess towards the success of a business. To have a vision is to identify the success factors and foresee the changes. It will fundamentally affect the design and the implementation of a system. We may view a system in many different ways. For example, it can be viewed on what and how a system does, or on changes a system can bring. In this subsection, we compare and contrast two opposite views on a business information system, from within the business and from its outside. The concepts are the business-centric systems and the customer-centric systems. They are treated as a case study for establishing a business vision in the IC Engineering.

Business-centric systems are those systems specialized in business activities. They may require prerequisite knowledge and training of users. Traditionally, non-web business information systems are almost all business-centric because consumers do not need to directly interact with the computer systems. Operators of business systems are trained and designated. There is no need to create interfaces especially for the targeted consumers who want to visit the business via computer systems. Business is conducted normally through facsimile, telephone, mails, or face to face. In business-to-business type of systems, business-centric application still has its *raison d'être* because there are mostly still machine-to-machine batch transactions and the specialized interfaces. However when the Internet comes into play, the business-centric style increasingly becomes a problem to the general users. **Customer-centric** systems are applications that are designed in a way that is to put the customer's needs first. The system will try to satisfy customer in all possible ways. Therefore a customer can use the system without much knowledge of computing. And a customer can get as much information as he/she can.

Let us discuss an example. We may have retail businesses to be put on-line for web users. We discuss this example in two different viewpoints: one is from a customer-shopping viewpoint, the other is from a product-sale viewpoint. From this example we try to show that the customer-centric development is important to the IC requirement analysis.

Case One

In a customer-shopping oriented design, a business is shown to the customers to view, to search, and to buy their products. General shoppers will be able to log onto the Internet shopping service to do the web shopping with a minimum knowledge of computer. We call it as Shopping World.

Much like the real shopping activities, the Shopping World may be a web service that

provides users with a walk-through experience on gaining information or doing on-line transactions for geographically locatable objects of the businesses. For example, the system may represent a large shopping center for general shopping lovers with the pictures and the floor plans. It provides information for the shoppers in a way that they would see in the real-world shops. All shops are presented in a cross-referenced structure. The customers are able to *walk* into the shop and browse the items, order and pay for the products. Of course there will be many questions or helps that may be asked from customers to a shop. The Shopping World may also use a search engine to answer questions such as where do I find dry wash for my suit? Which shops sell the screws sized 1.5mm in diameter and 5 cm in length? In a way that the Shopping World can help people to find where to buy things.

Case Two

In a product-sale oriented design, individual shops may have their own web sites. A shopping center may then have a general web page to link them together. The whole service is then presented as a list of categorized businesses. This requires a customer to understand the words and the categorization system used by that web site. In addition to a bombardment of advertisements, a customer may have to rely on a search engine interface to find what he/she needs. This search process may have to go from general to specific in terms of the categorization of the goods. In other words, the business is conducted as a catalogue based on-line ordering system. Whenever a customer finishes a business with a shop, she/he then has to re-start a web journey again for the next item to buy.

The main problem for the case two is that it is the business-centric. It puts the **shops sales** information ahead of the **customers shopping** information. The information is presented by the system in a way that may be suitable for the individual shops to maintain their own web sites but difficult for a customer to do shopping, who may have little knowledge on sales business. A shopper may need to buy things from many shops in a way she/he wants but not in a way the shops organizing their businesses. For example, a business-centric web site can restrict to a top-down presentation of their business information using a rationalized general-to-specific fashion in either search or presentation services. While a customer-centric web site may disregard the rationale of the categorization of goods but just use a specific-only way to search and use the services. It is common that one may buy some items without knowing what categories they are. Currently we can see both kinds of web sites on the Internet. For example, *shopping carts* as a web-shopping concept for the shopping lovers has been used in many web sites as a similar case to the above-mentioned case one. Unfortunately we can still see many businesses web sites that are in situations of case two. They are still very much relying on the search engines to discover them. It is revealing that the customer-centric business systems are taking advantage from the revisitors or the visitors who receive the message passed by the satisfied pre-visitors. Many successful web-enabled businesses have shown that their success is mainly because of the satisfaction provided to the customers. The trend shows that the business-centric web applications are phasing out.

On the other hand, the abuse of the web-technology in many systems does not provide customers with a simple method one normally uses in everyday shopping. For example, many search engines provide a multi-threaded search result (represented as a tree) instead of a simple linear search (represented as a list). Yet the simple search method may imply a

powerful back-end help that can be dynamic and context-sensitive on various problems during the shopping, such as the locations, prices, or on-sale information.

We can view the retail business as a two-side story that is about the customers shopping activities and the shops' sale effort. So a multi-threaded search may help to maximize the opportunities of the sale but less pleasant for a shopper. Instead, providing shoppers with a linear and pleasure-driven search may give a better imitated shopping experience that is much an enjoyment when she/he would be going for a real-life shopping.

There are still many questions yet to be answered to distinguish a business-centric and a customer-centric system. We discuss some of these aspects in next section as web databases and the data mining. The investigations on general *pull* technology in the shopping activities or the *push* technology in business sales may also improve the satisfaction of the web-shopping lovers.

Benchmarking: Business-Critical Applications

Successful IC Engineering requires an assessment benchmark that brings the confidence and better understanding to the IC. The building of business-critical applications requires the quality assurance. Under these assumptions, we identify availability, reliability, security, and the performance of a system as the critical factors for business applications. One needs to start from the analysis of business functions and specify the requirements for the system. A top-down design methodology can then be applied to bring a perfect marriage between business functions and Internet technology.

Availability

A system is available may imply that it is accessible, scalable, and attractive. Hence the availability of a system is a measurement of its accessibility, scalability, and attractiveness. The accessibility of a system harnesses the Internet technology for its business solutions. On the one hand, a system should be accessible for supplying a range of business services. On the other hand, a system should be accessible for providing a variety of business transactions. There should be no limit on supporting the businesses whether they are selling tangible or intangible goods, whether businesses are located in a shopping mall or as a warehouse-based distributor, whether they are the business partners in a suppliers chain or in a virtual enterprise. Moreover, different types of users of a system may have different purposes, such as for front-end business operations, for executive information needs, for decision supports, or for enterprise knowledge management. The accessibility can be achieved by a unified "on-the-web" standard user interface. It may provide users a great help on accessing information but may introduce security problems and other concerns. The Intranet and Extranet technologies can be applied as a leverage of business functions on the web.

The **scalability** of a system means that it is capable of being tailored to suit different environment or is flexible to satisfy different business needs. When considering the environment, the system may be scalable for deployment on mainframe machines, on personal computers, or on portable devices such as mobile phones. In this case the software should be platform independent and capable of dealing with different types of network traffics, and capable of processing different volume of data. When considering the business needs, the system may be scalable for different customised software configurations so that the system is installed to suit for particular business needs.

Availability may also refer to the **attractiveness** of the web site. The metrics of attractiveness consist of two factors: the publicity and the satisfaction. Both of these two

factors are important for the system viability. Without the publicity, the system will have no users. Without the satisfaction, the system will be useless. The analysis of the web site attractiveness may start with the statistics of the user numbers. It may give the numbers of the first-time visitors, second-time visitors, or the number of customers who have done the business with the web site (and come back again). On the other hand, there may be many attributes that affect the attractiveness of the web site. The web data mining technique may be used to discover the significance of those attributes (see next section).

Reliability

Reliability is about the robustness and soundness of a system. It can be operational related: so the system can be recovered from either software or the hardware crashes. It may also require the backups in order to recover from environmental disasters. On the other hand, it can be application related. It is very hard to have a system bug-free. In many cases the error is not easy to be classified as whether a logical error or a system error. A logical error could be a design error or a data error. In many cases, it may give no error message. A system error could be an operating system run-time error, network traffic problem, or a software-interfacing problem. A system error message may not indicate the real problem but just a report on the failure of a system operation. When an error is discerned, it is sometimes necessary to repeat an error in order to fix it. Theoretically, any error should be repeatable. However, the ability of repeating an error is very much dependent on the understanding of the error.

There are generally two ways to provide application reliability, the **verification** and the **software test**. These two ways may compensate each other. The verification is a process that uses the theorem proof technique to check the system specification against the system requirement, while the software test is a process that checks the system implementation against the system specification.

In system verification, a reasoning machine is used to reason the artifacts in the specifications for achieving the goals of the requirement. The verification process is based on a formal specification language that can provide the formal syntax and semantics. A specification language can be either executable or non-executable. A non-executable specification language may provide a non-ambiguous semantic specification and an implementation-independent document. It requires a translation process to perform the verification in an executable programming language. While an executable specification language may have a built-in verification process and can provide an executable program file for a system in a target computer language. An executable specification language may also reduce the software test work because the built-in verification process can be executed automatically whenever the target system program is to be generated.

In software test, the system is tested in many ways, such as the static and dynamic tests. **Static tests** are performed in a bottom-up fashion. Step-by-step, all components are tested before they are put together. Each component is tested against the specifications and their pre-post conditions. A *window-technique* can be used to exhaustively test all functions of a system for all possible input. The static tests are *ideal and deterministic* in a sense that the test environment is known and the output is pre-defined. An error can be regarded as a kind of system side effect. The complexity in static tests is that the system side effects are inherent in the system implementation. A metaphor of proving a side-effects free system is “to know what you do not know”. Another problem is that it may be difficult to feed a system with all possible input permutations. Although it is sometimes infeasible or unnecessary, the

decision on a non-exhaustive test can be subjective and problematic, particularly to the reused system software components.

Dynamic tests target on the demonstration of the system overall functions and the performance. Therefore, the process should be top-down. The system is to be tested in a dynamic and changing environment in order to find system problems from general to specific. Referring to a system's architecture, the system dynamics is considered in two dimensions: (1) the major business data flows between major system components (e.g., between clients and servers), and (2) the main business transactions in each stream of the data flow (e.g., the transition from product browsing to on-line ordering). By using this two-dimensional approach, the *bottleneck problems* can be isolated in terms of when, where and what the problem manifested. For example, a crash (or stress) test can be used to find out the most vulnerable part of a system. In this case, the system is engaged in an extreme situation in either maximum system load (of resource usage including the network traffic) or the maximum system charge (of log-on users or on-line services). The complexity of the dynamic tests is that the tests are mostly conducted in a test environment and the simulation of the system dynamics may be incomplete or incomprehensive.

Other issues may also affect the system availability, such as the virus protection, the network security, and the system administration including recovery, security control, performance tuning, etc. In dealing with a changing world, the availability can also be affected by the system viability in coping with the changes. Over all, the availability is the ability that keeps the system constantly alive on the Internet.

Performance

The performance is a dynamic factor of the system. It is related to three aspects of a system. Firstly there is a **cost-effective** decision made in the system design phase that is to bring a best integration of the software and the hardware at a cost within the system development budget. This decision is meant to give the system performance a base line. Secondly, there is a system architecture problem. In a client-server paradigm, the network traffic and the network dynamics are both considered in providing a balance in the principles of the distributed computing. In considering the network traffic, the performance is a problem of the **distributed database management** in order to minimise the network traffic by using techniques such as RPCs (remote procedure calls) or data fragmentation and data replica cache techniques. In considering the network dynamics, the performance is problem of the **network management** that is associated to the network topology and the configuration to maintain the network reachability and the high-level interoperability. Thirdly, the performance is a system administration matter in the performance tuning process that is to find a balance of the trade-offs between the system storage space and the system responding time in a dynamic business environment. Within a changing environment, the performance problem may involve above three aspects recursively.

Security. The security problem of an IC system is about the authenticity, confidentiality, and data integrity for the web-enabled business transactions. It is very important and crucial to an IC system. This problem is inherent in the Internet security and deserves a detailed discussion elsewhere.

A Viability Study: Growth Management

The growth management is about the strategies on how to deal with changes. The *business reconceptualization* is discussed in many research papers on how to re-build

business in facing the opportunities and challenges in the Information Age. Here is a to-do list:

- Anticipate and monitor changes in the business environment.
- Assess the changes and find out what the impact could be in both aspects of losses or gains in business.
- Understand the reason of changes and analyse the triggers or conditions of changes.
- Build the business model that is able to incorporate changes.

If the business model cannot reflect and represent the changes, the business model then needs to be changed. There are two ways to do so. (1) Change or redesign the business model. This involves the *reverse engineering* process. (2) Use a trial-and-error approach. This is to modify the system in order to adopt a change. The second way has been used by many systems to cope with the Y2K problem.

The growth management reflects the viability of a system. The strategies used to deal with changes can either be exploitative or defensive.

IC TECHNOLOGY

The IC technology is the application of the Internet technology on the building of IC systems. In this section we discuss three prominent issues: the web architecture, the web data databases and data mining, and the web intelligence.

Web Architecture

The web architecture concerns how the information flows among the system components and how the system components are organised. We will firstly give an overview on a progressive and reflective procedure of building the web business applications. This idea shows how the web architecture is established. Then, we introduce the basic concepts of front-end and back-end system components. Based on those concepts, the client-server paradigm will be discussed in the context of the information flow and the system software deployment. Finally we consider the system architecture in contrast with the business structure that can be *vertical* or *horizontal* in terms of the business connections.

Progressive and Reflective IC Engineering

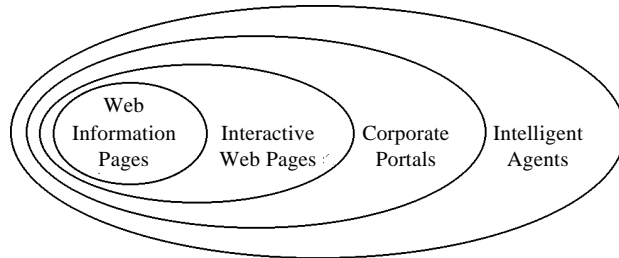
The globalisation and deregulation are two main drives to the changes of the business environment. The impact from the growth of the Internet has also accelerated these changes. The IC Engineering is challenging these changes by using a progressive and reflective process to build the IC systems.

Figure 2 is used to illustrate the idea that an IC system can be developed progressively with reflection of changes. The initial IC system may be a few primitive web information pages. Then the interactive features may be added on to establish a basic trading environment to allow customers to search for or purchase the goods and services. The operational business transactions are supported at this layer. Web-enabled database applications are deployed at this layer. When a trust has been built on the system, a full-fledged business information system can be built on the web using **Corporate Portals** (Finkelstein, 2000). Corporate Portals are the web applications that provide a unified information gateway for distributing, analysing, consolidating, and managing information across and outside an enterprise. Corporate Portals are also called Enterprise Information Portals (EIP). The trend has already started that the Corporate Portals are used to unify the effort of the enterprises

to integrate their information access methods. For example, Corporate Portals can be configured with data warehouses to enable different users to access information for their needs, such as Executive Information Systems (EIS), Decision Support Systems (DSS), On-line Analytical Processing (OLAP), and

Knowledge Management Systems (KMS). Above the Corporate Portals layer, the **Intelligent Agents** (Wooldridge, 1995) can be built to retrieve, discover, reason, or deliver the knowledge for various needs. They can be used for business purposes such as buying and selling products, or be used for strategic purposes. For example, Intelligent Agents can be proactive to mine data in data warehouses in order to discover the trends or changes in a business environment; or be reflective in order to solicit the expert advises to review business policies. Although the enterprise knowledge is available at the Corporate Portals layer, it is about its accessibility and maintenance. While at the Intelligent Agents layer, the enterprise knowledge is available with the interests of its manipulation and application.

Figure 2. The Progressive and Reflective IC Engineering



Front-end and Back-end Concepts

All web-enabled applications have two ends, front-end for the user support and the back-end for the system support. The front-end has the responsibility to:

- provide user interfaces,
- implement the representation logic,
- implement the application logic, and
- specify the interfaces with back-end components.

The back-end includes all those invisible system components that must be used to support the functions of front-end. The back-end may include functions of:

- database management,
- data communication,
- multimedia management,
- security control (e.g. firewall).

It is important to distinct the system front-end and the back-end components. Front-end is application-oriented and business specific. It should be user-friendly, flexible, and competent for business functions. The tools used to develop front-end should be easy to learn, easy to use, yet functionally sufficient. While back-end is implementation-oriented and business independent. It should be highly efficient, standard, and versatile for supporting different requirements. For example, the front-end of an IC system can be built based on XML, VRML, or SMIL (Synchronised Multimedia Integration Language, from W3C, www.w3c.org) tools, and the back-end can be built based on CORBA (from ODMG) or ADO (ActiveX Data Objects, from Microsoft®), for relational or object-oriented databases. By the distinction between the front-end and back-end components we can develop an IC system separately and we have a better chance to organise the system for different business environments. As a result of the distinction between front-end and back-end components, the programming tasks are also discussed as **client-side** and **server-side** programming tasks (Morrison, 2000).

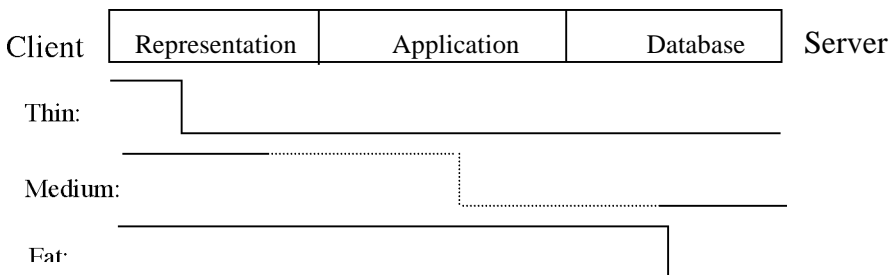
Client and Server Paradigm

In a client-server application, a client can initiate the information request and expect the server to reply. Here are two concerns: (1) the information flow between client and server, (2) the software deployment between client and server. The web applications are often decomposed into **two-tiers** or **three-tiers** (Umar, 1997b). In two-tiered architecture, when a web application is to be deployed on the Internet, there are at least three ways to deploy the software components: a *thin client*, a *medium client*, or a *fat client*. This is decided based on the principle that to minimise the information flow between client and server and to maximise the system performance. Figure 3 is used to illustrate this idea.

As a thin client, the software is deployed in a minimum. A simplest case is that the client only needs a web browser to access the service. The representation logic is presented in the web pages specified by a URL. Other kind of thin clients may have to install light client-control software where the web-browsing facility is embedded. In this case, the user may not see a standard web browser but a customised business-specific application. The problem with thin client is that it does not support application logic and database logic. So, the transactions are all performed on the server side. This may cause frequent accesses to the server for fetching or processing data. Then the responding time may fluctuate according to the status of the server that may be heavy loaded. The advantage of the thin client is that the light client software deployment may reduce the cost to the user and increase the accessibility from the general web users.

As a medium client, the software is deployed based on a balanced tuning of two factors: the network performance and the distributed database management. The network performance concerns the dynamics of the network traffic. From a network management viewpoint, the software deployment should make the least possibility of bursting data transmissions and should request for the least number of data communication channels. This requires the client software to be able to buffer the data in advance to reduce the possible data bursting; or to cache the data whenever they are buffered to reduce the chances of further requests of data communications. From a distributed database viewpoint, the data processing should be as local as possible. This requires the client software to locally support the representation logic, or the application logic, or even the database logic. It also requires the database fragmentation or the database replication in a way that can reduce the chances of moving data around. The medium client implies a complexity in the design of a balance between the front-end and the back-end software deployment for both client and the server. It also introduces the complexity of the database management. However the advantage is obvious that the client software is deployed in the best way to maximise the system performance.

Figure 3. Software Deployment of Client-Server Components



As a fat client, the system functions are mostly housed on the client side (e.g., automatically downloaded as .JAR files in an HTML page). It is sometimes a good idea for the semi-offline clients or mobile desktop clients. In this case, the client is responsible for the virtually all application processing. The request to the server side is limited to a minimum, such as the replica consistency maintenance, data dictionary or the data validation services. A fat client may make IC Engineering more complicated and expensive.

In three-tiered architecture, an application is decomposed into three levels of control: client, web server, and backend database management system. The middle-tier application is mostly referred as a web listener plus the firewall control. The three-tiered client server architecture provides a better balance between the application control and the system performances. More detailed discussions can be found in (Dickman 1995 and Umar 1997b).

The client-server paradigm can be incorporated into two kinds of IC frameworks, the **site-based** or the **agent-based** frameworks. In a site-based framework, the information flow is basically in a request-reply pattern: the client asks for information, then the server replies. This request-reply pattern is executed as either RPC (Remote Procedure Call) or RDA (Remote Data Access). In this pattern, the essence is the data and data processing. It is called site-based because the value of information is decided at the site: the one who is asking for information. In an agent-based framework, the information flow is basically in a goal-satisfying pattern: the agent travels to the server and interacts with the server for the goal to be achieved. The agent carries the goal and the knowledge. The server provides the inference mechanism and information for agent to reason out the goal. In this pattern, the essence is the knowledge and goal achieving. It is called agent-based because the agent, who has the autonomy to make decisions on behalf of its creator, decides the value of information. An agent is characterised by its mobility, intelligence, and autonomy.

Vertical Structure versus Horizontal Structure

There are two forms of the IC systems in terms of business connections, the **vertical** or the horizontal. IC system architecture is said to be vertical if it is an individual business and built as an isolated web application totally relying on the advertisement of its URL or on the search engines to publicize it. This is particularly the case for small and specialized businesses. IC system is said to be **horizontal** if it is a web application system that is cross-linked with other web application systems in a way of its business nature. Many IC systems building a network of trading partners as a *virtual enterprise* can be regarded as horizontal structured systems because there are multidimensional connections with other different businesses.

To gain information from vertical structured web sites, a web surfer may use a search engine that often returns a formidable list of result in a non-preferable order. Some very skillful web surfers may find their expected web sites. The horizontal structured web sites, on the other hand, is more accessible in a sense that businesses are linked together in cross-references. Unfortunately, many of the current IC systems are still vertically structured and struggling for their survival. Although there are many ways to make IC system appear more attractive or more functional, a vertical IC system is difficult to survive until it is well known.

To make vertical systems horizontal, we need to discover or identify the natural relationships that exist with the businesses in the real world. For example, those individual specialized businesses may be geographically related (in the same city or suburb), ISP-related (i.e., registered in the same ISP), or business partners. For the best interests of a business, it is important to make the system horizontally connected.

Web Database and Data Mining

This subsection is to consider the data and their meanings in a web environment. The web database is discussed in terms of the interconnectivity and interoperability. The web data mining is considered in terms of the knowledge discovering of **rule-based** patterns and **topology-based** patterns for the publicity problems and the attractiveness problems of web sites.

Web database

The database management systems (DBMS) provide functions for storing and accessing data. The DBMS is characterised by its platform independence and application independence. The platform independence allows the DBMS to be implemented on different operating environments, so that the database can be scalable and portable. The application independence allows the DBMS to be shared by different applications at different abstraction levels. The DBMS can be measured in terms of data integrity, security, accessibility, and reliability. The distributed (DDBMS) is a system that can handle the data fragmentation and replication over data communication networks.

IC systems need database support as the back-end. This challenges the DBMS to have its data web-accessible. Web database is a database that has its data accessible for web applications (Morrison, 2000). So the data can be inserted, retrieved, or updated from within a web addressable unit (e.g., an HTML document). A web addressable (i.e., hypertext-linked) unit is given as a URL (Uniform Resource Locator) that can be a document, a program of a business transaction, or an interactive user action (e.g., sending email or downloading software). There are generally two different ways to make a database web-enabled: a **connection-based** approach or a **content-based** approach. In the connection-based approach, a standard protocol is proposed to specify an interface between data and programs. Examples can be the CORBA architecture using a Java-based programming environment to manipulate data over the Internet; or the ADO architecture (Gutierrez, 2000) using Microsoft® proprietary ActiveX technique for the data interfacing in the network environment. In the content-based approach, a standard protocol is proposed to specify the meaning (content) of the data for different applications. Examples are the ICE (Information and Content Exchange, www.w3c.org) and XML/EDI (www.xmledi.com). In this approach, the data need to be extracted from the database according to the content-based protocols before they are exchanged. So the interaction is performed in terms of the document exchange instead of the interfacing.

The main difference between the connection-based and content-based approaches is that the former is computing-centric, while the latter is business-centric. The connection-based approaches are considered in terms of the independence between platforms and applications, so those application programs can share and exchange data over different database architectures. The essential point here is the interconnectivity of the databases. The content-based approaches are considered in terms of the independence between business data and business applications, so those different business applications can share and exchange enterprise information for the suppliers chain, purchasing, manufacturing, shipping, auditing, and other business transactions. The central point here is the interoperability that makes the data understandable to each other applications.

In a top-down design viewpoint, the content-based approaches should be considered and evaluated before we consider the web database implementations. However the engineering concerns can always be constrained by the availability of the web database techniques for their ability to support the content-based approaches.

Web Data mining

The web data mining (WDM) activity is to automatically discover rules, patterns, or associations from the web-collected data. The WDM can be applied on two categories of data recorded: the business transactions and the web-user patterns. The purpose of the WDM can be either for the improvement of the business functions or for the enhancement of the web site attractiveness. The data mining on the business transaction data is the traditional data mining while the data mining on the user-web patterns is an emerging research topic, for which the further discussion is given as follows.

For the understanding of web user patterns, current research follows two directions: computing *rule-based* patterns (Mobasher, 1997), or computing *topology-based* patterns (Chen, 1998, Lin, 1999). In a rule-based approach, the input of the data-mining algorithm is a relational database and the output is a set of association rules that report the intrinsic relationships between data items. In a topology-based approach, the input of the data-mining algorithm is a set of directed graphs that represent the user accesses to the web business components and the output is a pattern of a specific topology, which represent the frequent user traversal in a web environment. Business components are the web addressable units. By using the topology pattern, we can find out the causal relationships between the frequently visited web addressable units and the scenario of the web-visiting activities.

For the approach of rule-based patterns, we discuss the **popularity problem** of web sites (Li, 2000). We identify four kinds of visitors regarding a web site: the first-time visitors (FV), the second-time visitors (SV), the visitors who have elected to do business with the web site (FB), and the visitors who come back to do the business again (SB). The number of FV may reflect the successfulness of the web publicity. The number of SV may reflect the successfulness of the web site construction. The number of the FB may show the usefulness of the web site. And the number of SB may demonstrate the overall attractiveness of the web site including the successfulness of the business functions.

There are many questions to be answered in order to understand the user behaviour. We need to identify the factors that would affect the determination of the numbers of FV, SV, FB, and SB. Thus, a database is needed to record the data about the visitors' profile as well as the data that attribute web sites. Hopefully then we may be able to tell what kind of web sites may attract large numbers of four categories of the visitors. By using a system log, web site visitors can be classified into one of the above four categories. For example if such a database is available, we may be able to use data mining algorithms to find out that if a "free email account" is a good way to attract the FV for a web site. Or we may be able to verify that "the most effective way to publicise a web site in retail business is the local newspaper advertisements", etc. So the rule-base data mining in this case can be used to find a cost-effective way to attract the large number of the first-time web site visitors.

It may be useful to understand not only how but also why user visits the web site. We now discuss the **satisfaction problem** of web sites as an example of the topology-based web data mining approach. We may consider the satisfaction problem in two phases: how and why. If we can understand how users behave in visiting web sites, we may have a better idea to organise the web site. If we know why user is attracted to visit a web site, we may be able to make web sites more satisfactory to the user needs. We assume that the metric of the satisfaction is the frequency of web visits. A larger number of the frequency implies a higher level of the user satisfaction. By viewing the frequently visited topology pattern of the hypertext-linked web business components, we may find out what business components are visited frequently and how they are visited.

We assume that a system log is used to record the detailed information about web accesses in a local system. The system log should provide the information on the following items: the user identification, the current focused web business component, the duration of the visit, and the previous visited business component. The system log then can give the history of the web access activities of users. By using the user identification, we are able to group the web activities in terms of users. By using the duration of transitions, we may know the average time spent by user in that function. By recording the focused and the previous visited business components, we can draw a directed graph of the web visit. The nodes in the graph represent the web addressable units and the edges represent the transitions of the visits. The edges are weighted by the frequency of the user's visits. Then we can collect a set of graphs from different users. The task now is to compute a topology pattern that is the most common to all graphs in the set. This topology pattern would show how users have visited the web site.

After the topology pattern of the web access is discovered, the further study is to find out why this pattern is. This triggers the further investigation on the properties of the topology pattern. It should reveal the scenario of the user visit, the causal connections between the web addressable units, and orientations of the user visits towards the attributes of those business components. This further study may require a new phase of the web data mining that need to collect the topology patterns from all different web sites that use different web addressable units for different business components. When this kind of database becomes available, we would be able to run the data-mining algorithm again to find out an interpretation for a frequently visited web site.

The research on web data mining has just started and need much more attention.

Web Intelligence

Web intelligence is to apply artificial intelligence in the WWW (World Wide Web) environment. To this end, the intelligence refers to the knowledge and the application of knowledge in problem solving. Knowledge can bring change and knowledge itself may change. If we view the Internet as a network of servers and clients, we can then see that the collection of servers and clients is a representation of the virtual world that reflects the image of our perceptual world. In the virtual world, knowledge can be discovered, stored, transported, and applied.

Intelligent Search Engine

In information search, the problem is to search an information base thoroughly. The fundamental question is to understand at the first place about what is to be searched. So, the key issue is to decide the search criteria in terms of both denotations and connotations provided by user explicitly and implicitly. We may view the Internet as a huge distributed information base, the searched result can then be said complete if the search goal is decidable. A search goal is decidable if all implicated meaning can be explicitly expressed by the search syntax. Figure 4 shows a complete search paradigm on the Internet. There have been three generations of the Internet search engines.

The **first-generation search engines** use a simple key-word search methodology. It has two stages. Stage one is to use a software agent program (called *spider* or *robot*) to fetch information from the Internet. Then a database is used to store the categorised information. Stage two is to search the database to provide the resource (URL) addresses according to users request. The first-generation search engines are characterised by a fetch-store database approach. It is sometimes difficult for a use to digest a vast amount of returned

search result. Depending on the quality of a search engine, a user may have to use different search engines to get the information.

The **second-generation search engines** use a *meta-search* technique that uses software agents to simultaneously search multiple first-generation search engine databases and bring back the data and then compile them dynamically to satisfy users request. The second-generation search engines are characterised by a fetch-processing analytical approach. The advantage of the second-generation search engines is that they pay attention to the content of search result and present the content in a way that is best suitable to users. For example, some search engines can provide a multi-dimensional URL graph as a user interface to allow user to pursue the further search. A *fish-eye* technique can be used to *zoom in* or *zoom out* in a URL graph. The second-generation search engine can be content-based in a sense that the *search pattern* can be recognised in the search process. The meta-search for the universal resource discovery is a research paradigm that is aimed to provide the structural information for the web data.

The **third-generation search engines** are characterised by a semantic-driven approach. The main advantage is that “it knows what it is trying to find”. Or in other words, it is a goal-driven search. It has the ability to learn from search behaviours of the user. During the search a user profile may be used or updated. Comparing to the second-generation search engine, the third-generation search engine is the content-based search plus the interpretation. It is an **incremental search** on all previously searched results. By using the context-sensitive analysis, the search pattern is recognised from its context. For example, ontologically a word “menu” in a software user guide must be different from that in a restaurant blurb. One of the advantages of the third-generation search engines is that it has greater search power than traditional database-oriented search engines. In addition to the ability to query structured data, it can also *filter* or *digest* unstructured data.

There is no means of that the third-generation search engines should replace the first two generations of search engines. Actually, they will all co-exist for their own applications. In implementation, the intelligent mobile agents can play a role in third-generation search engines.

In the Internet Commerce, the availability of business information is crucial. According to the types of business (i.e., business-to-business or business-to-customer), the information flow can be categorised based on the business workflow. Either about product advertisement or about the current status of service delivery, business transactions may generate infinite number of pieces of information and need to be organised in a way suitable for search agents.

Reflective Reasoning

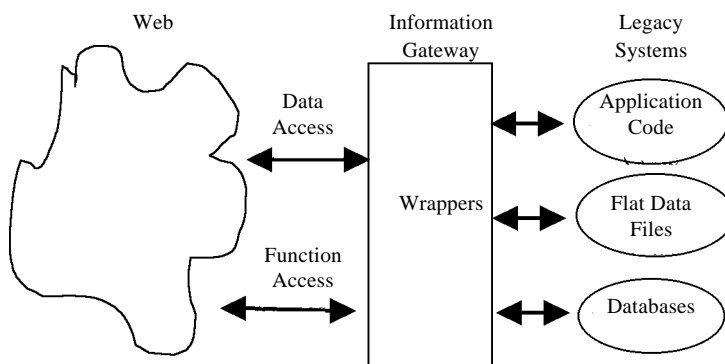
Agent server on the Internet provides an environment for the intelligent agents. Intelligent agents work with the server in an *open system* environment. Thereby the inference engine in the server should be able to deal with the dynamics in the knowledge management and manipulation. The server should be reflective to the changes of the domain knowledge. This requires a *causal connection* between its modeling mechanism and domain knowledge. The causal connection works like a mirror: it reflects changes in real world into an image-

Figure 4. The Internet Information Search Space

		Searched	Non-searched
Wanted		Expected	Minimized
Unwanted		Minimized	Expected

recording system. If the mirror is not functioning well, the recorded image can be distorted. The causal connection modeling is a meta level activity that server can incorporate changes into the knowledge base. The knowledge representation approach so need to

Figure 5. Web-Enabled Legacy Systems.



be reflective: the representation on both domain and meta knowledge.

To allow an inference engine capable of reflective reasoning, it requires the reasoning of *meta knowledge* triggered by exceptions in a reasoning process. It is context sensitive and retrospective in order to provide agents the answers that are normally not derivable from a first-order inference engine. The reflective reasoning is treated as an interrupt in the first-order reasoning process and will derive the goals that can handle abnormally situations of normal reasoning process. A kernel high-order reasoning mechanism will need to be built in an agent server. Although the research work on reflective reasoning has been started early (Maes, 1988), its reflection to the Internet intelligent agents is yet to be seen.

Counter Inference

Agents are often dealing with conflict interests. When an agent is working in a closed system environment, the outcome is predictable in a sense that the inference engine is a search machine working on the knowledge base plus the calculation of confidence factors. However in an open system environment, the goals are dynamic (i.e. changing and conflict), the knowledge bases are inconsistent, the reasoning strategies vary. The interaction between agents may yield win-lose situations (e.g., game playing) or win-win situations (e.g., negotiation). The kernel inference engine in an agent server should take the counter goals from different agents and progresses with the goal-refinement and reflective control on the inference process. The counter inference engine in this sense should be objective, and meta knowledge used in the control should be domain-knowledge independent.

In practice, an agent server should integrate the inference engine with both of the reflective reasoning and the counter inference.

IC DEVELOPMENT METHODOLOGY

The IC development methodology is a procedural work that can be followed step-by-step to create the IC systems. In this section, we are not going to give the detailed procedures but to discuss some issues that can affect the application of proper procedures. Firstly we discuss the modelling methodologies that provide the formal method for catching semantics of IC systems. Then we discuss the methodologies that are used to make *legacy systems* web accessible. Finally, we give an overview on the system development methodologies that are applicable to different engineering environments.

Modelling of IC Systems

The modeling of an IC system is to create a semantic model for the representation of the business in a trading environment. It is a conceptualization process. In this process, we need to use a syntactical tool to represent the concepts in the business environment. This tool should be capable of modeling the business in two aspects: the structural and the behavioral. In the structural representation, the business is recognized as entities, properties of the entities, and the relationships amongst the entities. Furthermore, the domain (static) constraints should also be identified to qualify the entities and their relationships. In the behavioral representation, the business is analyzed according to its workflow. So the business transactions are identified in terms of the functions performed by the entities. Moreover, the dynamic constraints are specified to control the transitions of the data updates. For example, a constraint may be specified to prevent the update on the employee data under the assumption that “an employee’s salary should never be decreased”. To sum up, a business semantic model is a representation of the business-trading environment in terms of its structural and behavioral aspects, plus the constraints.

The modeling process has two general questions to be answered: the **semantic completeness** and the **representation uniqueness**. By the semantic completeness, two aspects are considered: (1) everything in the business-trading environment will be described; (2) anything expressed within the resulting model is true in the business-trading environment. By this two-way checking we can be sure that the business semantic model derived from the modeling process is a true representation.

By the representation uniqueness, two aspects are concerned: (1) the *semantic non-ambiguity* is maintained; (2) the *minimum representation* is achieved. The semantic non-ambiguity requires that the typing system enforce both the unique naming convention and the elementary domain values. Therefore complex values can be constructed from elementary values without semantic ambiguity. The minimum representation requires that all derivable artifacts cannot be directly stored but represented as formulas or the derivation rules, so that there is no redundancy in the system representation. The minimum representation will guarantee that the smallest number of syntactic units is used to produce the model. So this ensures a way of standardization for the modeling process. By checking on the representation uniqueness we can be sure that the business semantic model derived from the modeling process is a *good* representation.

There have been many modeling methodologies proposed in recent years. The current consensus is the UML (Unified Modeling Language) promoted by the OMG (www.omg.org). The UML can be used to visualize, specify, construct, and document artifacts of the systems ranging from enterprise information systems to distributed web-based applications (Booch, 1999).

From a Legacy System to a Web-Enabled System

Many methodologies available currently are:

- the traditional software design and implementation methodologies which have little support for the system evolution towards the Internet information systems, or
- the software tool-based design and implementation methodologies that are applicable only on that tool. This leaves little room for a software engineer to minimise his/her effort in transforming a legacy system into a web-enabled system.

The legacy systems are defined and viewed in many different ways (Alderson, 1999 and Umar, 1997). In our discussion we regard the legacy systems as the systems that are disadvantaged by not having their data and functions web accessible. So, the question now

is to consider how we can convert the legacy systems into the web-enabled systems. There are two fundamental issues: (1) How do we make legacy data web accessible? (2) How do we make legacy systems run in the web environment? The former is a **data conversion** problem; the latter is a **system integration** problem.

For the both problems the *middleware programming* can provide the solutions. In the middleware programming, the legacy systems are *wrapped* with web accessible interface-protocol specifications, called *wrappers*. One example of the interface-protocol specifications is the IDL (Interface Definition Language) in the CORBA architecture (www.odmg.org). The IDL-specified CORBA wrappers are web accessible (Umar, 1997). The wrappers form a software layer that allows the legacy systems to exchange data or to interact with the web. We call this layer as the information gateway that solves the data conversion and the system integration problems for the legacy systems. Figure 5 illustrates this idea.

Methodologies in System Development

Many different approaches to IC system development currently exist. These approaches are applicable to different situations in the system development. We identify four different system development methodologies according to the focus of the developers: user-driven, data-driven, process-driven, and system-driven.

The user-driven methodology is to let the user to play the important role in the system development. It is an iterative process that is to build a prototype first and then the user will be working with the developer to improve the system functions. This approach is usually focused on small applications and the system can be quickly built and refined.

The data-driven methodology is used as a formal top-down refinement methodology. Following down a few successive stages, the system is developed. This approach starts with the formal specifications of the system requirement. Then the data model is derived. Based on the data model, the system transactions are defined and implemented. This methodology provides a chance to perform the verification for checking the system specification against the requirement. The data-driven methodology follows a formal software-engineering path and is good for the large systems that have clearly specified system requirements.

The process-driven methodology is used for the systems that consider the understanding of the system functions is more important than that of the data structure. In this kind of systems the fundamental processes are identified before the specification of a data model. This methodology is suitable for the systems that have complicated functions but have relatively simple data structure.

The system-driven methodology is applied on the existing system to reverse engineer it to a new one. By the reverse engineering, the new system interfaces and added functionality will be integrated with the old system. This will minimise the effort of creating a new system from the beginning. The new requirements can be compared with the old requirements to improve the system quality.

In the IC Engineering, all these four methodologies may be applicable. For example, a prototyping approach may be a good way to quickly develop a system for the demonstration of ideals. But it may be difficult for the further development regarding the change of the underlying system architecture. A reverse engineering approach may be helpful on re-developing a new system similar to an old one. But it is difficult to formally verify the new system because the reverse engineering may not be able to guarantee that the reversed system will be a true recover of the old one. Sometimes, the reverse engineering may have to “reinvent the wheel”, in which the problems were overcome before.

IC IMPLEMENTATION

In this section we consider how we can implement an IC system properly. The implementation is an issue of the **applicability** of the Internet technology. The availability does not equal to the applicability. The issue of the implementation is to understand the needs of the system and select the most suitable technology to implement the system. Instead of giving detailed comparing and contrasting different technologies, we discuss the background of the implementation. Firstly, we discuss the independent relationships existing in the implementation environment. The understanding of this may help us on selecting tools and back-end systems for the implementation. Secondly, we discuss the component technology that is used to construct a system. Finally, we give a brief list of the critical factors in IC Engineering.

Implementation Independence

When the system development comes to an implementation phase, it faces an intricate task: the decisions on building tools, on adherence of protocol standards, and on system construction. The complexity consists in the diversity of the current available technological products and the proprietary protocols that are all tangled. In this subsection, we identify the independent relationships that may help us to consolidate the decisions in an effort to unify the implementation process. The following independent relationships are identified:

- Front-end and Back-end system independence,
- Platform independence, and
- Proprietary protocol independence.

The front-end is independent from the back-end so that the implementation work can have different focuses as mentioned in the previous discussions of web architecture. However, this implies that we have to design a standard interface for the data flow between front-end and back-end systems. For example, XML can be used to specify Corporate Portals (Finkelstein, 2000) as front-end and access to the back-end of standard relational or standard (ODMG) object-oriented databases. Another example is to use some scripting programs embedded in an HTML document (e.g., CGI, Java applets, or ActiveX components) in the front-end system and use CORBA (www.odmg.org) or ADO (Gutierrez, 2000) for the back-end support for accessing databases. It may be worthwhile to mention that some front-end supporting functions may have minor compatibility problems such as that between the Netscape and the IE (Internet Explorer). In this case, the front-end development should limit the functions to use only those compatible features.

Platform independence is implemented by choosing a development tool that is supported by all operating systems. It is not difficulty for implementing the platform independence as long as we use OMG/ODMG Standard and Java, XML, tools. Many *off-shelf* tools are now providing the developers a freedom to implement systems on different platforms.

Proprietary protocol independence is an effort to improve the interconnectivity or the interoperability of the system for the potential incremental development. Many software companies provide the tools that are not designed for an open system environment. Although they may provide some API (Application Program Interfaces) for standard protocols, the integration of these tools may require an extra effort on leaning and programming of the API, and an extra layer of the system software for the bugs. The *middleware* programmers are now still in high demand. It is expected that with the development of the convergence of the universal protocols on the Internet, the proprietary protocol should have less interest for its existence.

Component Technology

The component technology (Szyperki, 1998) is developed based on the assumption of the open system environment. This technology is characterised by the reusability, portability, and the interoperability within an open system environment. A component is an independent and portable program that has its functions defined within the component interface. A component interface consists of *interface elements* that are the ontological descriptive notions of the interface including the syntax, semantics, and the functions of the component. A component has ability to response to the events that trigger the execution of the component or the interaction with other components. Components can co-operate together in a pre-designed architecture to perform the tasks of data processing or the process control.

The component-based systems are constructed based on a framework of the *component packaging*. When a system is designed, the system functions are refined from top-down to generate framework of components. The component packaging is the grouping of the functions to composite a system. On the other hand, the *system assembly* is considered to respond on the events that may happen to the system. The system assembly is regarded as a process that organises the components in terms of the event control.

In building the IC systems, the component-based system development may start with the transformation from a business model into a component-based architecture. Then the system is assembled based on the transactions in the business trading. Currently the available tools are the EJB (Enterprise JavaBeans) from Sun Microsystems® and the ETS (Enterprise Transaction Server) from Microsoft®. Both of them are used for the component-based and transaction-oriented applications. In EJB, transactions are packaged as the JavaBean objects. In ETS, transactions are packaged as the COM (Component Object Model) objects.

The future however, is still considering using the component technology in a higher lever of the component packaging that is to group the components in terms of the business types instead of the transactions. Furthermore the packaged components should be web addressable so a business (e.g., a virtual enterprise) can be established based on the systems that are already functioning. We may call this business-oriented component technology as Web Addressable Business Objects (WABO). Since the WABO is web addressable, it inherits all properties from the component technology plus that it is interconnected and interoperable. In designing a WABO-based system, we need to concentrate on the business requirements and the business functions. After that, the WABO-based system should be generated automatically for the implementation of the business model.

Critical Factors for the IC System Development

To bring a project to a success, there are three important aspects and the expertise: the **business management**, the **project management**, and the **technology management**. For simplicity of our discussion we just itemize the important points as follows.

Business Management

- From a business point of view to ensure the success of the idea.
- Keeping the business idea alive.
- Manage the financial support.
- Direct and steering the process and the new ideas of the project.
- Making contact to the business partners.
- Supervise and monitor the project progress.

Project Management

- From a management point of view to ensure the success of the project.
- Plan and schedule the project.
- Discover the problems and predict the possible impasse of the project development.
- Provide the information and the assessment for the business strategies.
- Organise the execution of the project (provide resources and the personnel).
- House keeping (documentation, financing, marketing, etc).

Technology Management

- From a technology point of view to ensure the success of the project.
- Make sure the best technology and the effective and efficient methodologies are applied to the implementation of the project.
- Provide the expertise and know-how to solve the problems in the process of implementation.
- Carry out the implementation of the project.
- Help the project manager for the planning and the scheduling.

The critical factors of the success in IC Engineering are that:

- The constant communication must be maintained to ensure the consensus of the project.
- Timing is important — a sense of entrepreneur.
- Be informative — monitor the current web new ideas and new development of web technology.
- Necessary resources must be guaranteed.
- Provide a simple yet functional and attractive model for the illustration of the idea.

CONCLUSIONS

The ultimate goal of the IC Engineering is to produce robust, competent, and viable web-enabled enterprise information systems. In this chapter, we have tried to reason out some important factors in building such systems. A scenario of IC Engineering may start with the creating of a business model in the process of business analysis and design. Then transform it into a system model that has its main interests in the system architecture and components. Finally the system implementation will be carried out. Within this scenario, three aspects are considered: requirement specification, Internet technology, and development methodology.

To achieve the success in IC Engineering, this chapter has given an overview on some current important issues. It aimed to provide a better understanding on the questions such as:

- What is the IC Engineering about?
- What is an IC system?
- What are the general requirements for an IC system?
- How do we benchmark an IC system?
- How do we develop such a system?
- How do we implement such a system?
- How do we make an IC Engineering project successful?

We consider IC Engineering as a progressive and reflective process. Within this process, we should not only understand what we are doing but also manage what we are doing. This chapter has given some rationale on the issues of this process. Many IC tools

provided by major software competitors are still weak in interoperability and interconnectivity mainly because of their proprietary protocols. It is expected that in a near future we may be able to build IC systems faster and easier by assembling systems with the plug-and-play WABO.

REFERENCES

- Alderson A. & Shah H., (1999). "Viewpoints on Legacy Systems", 42(3), March, Communications of ACM, 115-116.
- Booch G., Rumbaugh J., & Jacobson I., (1999). "The Unified Modelling Language User Guide", Addison Wesley.
- Chen M.S., Park J.S., & Yu P.S., (1998). "Efficient Data Mining for Path Traversal Patterns", IEEE Transactions on Knowledge and Data Engineering, 10(2), 209-221.
- Davis A.M., (1993). "Software Requirements Objects, Functions, and States", Prentice Hall.
- Dickman A., (1995). "Two-Tier versus Three-Tier Applications", Information Week, Nov, 13, 74-80.
- Finkelstein C. & Aiken P., (2000). "Building Corporate Portals with XML", McGraw-Hill.
- Glushko R.J., Tenenbaum J.M. & Meltzer B., (1999). "An XML Framework for Agent-Based E-Commerce", 42(3), March, Communications of ACM, 106-114.
- Gutierrez D.D., (2000). "Web Database Development for Windows Platforms", Prentice Hall.
- Kotonya G. & Sommerville I., (1998). "Requirements Engineering - Processes and Techniques", Wiley.
- Laplante M., (1998). "Making EDI Accessible with XML", E.COM Magazine, 4(2), March 1998, 23-26.
- Li X. & Low G., (2000). "Fuzzy Logic in Web Data Mining", Session on Fuzzy Systems, Proc of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI' 2000), Orlando, USA, July, 2000.
- Lin X., (1999). "Efficiently Computing Frequent Tree-Like Topology Patterns in a Web Environment", Proceedings of the 31 IEEE International Conference on Technology of Object-Oriented Languages and Systems (TOOLS 31), 440-447.
- Maes P., (1988). "Computational Reflection", Knowledge Engineering Review, 3(1), 1-19.
- Mobasher B., Jain N., Han E.H., & Srivastava, (1997). "Web Mining, Pattern Discovery from World Wide Web Transactions", Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97).
- Morrison M. & Morrison J. (2000). "Database-Driven Web Sites", Web Warrior Series, Thomson Learning.
- Szyperski C., (1998), "Component Software - Beyond Object-Oriented Programming", Addison-Wesley / ACM Press.
- Wooldridge & N.R. Jennings, (1995). "Intelligent Agents: Theory and Practice," The Knowledge Engineering Review 19(2) 115-152.
- Umar A., (1997). "Object Oriented Client/Server Internet Environments", Prentice Hall.
- Umar A., (1997b). Chapter 6: Web-Based Application Software Architectures, "Application (Re)Engineering - Building Web-Based Applications and Dealing with Legacies", Prentice Hall.