Overview

Unit 10 is organized in six sections, the first being this overview, the introduction and the objectives. The second section examines how a logical model can be used to derive the possible components of the systems architecture (network, database, interface and process). The third section deals with a strategic approach to application architecture selection and design implications. The fourth section relates to the steps involved in construction of a **physical data flow diagram (DFD)**. The fifth section covers the evaluation of the feasibility of different options and the presentation of this information in an updated feasibility report. This document compares and assesses detailed options giving management a firm basis to decide which option to adopt. The last section includes a further installment of the case study on the Orient-Pacific Insurance Corporation. This final section also includes a unit summary, list of references, glossary of terms and solutions for the self-tests and activities.
Introduction

In Unit 9, you learned that general design provides a blueprint (a road map, as it were) for subsequent detailed design and implementation, topics which will be covered in Units 11 to 14. Such a blueprint, referred to as an application architecture, is used as a general framework to define the technologies that will be used in the system. In other words, the application architecture defines the technology to be used to develop an information system, in terms of its data, process, interface and network components. You learned in Unit 1 that different stakeholders have varying views of the system’s architecture in terms of its different focuses — data, processes, interfaces and geography. Therefore, the objective at this stage of development is to make the technical decisions regarding various components of the system as they relate to data, processes, interfaces and geography. Such decisions include issues related to type of:

- computing (centralized versus distributed)
- databases (centralized versus distributed)
- input and output methods
- user interface
- software acquisition (make/buy).

You should note that in defining application architecture you need to deal with a variety of technical options about competing technologies, databases and networks.
Objectives

By the end of Unit 10, you should be able to:

1. Define application architecture in terms of data, processes, interfaces and networks.
2. Describe centralized and distributed computing alternatives, including client/server options.
3. Describe database and data distribution alternatives, and their importance in information systems (IS) design.
4. Describe user and system interface alternatives, and their importance in IS design.
5. Describe various software and development environments (process architecture) for IS design.
6. Describe strategies for determining IS architecture, including enterprise application and tactical strategies.
7. Describe the role of modelling in determining IS architecture.
8. Differentiate between logical and physical data flow diagrams.
9. Draw physical data flow diagrams for an IS architecture and process.
10. Compare competing systems architectures.
Information technology architecture

The primary force behind application architecture is the network architecture because the components of the database, process and interface architectures are all distributed across networks. For this reason, you should make sure that you have a solid understanding of the issues surrounding the design and management of network architecture.

Reading

‘Application architecture’, p. 423 in your text. The figure on p. 420 provides a detailed description of the components of an application architecture. This reading relates to Objective 1 of this unit.

Self-test 10.1

What role does an application architecture play in the development of an information system?

Network architectures for client/server computing

The latest generation of networked and distributed computing architectures is based on the client/server model. In this model, a multiple user computer called a server has the roles of storing a database, providing data processing and serving as a network interface to other systems (other servers and clients). Clients are typically single user stations (PC’s or workstations) that access the server for data or data processing services.

The benefit of this architecture has been due to a couple of major reasons:

• reduced processing cycles required of centralized computers

Data is ‘served’ to the ‘clients’ which are inexpensive local PC’s/workstations that have sufficient processing power. Savings come in the form of not having to continually purchase more powerful centralized mainframe computers.
• convenience of graphical user interfaces provided by PCs/workstations

These convenient graphical user interfaces are not available from traditional log-in terminal sessions found in centralized systems; and, thus, improve the ease of use for the end-user.

Client/server architectures come in two general categories of design: two-tier and three-tier (or n-tier). The two-tier architecture represents a simpler design approach where the first tier represents the client side with application logic and presentation (user-interface) logic. The second tier is the server side with the database back-end. An inherent problem with this design is that it is not scalable. That is, as the number of users goes up, the server has to deal with more and more connections of clients, which ultimately slows down the server's performance, something which in turn is cascaded to all the clients.

A solution is to use the three-tier architecture which partitions an application into three components:

• user-interface or presentation tier
• application logic tier
• backend database as the third tier.

The benefit of this design is that users of the system are likely to run the same logic on each client. That is, if there is a module that calculates statistical summaries of stock data, likely there are more than a handful of users needing to run the same module for the same data. Thus, by partitioning the application logic into a second tier and making it available only to those applications that need it, the performance and scalability improves for both the client and the server. It should be clear that the choice of a particular architecture determines the partitioning of the user interface, logic/processing and data.

In recent years, the choice and strategies of distribution have been improved with the popularity of local area networks (LANs), wide are networks (WANs), the Internet (or internetworking) and intranet (or intranetworking).

• LANs: Local area networks interconnect computers over a short distance (e.g., within 4 km) typically to a workgroup, building, campus, or so on.

• WANs: Wide area networks interconnect computers over a wider geographical distance (e.g., greater than 4 km; country-wide and international).

• Internet: A global and public network that originated in academic computing supported by the US government services (military). Currently, it represents the most important network infrastructure for corporations. Additional design details are discussed in Unit 13.

• Intranet: A secure, usually corporate, network that uses the Internet technology to integrate workstations, workgroups and enterprise computing into a single cohesive system.
Reading

‘Information technology architecture’, pp. 430–52 in your text. This reading reviews the client/server computing concepts in greater depth and provides example diagrams of such architectures. It relates to Objective 2 of this unit.

Self-test 10.2

1 Why has the client/server computing model been popular?

2 What is the difference between LANs, WANs, the Internet and intranet?

3 Differentiate between:

   (a) centralized and distributed computing

   (b) two-tiered and three-tiered client/server computing.

Data and database architectures

Physical data distribution is another consideration in the overall application systems architecture. Since the network provides the ‘roads’ that connect the information resources (the applications, interfaces and data), the next step is to consider how to physically locate the data resources. It may be easier to manage in a centralized database, but at the cost of performance and the risk that the organization is solely dependent on a single computer for its operations. In a distributed data design using client/server technology, data can be physically located at where it is created and most heavily used and yet accessible throughout the organization. In general, there are two strategies for data distribution:

- distributed database management system
  
  Since problems can arise with ‘synchronizing’ all the related data that is physically distributed, one can choose a distributed database management system (DDBMS) to take care of such overheads.

- data replication
  
  Another strategy to achieve higher performance at distributed locations is through data replication, where entire databases are duplicated and located in multiple locations. Mechanisms must be in place to ensure the data is consistent throughout all sites.

It should be noted that a data/database architecture should show the appropriate physical partitioning of data.
‘Data architectures — distributed relational databases’, pp. 443–45 in your text. This reading provides an overview of distributed relational database concepts, which relates to Objective 3 of this unit.

Self-test 10.3

1. What is the difference between a distributed relational database and a PC database?

2. What are the costs/benefits for the replication of data?

Interface architecture

Another important architecture is the interface which represents the input/output and middleware connections. Here are some of the design issues and solution technologies available to address the interface architecture:

- batch input/output
  This method is based on the accumulation of work (transactions) that will be processed together at defined times. This represents the old method of processing information. It still has its use and value today, but its popularity is on the decline.

- online processing
  This method is based on processing input/output as it occurs. Thus, there is no need for accumulated work as in batch operations. Its popularity has been due to end-user requirements for immediate responses from the system, such as customer queries to be answered immediately rather than queued and batched for later response.

- remote batch
  Similar to batch processing, it refers to the preparation and submission of a set of transactions to a remote system for processing and the immediate re-transmission of the results back to the user. It is a combination of batch and online processing, that is well suited to today’s mobile computing environments.

- keyless data entry
  Input of data into systems has been greatly simplified and can be attained at a higher level of accuracy thanks to new technologies in optical coding and scanning. Barcode scanning (of groceries, for example) is a de facto standard today, with the technologies extended to other
areas such as component monitoring on an assembly line. Optical character recognition for input of data (addresses, serial numbers, etc.) is another application of this input technology.

- **pen input**

Pen computing has not yet achieved a high-level of acceptance in mainstream IS applications, but has had its successes in the package delivery services industry. It has become an essential form of user interface for systems requiring Chinese character input and similar character-based language support.

- **voice input**

Recent progresses in the technology of voice input has made dictation and translation to computer more viable than ever before. Adoption of such new technology will need to evaluate the performance requirement, support for general users and extent of vocabulary support.

- **graphical user interfaces**

Ubiquitous today in PCs, the graphical user interface (GUI) has made computing more colourful and convenient. Using a desktop metaphor, icons and a pointing device, we can manipulate information presented in multiple windows of workspace. High-resolution graphics has made the concept of ‘what-you-see-is-what-you-get’ (WYSIWYG) a compelling interface for most word processing and document-based processing.

- **electronic message and work group technologies**

E-mail communication and topic-based newsgroups represent new trends in communications and information sharing called groupware. The concept is based on the notion that workgroups are able to share information conveniently through the convenience of delayed communications and multiple distribution of e-mail. Topic-based newsgroups are used as open discussion forums where problems and possible solutions can be discussed collectively.

- **electronic data interchange (EDI)**

**Electronic data interchange (EDI)** represents the inter-corporate electronic flow of documentation associated with trade transactions. This paperless technology aims to streamline the high cost of transacting goods and services. Currently, many global retail operations demand that all suppliers transact through EDI so as to gain a high level of efficiency previously bounded by paper-based activity. In Hong Kong, Tradelink, which is a consortium of some 20 major companies and the government, is in charge of the design and promotion of EDI.

- **imaging and document exchange**

File cabinet space has become a premium in many businesses. Hong Kong businesses are no exception; indeed, they probably suffer more than others from the acute costs of office space. Therefore, they can benefit significantly from the use of document imaging and exchange. Benefits such as faster access and document sharing compared to paper-based operations make this technology a major convenience for many industries.
Middleware is the utility software that allows for communication and data sharing among incompatible systems. Systems compatibility is a major issue plaguing many established organizations. From historical legacy systems to unique platform solutions for business, the need to link systems together for information sharing is a continued activity. Fortunately, solutions in the form of middleware have been forthcoming to addressing this issue.

In summary, the interface architecture provides many options and solutions for an information system. Understanding the different choices, costs and benefits helps the systems designer choose an appropriate solution mix.

**Reading**

‘Interface architectures — inputs, outputs and middleware’, pp. 445–49 in your text. This reading discusses in greater detail and gives examples of the different options available for an interface architecture. It relates to Objective 4 of this unit.

**Activity 10.1**

Search the Internet for companies providing one of the above technologies or services. One example is EDI Asia at the following URL:

http://www.EDIasia.com

Review the products and understand their capabilities.

**Process architecture**

Process architectures are associated with the software development environment (SDE) of programming language and tools used to define the business logic and applications. Most modern software/applications development environments today target a variety of platforms. They provide sophisticated tool sets for the software developer. Beyond the traditional code compiler (COBOL, C/C++, Visual Basic, etc.), many now integrate the following components as part of the development environment:

- integrated editors

These editors are built into the ‘edit, compile, run and debug’ cycle common to code development. Rather than execute a single program for each step, the editing environment becomes the interface to the environment. For example, lines of code are highlighted in the editor to indicate errors after running the compiler.
• GUI builders

In today’s graphical user environments of windows-based operating systems, a GUI screen designer is now a standard feature. Screens are drawn and controls such as buttons, text and input fields are drawn. Codes for each control will later be attached to make it functional.

• debuggers

Fortunately, programmers today can now be armed with advanced debugging tools that allow them to locate and resolve what was once difficult to find bugs. The ability to trace variables and lines of code through a debugger has improved the overall efficiency of programmers.

• report generators

Applications development for business environments inextricably requires reports — both paper-based and online. Thus, built-in report generators make it convenient for the developer to design useful reports quickly.

• software project management

Development of software requires the management of many component modules that represent a working system. Built-in to these environments are capabilities to control different versions of software and support for multiple developers.

• middleware to databases

Databases typically find themselves at the core of many information systems. Thus, applications development environments are now including middleware modules that allow access into a variety of popular databases as a standard option.

• profilers

These are code analysers used for performance analysis. A profiler will monitor the modules in a program in execution and report on which modules consume the most computing time or input/output requests.

• Internet/intranet support

The emergence and popularity of the Internet and intranet have caused the extension of many applications development environments to provide for its support. These environments facilitate the transitional development of existing applications for Internet access.

Reading

‘Process architecture — the software development environment’, pp. 449–52 in your text. It provides additional context to the process architecture in terms of the client/server approach, which relates to Objective 5 of this unit.
Activity 10.2

Evaluate the application development environments from the following companies by referring to their Web sites and observing what capabilities they provide with respect to the above components:

- **Microsoft**
  - http://www.microsoft.com
- **Lotus**
  - http://www.lotus.com
- **Powersoft (Powerbuilder)**
  - http://www.powerbuilder.com
- **Oracle**
  - http://www.oracle.com

Use your browser to locate their product sites.

Self-test 10.4

How has the productivity of the system builder improved using SDEs?
Application architecture and design implications

The decision on process architecture incorporating an appropriate applications development environment will rest on a number of issues:

- the existing software environment
- the requirements of the new system
- the organization’s ability to handle the technology
- the organization’s ability to afford the development costs.

An organization with a mature information technology (IT) department is likely to establish some standards in the process architecture for the development of applications. This is typically defined as the enterprise application architecture strategy. Alternatively, an organizational strategy needs to address the process architecture for each applications development effort individually, based on a feasibility analysis. This is called the tactical application architecture strategy.

Enterprise application architecture strategy

This particular strategy leverages on the mature organization in terms of corporate standards. Furthermore, one can expect that established controls for future integration and developmental planning exist and are well defined.

Tactical application architecture strategy

In this particular strategy, greater freedom is available for evaluating, choosing and deploying new technologies. It represents a localized or federated development approach, where each applications development effort is individually assessed with a feasibility analysis of a wider latitude.

Build versus buy

In the context of systems analysis and design, the focus tends to be towards the complete development of an information system. Alternatively, the option for purchasing some or all of the components for an IS needs to be considered. This ‘build versus buy’ issue has some implications for the process architecture. In particular the following issues need to be considered:
• the level of fulfilment of the software package for business requirements.
  Essentially, we need to determine whether there are any unfulfilled requirements.

• the degree of customizability of software to the environment.
  How customizable is the software to the required needs? Packaged software typically provides some level of customization.

• the fit of the software package into the existing operating environment.
  Essentially, we need to determine whether the software integrates into the existing computing and business environments successfully.

• the ability to import existing data into the new packaged system
  What facilities are available and how much effort is involved?

• analysis of software interfaces for requirements.
  We need to determine the ease or difficulty in integrating a package into the existing system.

**Reading**

‘Application architecture strategies for systems design’, pp. 452–53 in your text. Additional details are provided along with feasibility issues. This reading relates to Objective 6 of this unit.

**Activity 10.3**

This section provides you with a technical view of application architecture. Discuss what sorts of criteria may be considered when measuring an application architecture’s performance at the strategic level in your business organization.
Modelling the application architecture

The following are the modelling tools that help to define the application architecture and the distribution of processes.

Physical data flow diagrams

The data flow diagrams (DFDs) presented so far relate to the logical model of processes that occur in a system of concern. DFDs have been adapted to model the technology architecture in the systems design phase. These physical DFDs serve as the technical blueprints from which a system architecture can be built.

In the adaptation of logical DFDs to physical DFDs, there are some differences that need to be considered. Here are a list of the revised construct definitions:

Physical process: A processor such as a computer, application or person who will take input and generate output. In the declaration of the process, the technology used (e.g., barcode scanning) needs to be specified.

Physical data flow: This is the planned implementation of an input/output to/from a physical process. This could represent actual movement of paper forms (e.g., membership application forms) into the physical process or query transactions into a database server process.

Physical external agents: As they only serve to identify the interface and boundaries of the system, external agents are the same in definition to the logical DFDs.

Physical data stores: This is the actual database/file or specific database table where data are stored or to be retrieved from.

The physical DFD is essentially expanded out from the logical DFD, but embellished with planned physical technology implementations. Once developed, the physical DFD can be used as the blueprint to implement the system.

Reading

‘Physical data flow diagrams’, pp. 424–30 in your text. Figure 11-1 provides an example physical DFD and additional material discusses notation conventions. This reading relates to Objective 7 of this unit.
How to construct physical data flow diagrams

In the development of the physical DFD, the steps the system designer needs to follow are:

1. **draw the network topology or layout of the systems**
   
   This is what is called the **network topology data flow diagram (DFD)**. Identify servers and their physical locations (the servers can be either physical processors or data stores). The flows should indicate the network implementation details, such as protocol and speed. Clients on the network DFD can be grouped into classes of users for simplicity and practicality. This network DFD can be directly used to implement a network.

2. **assign the data distribution and technology**
   
   Using the data model of the entity relationship (ER) diagram and the logical network model developed in Unit 7, the data need to be assigned to the physical servers in the network topology. The physical distribution of the data can involve a number of strategies:
   
   a) storing all data on a single server
   
   b) partitioning out some of the database tables to reside on different servers
   
   c) partitioning the database tables further so that subsets of them reside in particular servers
   
   d) replicating some of the data among different servers.

   These decisions need to be made according to issues ranging from performance requirements to ownership and manageability of the data.

3. **assign the logical processes to physical processors**
   
   The decision on whether it is a one-, two- or three-tier architecture will depend on how the processes will be distributed to the servers (processors). Clearly, in a one-tier environment, all processes are stored on the server. In a two-tier client/server architecture, the processes are stored at the client side. In a three-tier client/server architecture, we need to determine where to locate the processes on the application server or at the client. The distribution will probably be based on whether processes are likely to be shared by many clients, in which case it makes more sense to place it on the application server.

4. **factor out manual processes from computer-based processes**
   
   The purpose of this final step is to identify the person/machine boundary. This will serve the purpose of identifying entry points of the human user interfaces into the appropriate computer processes.
**Reading**

‘Modelling the application architecture of an information system’, pp. 453–61 in your text. The complex and detailed diagrams in Figures 11.11 to 11.15 provide perspective to the physical DFD modelling issues. This reading relates to Objective 9 of this unit.

**Self-test 10.5**

1. What information do you need to gather in order to construct a physical DFD?

**Activity 10.4**

Prepare a physical data flow diagram to show the walk-in registration for enrolling OUHK courses.
Evaluating and comparing systems architectures

Usually, you will have many competing architectural designs to consider for a system. For a typical systems development effort, you need to formulate two to three reasonable architectural designs. Then for each, you will do a feasibility study (refer back to Unit 8 for a review) to determine its strengths and weaknesses. Note that a technical feasibility study can only be undertaken when detailed systems options are available; that is, when you have created a viable systems architecture. You then update your original feasibility study with more realistic estimates based on a viable systems architecture. This section relates to Objective 10 of this unit.

Assessing systems architecture feasibility

A technical feasibility study aims to assess the impact of a system on an organization. It usually contains an updated assessment of economic (financial) rationale, technical quality, operational impact, schedule feasibility, human factors (the user interface) and risk analysis. All of these evaluation methods rely on an accurate estimation of the impacts of a new system. This is a very difficult task. A common approach is to consider previous computing developments and to attempt to estimate by analogy.

Requirements evaluation

As a developer, for each systems architecture under evaluation you need to determine the degree to which that architecture meets the requirements specification.

You may have changed system constraints or scope at this stage of development (for example, to take advantage of new information technology opportunities). You may have also uncovered additional quality requirements at this stage. You need to evaluate technical quality by considering each of the performance and quality criteria documented in the requirements specification.

Comparing competing architectures

Once you have evaluated each alternative, you make an objective comparison of the various options. That is, for each architectural option, you need to do a cost/benefit analysis, risk analysis and an organizational impact analysis. A weighted matrix is a good method for this task. It makes decision criteria explicit.
Update feasibility report

A revised feasibility report is now updated with more realistic estimates based on a viable system architecture. It lays out clearly the options that management has for developing a new information system. It describes each option, or systems architecture, evaluates each and compares them. Therefore you will need to update your executive summary in the requirements specification. Of course, the content is different, but its purposes are the same; that is:

- to explain to executive management the reasons for the report
- to highlight significant findings
- to highlight significant recommendations for further action.

Each systems architecture is introduced with a statement of the philosophy on which it is based (for example, a distributed system). It includes physical DFDs and a development strategy. Each evaluation covers feasibility factors such as cost/benefit analysis, risk analysis, financial evaluation and organizational impact analysis.

Activity 10.5

Match the term with its correct definition:

1. partitioning
2. intranet
3. Internet
4. connectivity
5. interoperability

A. Defines how computers are connected to ‘talk’ to one another

B. A secure network, usually corporate, that uses Internet technology to integrate desktop, work group and enterprise computing into a single cohesive framework.

C. The art of determining how to best distribute or duplicate application components (data, process and interfaces) across the network.

D. An information superhighway that permits computers of all types and sizes all over the world to exchange data and information using standard communication protocols.

E. An ideal state in which connected computer cooperate with one another in a manner that is transparent to their users.

Summary

This unit has reviewed the aspects of turning a logical design into a physical/technical applications architecture. This general design serves as the blueprint for the system builder, identifying issues in network design, data distribution and process distribution in the implementation.

Generating a design option is a technical task which requires a lot of information technology knowledge. Evaluating that design is still a technical task, but also requires organizational and financial knowledge. Comparing options is a management task. A decision is usually made for all kinds of reasons, some of which are not available to systems developers (and not in their sphere of interest). An information technology professional is responsible for presenting viable, well-costed and well-analysed options. Thus, a requirements specification is used to bridge the gap between analysis and design. It forms a major input to a technical feasibility study. You evaluate and compare several physical design options. You prepare an updated feasibility report, which management uses to support decision making.

What lies next is the conversion of the above general design into specific design details of the database design, human-computer interface design and Internet design; topics which will be covered in Units 11 to 13, respectively.
References

You should be able to meet the learning objectives of this course and successfully complete your assessment on the basis of your study units and your textbook. These references are not prescribed reading; they are provided to enable you to develop your knowledge beyond the requirements of this course.


Glossary

This glossary provides brief definitions of the main technical terms you encountered in Unit 10. The definitions given here are offered in addition to, or in place of, those found in your textbook.

**Application architecture:** Defines the technologies to be used by (and to build) one, more or all information systems in terms of its data, process, interface and network components. It serves as a framework for general design.

**Centralized computing:** When a multi-user computer (usually a mainframe or minicomputer) hosts all the information system components, including (1) the data storage (files and databases), (2) the business logic (software and programs), (3) the user interfaces (input and output) and (4) any system interfaces (networking to other computers and systems). The user may interact with this host computer via a terminal (or, today, a PC emulating a terminal), but all the work is done on the host computer.

**Client:** Single-user computer that provides (1) user interface services and appropriate database and processing services, and (2) connectivity services to servers (and possibly other clients).

**Client/server computing:** When an information system’s database, software and interfaces are distributed across a network of clients and servers that communicate and co-operate to achieve systems objectives. Despite the distribution of computing resources, each system is perceived by the user as a single computer (the user’s own client PC) performing all the work. Synonyms include distributed computing.

**Connectivity:** Defines how computers are connected to ‘talk’ to one another.

**Data distribution:** Partitions data to one or more database servers. Entire tables can be allocated to different servers, or subsets of rows in a table can be allocated to different servers. A distributed relational database management system controls access to and manages each server.

**Data replication:** Duplicates data on one or more database servers. Entire tables can be duplicated on different servers, or subsets of rows in a table can be duplicated to different servers. The distributed relational database management system not only controls access to and management of each server database, but it also ensures that updates on one server are updated on any server where the data are duplicated.

**Database engine:** That part of the database management system that executes database commands to create, read, update and delete records (rows) in the tables.

**Database servers:** Store the database, but the database commands are also executed on those servers. The clients merely send their database commands to the server. The server returns only the result of the database command processing – not entire databases or tables. Thus, database servers generate much less network traffic. This approach is used by high-end database software such as Oracle and Microsoft SQL Server.
**Distributed relational database:** Distributes or duplicates tables to multiple database servers (and in rare cases clients).

**Distributed relational database management system (RDBMS):** A software program that controls access to and maintenance of the stored data. It also provides for backup, recovery and security. It is sometimes called a client/server database management system.

**Electronic data interchange (EDI):** The electronic flow of business transactions between customers and suppliers.

**File servers:** Store the database, but the client computers must execute all database instructions. This means that entire database and tables may have to be transported to and from the client across the network. The approach, used by database software such as Microsoft Access and Borland dBASE, typically generates excessive network traffic.

**Internet:** A (but not necessarily the) information superhighway that permits computers of all types and sizes all over the world to exchange data and information using standard languages and protocols.

**Interoperability:** An ideal state in which connected computers cooperate with one another in a manner that is transparent to their users (the clients).

**Intranet:** A secure network, usually corporate, that uses Internet technology to integrate desktop, work group and enterprise computing into a single cohesive framework.

**Local area network (LAN):** A set of client computers (usually PCs) connected to one or more server computers (usually microprocessor-based, but could also include mainframes or minicomputers) through network cabling over relatively short distances; for instance, in a single department or in a single building.

**Middleware:** Utility software that interfaces systems built with incompatible technologies. Middleware serves as a consistent bridge between two or more technologies. It may be built into operating systems, but it is also frequently sold as a separate product.

**Network topology:** Describes how a network provides connectivity between the computers on that network.

**Network topology data flow diagram (DFD):** A physical date flow diagram that allocates processors (clients and servers) and devices (e.g., machines and robots) to a network and establishes (1) the connectivity between the clients and servers and (2) where users will interact with the processors (usually only the clients).

**Partitioning:** The act of determining how to best distribute or duplicate application components (data, process and interfaces) across the network.

**Physical data flow:** Represents the planned implementation of an input to or output from a physical process. It can also indicate database action such as create, delete, read, or update a record. It can also represent the import of data from or the export of data to another information system across a
network. Finally, it can represent the data flows between two modules or subroutines within the same program.

**Physical data flow diagrams (DFDs):** Model the technical and human design decision to be implemented as part of an information system. They communicate technical and other design constraints to those who will actually implement the system — in other words, they serve as a technical blueprint for the implementation.

**Physical data stores:** Represent a single file or a single database or table in the database. Additional physical data stores may be added to represent temporary files or batches necessitated by physical processes.

**Physical process:** Either (1) a processor, such as a client PC, network server, or robot, or (2) specific work or actions to be performed on incoming data flows to produce outgoing data flows. In the latter case, the physical process must clearly designate which person or what technology will be assigned to do the work.

**Relational databases:** Store data in a tabular form. Each file is implemented as a table. Each field is a column in the table. Each record in the file is a row in the table.

**Server:** A multiple-user computer that provides (1) shared database, processing and interface services and (2) connectivity to clients and other servers.

**Software development environment (SDE):** A language and tool kit for constructing information system applications. They are usually built around one or more programming languages such as COBOL, BASIC, C or C++.

**Systems flowcharts:** Diagrams that show the flow of control through a system while specifying all programs, inputs, outputs and file/database accesses and retrievals.

**Systems integration:** The process of making heterogeneous information systems (and computer systems) inter-operate. A key technology used to interface and integrate systems is middleware.

**Version control and configuration managers:** Software that tracks ongoing changes to software that is usually developed by teams of programmers. The software also allows management to roll back to a prior version of an application if the current version encounters unanticipated problems.

**Wide area network (WAN):** An interconnected set of LANs, or the connection of PCs over a longer distance — such as between buildings, cities, states or countries.
Answer key for self-tests and activities

Self-test 10.1

1 The application architecture is a necessary component of the physical or technical system design. Decisions made regarding the network topology, hardware, database distribution and process development environment will guide the system builders in the implementations.

Self-test 10.2

1 The client/server model has been popular due to the rise in low-cost desktop PCs and workstations. Instead of relying on more powerful mainframes for an organization’s computing needs, desktop PCs which serve as clients are able to perform the computational logic. The server machine (PC, workstation, mini- or mainframe) now becomes the data/database server for information. This along with the benefits of the graphical user-interface on PCs/workstations have made client/server computing popular.

2 A LAN is a local area network, which is a small private network usually for a workgroup, building or company complex within 2 km in distance. A WAN is a wide area network, which is a private network linking a geographically dispersed organization beyond the 2 km reference boundary. The Internet is a public (wide area) network servicing first the academic, then the commercial world and the general public for dissemination of all types of information and applications. Intranet is a secure corporate network that is developed using Internet technology.

3 (a) Centralized computing:

The use of a host computer to handle all processing including input, output, data storage and retrieval and business logic. Although users may interact with the host computer via a terminal or a PC emulation program, all the works are done on the host.

Distributed computing:

The use of multiple computers handle all activities of information processing. Each computer in the network handles its own input, output, data storage and retrieval and business logic.

(b) Two-tiered client-server computing:

(Similar to the concepts of distributed computing.) This architecture places the information system’s stored data on a server (database server), and the business logic and user interfaces on the clients.
Three-tiered client-server computing:

Uses the same database servers as in the two-tiered approach. Additionally, it introduces an application server to handle business logic — the transaction applications. Therefore, some or all the business logic of all of the application can be moved from the clients to the server. The user interfaces are still on the client side.

Self-test 10.3

1  A distributed relational database is a multi-user database that is viewed logically as one database but is geographically dispersed or partitioned at different sites. It requires a distributed database management system in order to manage and keep up to date the information at all sites. A PC database is typically a single user (though can be multi-user) and is located at one physical location.

2  The benefit of replication of data is the speed of access at each site or region for data that would otherwise be geographically dispersed over longer distances and/or in fragments. It provides a single point of access for most site in retrieving data. The cost is associated with the overhead and complexity in maintaining the correctness of replicated data in multiple sites.

Self-test 10.4

System builder productivity has significantly improved through the use of SDEs. Many SDEs integrate the edit, compile, run and debug cycle in the environment and offer a slew of features that greatly facilitate the robust development of applications. They are available for a number of platforms from PCs to large scale systems.

Self-test 10.5

In order to construct a physical DFD (PDFD), we need to gather the process models (logical DFDs), data models (ERDs), logical network models and location connectivity matrices. Additional information such as the people/machine boundary need to be provided as well. Chosen network hardware, SDEs and databases also need to be considered.

Activity 10.3

The answer will various. Here is a list of suggested criteria for consideration.

1  The ability of the application architecture to provide good-quality information relating to the internal and external business environment.

2  Provision of the appropriate information infrastructure to support the needs of decision makers and to complement the external information.
3 Capacity to support the business managers' needs in terms of analytical models, tools and techniques.

4 The ability to provide opportunities for the business organization to gain competitive advantage through the resulting information generated by the application architecture.

**Activity 10.4**

![Diagram](image)

**Activity 10.5**

1 C
2 B
3 D
4 A
5 E
Unit 10

Architecture and process design
Course Team (Original course)

Project manager: Steve Elliot, OUHK

Course developers: University of Technology, Sydney
Craig McDonald, Consultant
David Wilson, Consultant

Course designer: Shannon Timmers, OUHK

Course Team (Revised course)

Course developers: Ali F. Farhoomand, Consultant
Lester W. Yee, Consultant

Course designer: Linda Chung, OUHK

Course coordinator: John Mak, OUHK

Member: Nigel Leung, OUHK

External course assessor

Professor NV Balasubramanian, City University of Hong Kong

Production

ETPU Publishing Team

Copyright acknowledgements

The following copyright materials are reproduced under the terms of a licence issued by Hong Kong Reprographic Licensing Society. No further copying is permitted.

Information in Activity 10.5.

Acknowledgement

The OUHK gratefully acknowledges a grant of $100,000,000 from the Hong Kong Government designated for course development.

B329 Systems Analysis and Design was developed in part using resources provided by this generous grant.

Copyright © The Open University of Hong Kong, 1998, 1999.

All rights reserved.
No part of this material may be reproduced in any form by any means without permission in writing from the President, The Open University of Hong Kong.

The Open University of Hong Kong
30 Good Shepherd Street
Ho Man Tin, Kowloon
Hong Kong
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Objectives</td>
<td>3</td>
</tr>
<tr>
<td><strong>Information technology architecture</strong></td>
<td>4</td>
</tr>
<tr>
<td>Network architectures for client/server computing</td>
<td>4</td>
</tr>
<tr>
<td>Data and database architectures</td>
<td>6</td>
</tr>
<tr>
<td>Interface architecture</td>
<td>7</td>
</tr>
<tr>
<td>Process architecture</td>
<td>9</td>
</tr>
<tr>
<td><strong>Application architecture and design implications</strong></td>
<td>12</td>
</tr>
<tr>
<td>Enterprise application architecture strategy</td>
<td>12</td>
</tr>
<tr>
<td>Tactical application architecture strategy</td>
<td>12</td>
</tr>
<tr>
<td>Build versus buy</td>
<td>12</td>
</tr>
<tr>
<td><strong>Modelling the application architecture</strong></td>
<td>14</td>
</tr>
<tr>
<td>Physical data flow diagrams</td>
<td>14</td>
</tr>
<tr>
<td>How to construct physical data flow diagrams</td>
<td>15</td>
</tr>
<tr>
<td><strong>Evaluating and comparing systems architectures</strong></td>
<td>17</td>
</tr>
<tr>
<td>Assessing systems architecture feasibility</td>
<td>17</td>
</tr>
<tr>
<td>Requirements evaluation</td>
<td>17</td>
</tr>
<tr>
<td>Comparing competing architectures</td>
<td>17</td>
</tr>
<tr>
<td>Update feasibility report</td>
<td>18</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>19</td>
</tr>
<tr>
<td><strong>References</strong></td>
<td>20</td>
</tr>
<tr>
<td><strong>Glossary</strong></td>
<td>21</td>
</tr>
<tr>
<td><strong>Answer key for self-tests and activities</strong></td>
<td>24</td>
</tr>
</tbody>
</table>