Overview

The fifth unit of the course B329 Systems Analysis and Design is organized into eight sections, of which this overview and the introduction is the first.

The second section introduces the concepts of process modelling for a business information system.

The third section introduces the most widely used process modelling methodology of data flow diagrams (DFDs) and its constructs.

The fourth section outlines out the general design principles for developing good data flow diagrams.

The fifth section lists out the most common data flow diagram construction errors, along with examples.

The sixth section presents the detailed specifications that completes the data flow diagram models. The specification includes declaration of logic and data.

The seventh section summarizes the iterative process of data flow diagram creation.

The last section contains a summary of Unit 5, case study, references, a glossary of terms, and answers to the self-tests.
Introduction

After the determining the basic requirements analysis of what users need from their information system in Unit 4 it is now time to develop systems models. These systems models serve as a road map that is used by analysts, managers, programmers, and end-users to communicate the system itself. Since a single modelling methodology cannot fully communicate the various aspects of an information system, we will cover three distinct but related modelling methods that will be used together to fully specify the system. Specifically, this Unit 5 will first cover process modelling. Unit 6 will cover data modelling and Unit 7 will address network modelling. These three units together form a coherent modelling methodology for systems analysis.

Process modelling helps the analyst to specify what a system needs to do. The tools for modelling are thus focused on these actions (also called processes or functions) of a system. The set of tools that we will use to form an integrated process model are data flow diagrams (DFDs), process descriptions, and a data dictionary. Together these tools can completely specify the processing functions for an information system. Thus, this unit will go over the tools themselves and how they are to be used.
Objectives

By the end of Unit 5, you should be able to:

1 Define process modelling and explain its value.
2 Analyse a situation from a process perspective for constructing a process model.
3 Construct a data flow diagram.
4 Apply proper design principles in constructing data flow diagrams.
5 Identify common data flow diagram errors.
6 Record the details in data flow diagrams as process definitions and data structures.
7 Explain the use of structured English and decision tables in process specifications.
8 Read and construct data structures for defining data flows and data stores.
Concept of process modelling

In process modelling, we identify our information system for development first as a process. That is, for our system, there are some inputs which are transformed within and resultant outputs. This is essentially what a process is (Figure 5.1).

![Figure 5.1 Basic process model](image)

Process models are further distinguished into two types — logical models and physical models. These two types represent different views of the information. The definitions are as follows:

**Logical models** depict with some abstraction the functions of a system that is independent of any implementation. Typically a logical model will include manual processes as well as computer-based processes without distinctions.

**Physical models.** Physical models tend to be more technical in nature. They show how a system is physically and technically implemented. In specific processes, they may depict the actual flow of documents, whereas a logical model would just indicate flow of information only.

The emphasis of this unit will be on the logical models for a number of practical reasons. They are:

1. to avoid biasing a design with existing physical implementations or from the way someone thinks it should be implemented. It also naturally simplifies looking at systems in terms without all the details which can help to encourage creativity in designs.

2. analysis is better considered in terms of ‘what to do’ (logical) as opposed to ‘how it is done’ (physical).

3. Logical models are much easier in communicating to a wider group of stakeholders — from managers to end-users. A complex system is much easier to understand without all the physical details to consider.

**Reading**

‘An introduction to systems modelling’, pp. 307–310 in your text. This reading relates to Objective 1 of this unit. It also provides an additional background information on system process modelling.
Data Flow Diagrams (DFDs)

A Data Flow Diagram (DFD) is the primary technique for representing processing functions of an information system and showing how data used by those functions flows through the system. It is also the most popular and well-established process modelling methodology in use today. Using a simple four-construct (three symbols and a connection) diagramming method (Figure 5.2) we can fully construct a process model of an information system.

**Figure 5.2** Modelling constructs for DFD

**Modelling constructs**

Descriptions of the constructs used in data flow diagrams are explained below.

**External agents** or **external entities** are people/systems that are outside the boundary of the system but do interact with the processes in terms of providing input or receiving outputs from the system processes. Examples can include customer, bank, accounting system, and so on. External agents can only connect to processes.

A **process** or **process bubble** is a construct that represents a process within a system. It is usually identified as a round-cornered rectangle or simply a circle. Both diagram representations are commonly recognized and accepted. Choose one for consistency, your book uniformly uses the oval representation.

A **data flow** is simply lines with arrows that flow in or out of a process. It represents data in motion to and from a process.

A **data store** is essentially data at rest. It can be some temporary storage of output from a process or more permanently a data-file/database for input and output. Data stores can only connect to processes.
The systems approach

Since information systems can vary in size and extent, trying to model a large scale system with a single logical diagram can be nearly impossible. A better way to do it is to partition the effort into smaller logical chunks. This is also called the **systems approach** to modelling — breaking a large problem (system) down into more manageable components. For DFDs, this allows for ease of understanding in a large scale system which could potentially have tens, hundreds, or more individual processes; which would be nearly impossible to understand as a single diagram. In process modelling the term used is called **process decomposition** or **hierarchical partitioning**.

**Reading**

‘System concepts for process modelling’, pp. 310–331 in your text. This reading relates to Objective 2 of this unit. It also provides a deeper perspective into the systems approach including a number of diagrams.

Process decomposition

Process decomposition or hierarchical partitioning is a top-down approach whereby a complex information system is analysed and broken down into small parts (or processes). Each part may itself be broken into smaller parts and so on until an elementary level is reached where the part (or process) is absolutely simple. These lowest level processes are called **elementary processes** or **primitive processes**.

Balancing the depth, breadth, and complexity are strong rules of process decomposition. The idea here is that a DFD should not be cluttered with so many flows and bubbles that the sense of the diagram is lost. Remember that a DFD aims to communicate components of a system and how those components are related to a user — to have a dozen processes is just too much to comprehend. Also, all processes on a DFD should be more or less the same level of complexity. That is, we should not have one process expanded down 3 levels whereas the other processes expand down only 1 more level. Consider the possibility to ‘regroup’ the processes that may create only 2 levels of easy to read DFDs.

To help identify lower level DFDs, a standard numbering convention for processes are used. Consider that in partitioning a DFD, you will use a standardized numbering system to clearly indicate from which parent process a child DFD is exploded. A process P5 may be exploded into a child DFD containing processes P5.1, P5.2 and so on. This numbering is critical because without it there is no way of telling which is the parent of a given DFD! The standard levels are defined accordingly (see Table 5.1).
Table 5.1 Process identification notation

<table>
<thead>
<tr>
<th>Process level</th>
<th>Process numbering convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context diagram</td>
<td>none — only one process</td>
</tr>
<tr>
<td>Level 0</td>
<td>format no. (e.g. 1, 2, 3, ...)</td>
</tr>
<tr>
<td>Level 1</td>
<td>format no..no. (e.g. 1.1, 1.2, 2.5, ...)</td>
</tr>
<tr>
<td>Level 2</td>
<td>format no..no..no. (e.g. 1.1.1, 1.1.2, 3.1.5, ...)</td>
</tr>
<tr>
<td>Level n</td>
<td>format no..no... (process format with n dots)</td>
</tr>
</tbody>
</table>

With process decomposition, there is always an upper-most (parent) level diagram. This is called the context diagram. The context diagram is important in that it defines the boundary of the system under consideration. Only one process is defined, that of the information system. However, external agents are also defined to identify the sources and sinks of data flow to and from the system respectively. Additionally, data stores can be defined here to represent external databases that the system may retrieve from or store to, that is not under the control of the system itself. See Figure 5.3 for an example of a context diagram.

Figure 5.3 Context diagram of comic book rental company

A level 0 diagram is an exploded view of the context diagram showing the major component processes within (Figure 5.4). If you compare it with the context diagram, you can see the additional level of details. If I was to expand out process P2 into a level 1 diagram, you can see an additional layer of details (Figure 5.5).
Figure 5.4  Level 0 diagram of comic book rental company

Figure 5.5  Expanded level 1 diagram of comic book rental process
Self-test 5.1

1. What is the difference between logical and physical models in process modelling?

2. What is meant by the systems approach to process modelling?

3. Write a brief definition of (a) context diagram and (b) diagram 0.

4. For Figures 5.3–5.5, list out the external agents, processes, data flows, and data stores.

5. Decompose processes 1 and 3 in Figure 5.4 of the level 0 diagram. Make assumptions about the process as necessary.
Design principles in DFDs

Data flow diagrams are supposed to be easy to understand and sharable with the wide range of users and stakeholders of the information system being considered. Having said that, there are some very basic principles in using DFDs that should be followed when developing these process models. The following is a list of these principles.

- A general rule is that each DFD should have no more than ten processes maximum. An alternative rule would be to use the seven (+/-) two rule for number of processes in a data flow diagram. Thus the role of the systems analyst is to properly determine the logical component processes for a given (sub-)system view. We can then expect that individual analysts are likely to derive somewhat different models of the same system. Typically there is no one correct DFD model, but any assessment will be on the reasonableness of the model in representing the reality (system processes and requirements).

- The importance of careful naming of all components in a DFD cannot be over emphasized. It is a good indication that your thoughts about a component are fuzzy if you cannot think of a strong name — a strong verb and object for a process (for example, ‘Calculate net pay’) and a strong noun for a data flow or store (for example, ‘Invoice’). An additional reason why naming is so critical is that as you tackle increasingly large systems using several analysis tools you will find the name of an item appears over and over again as that item is identified from different points of view. Careful naming means that you will have consistent naming. Even in exercises and assignments during this course, you will have problems if you call an item one thing in a DFD and another in a process description.

- **Data flow conservation** states that flows should represent only relevant or usable data needed by and generated from the process. That is, data that is passed through a process without any manipulation should not be declared in a flow.

- Duplication of external entities or data stores may be necessary in a single DFD to avoid flows criss-crossing each other. A common way to assist a user in recognising that there are duplicates is to put a slash “/” or “\” mark on the lower corners of duplicated external entities or data stores. Note that your textbook does not use this notation, but many CASE tools do employ this notation method for convenience. Duplication in general should be minimized whenever possible.

- **Synchronization** is the balanced flows of upper and lower level DFDs. If an upper level process has two input flows and three output flows passing through its boundary, the lower level DFD should reflect this exactly. Note that some upper and lower level DFDs show common data stores — so this principle should be viewed in a logical sense. Depending on the CASE tool that you may be using, they may or may not check for such consistency.
Reading

‘Data flows’, pp. 321–329 in your text. This reading relates to Objectives 3 and 4 of this unit. Additional design principles are covered.
Common DFD modelling errors

Though developing DFD models is relatively straightforward with its four constructs, it is not without its pitfalls — particularly for less experienced systems analysts. The following are common modelling errors found in DFDs. Understanding these mistakes will help to avoid making them.

A **black hole** is a process where all flows connected with it are input flows. A process must have both inputs and outputs. Mistakes here are derived from misdirected flow arrows and forgotten outputs. See Figure 5.6.

A **miracle** is a process where all flows connected with it are output flows. Again a process must have both inputs and outputs. Mistakes here are derived from misdirected flow arrows and forgotten inputs. See Figure 5.6.

![Figure 5.6 Examples of black hole and miracle DFD errors](image)

A **gray hole** reflects a process in which the inputs are insufficient or unbalanced to produce the output. Sources of these problems include missing flows, mislabelled flows, and incorrect process label. See Figure 5.7.

![Figure 5.7 Example of a gray hole DFD error](image)

**Reading**

‘Logical processes and conventions’, pp. 323–324 in your text. This reading relates to Objective 5 of this unit.
Worked example
Consider the following DFD which contains several errors. Please identify and list the errors.

Answers
The identified errors are:

1. Process p4 is a black hole.
2. Process p3 is a miracle.
3. A data flow coming out of p2 is not labelled.
4. Data store s1 cannot be directly connect to data store s2 without a process.
5. Data flow f6 is used in two places. It is an input flow from p3 and a flow from s1 to s2.
Detail specifications for DFD

A DFD allows you to analyse requirements for data and information processes. However, it does not describe components in detail, it only names them. The next step is to declare the logic specifications that define each elementary process and the data structure (or data dictionary) entries for each data flow (or data store).

Process logic specification

In specifying process logic or alternately process descriptions, we typically use two popular ways to specify how data is transformed or manipulated. For loosely structured logic, we have structured English, and for highly structured logic such as decision rules, we have decision tables. Details on each are described below.

Structured English

Structured English is used as an easy to read specification of logic that a process may have. The syntax is similar to a structured programming language but replaces the vagueness of variables and functions with high-level English language expressions. If you are familiar with programming, you can think of it as a structured pseudocode that can be used for communicating logical details to a broader class of users (managers to end-users) as well as serving the programmers.

The structured English vocabulary comprises:

**Sequence.** This is a series of steps of action that typically starts with action verbs such as GET, EDIT, FIND, COPY, RECORD, CREATE, READ, UPDATE, DELETE, CALCULATE, WRITE, SORT, MERGE, and so forth. The notation for sequence of actions is `[action...]` which is used to describe the remaining structured English language.

**Conditions (or Selection).** Representing simple rules or decision logic, this can be expressed using constructs of IF [condition] THEN [actions...]. The conditions themselves will evaluate to True or False and can include the standard logical operators of AND, OR, NOT, >, <, =, and so on. Multiple branching conditions can also be expressed with the CASE construct (e.g. CASE 1: IF [condition] THEN [action], CASE 2: IF [condition] THEN [action], ...; where [conditions] tend to follow a pattern. Complex decisions can default to a decision table as well.

**Iterations.** In cases where you need to repeat a sequence of actions based on some condition, we have the following set of structured English constructs that by the way is not unlike those offered in many programming languages. Specifically:
1. REPEAT [actions...] UNTIL [condition]
2. DO WHILE [condition], [actions...]
3. FOR [indexing condition], [actions...]

**Decision table**

A decision table can also be used to specify the logic within a process. It is particularly well suited for describing complex decisions completely and concisely. So if a process is really a series of related rules (e.g. IF... THEN...), then this manner of presenting the logic is best for communicating the intricacies involved where mistakes can go unnoticed.

**Worked example**

Consider a process that determines the import tax for cars into Hong Kong.

Cars are taxed according to the following rates which are based on region of origin. Asian cars are taxed at 20%, European cars are taxed at 25% and North American cars are taxed at 15%. Additionally any car with an engine over 3 litres in size is taxed an additional 5% as a pollution tax. Additionally, any car that sells in access of HK$1,000,000 is assessed an additional 5% luxury tax. Draw the decision table for the above decision logic.

**Answer**

There are 3 decision variables (location, engine size, and cost), thus the table will have 4 x 2 x 2 = 16 combinations of results.

<table>
<thead>
<tr>
<th>CONDITION</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
<th>35%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>European</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>North American</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Over 3 litres eng.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Luxury $1M</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAX</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>25%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>30%</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>35%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
One common problem encountered in developing decision tables is in specifying the ranges of choices. It is critical that ranges do not overlap and that they cover a full spectrum of possibilities.

The techniques of structured English and decision tables are not exclusive of one another. You often use a mixture of them in a process logic specification. A common combination is the use of structured English to provide overall form and decision tables to specify complex decisions.

Data flow and data store descriptions

A data dictionary is a collection of data structures that specifies data stores and data flows that were identified in a DFD. Another way of looking at data flows is to imagine (may actually be reality!) it as a physical document moving to or from a process. You can easily imagine that an application forms for membership or a purchase order for goods will pass through many human hands (processes) before arriving at some destination. Data flows contain data (e.g. fill-in values of a form) that need to be declared precisely. As such, there is a data structure representation used for this. Data stores likewise can be described using the data structure notation as well. (Note: In Unit 6, we will look at data modelling that will take into greater consideration the data stores.)

Consider the following example of a data structure definition for a sales report by individual sales-person:

```
MONTH = {SALESPERSON-NAME +
         {CUSTOMER-NAME +
           AMOUNT-SOLD } +
         SALESPERSON-TOTAL }
```

An English interpretation of the meaning of the above data structure definition says that the monthly report contains a listing of sales people and their customers — grouped by the sales person’s name. Thus each customer of the salesperson along with the amount of the sales are also listed. Then a monthly total sales figure for each salesperson is presented.

Data structure constructs

Though the above example is quite simple, the data structure constructs show repetition “{ | }” for groups of attributes. Required sequences of attributes are connected by “+”. Thus the complete construct specification for data structures includes:

- “=” States that a data structure (name on left-hand-side of equal sign) is composed of the following attributes on the right-hand-side.
- “[...]” States repetition of attributes or sub-structures within.
- “[..]” Means that one of the attributes of a choice set is to be chosen.
- “(...)” States that attribute(s) within are optional.
As these data structures are named, they should be arranged alphabetically in the data dictionary for easy access and look-up. Notice that there is no information about how data is processed. That is the task of a process logic specification.

Reading

‘The process of logical process modelling’, pp. 331–335 in your text. This reading relates to Objectives 6 and 7 of this unit. Detailed guidelines on using structured English, decision tables, and data structures are presented.
Constructing a set of DFDs

It is usual to take a step-by-step approach to constructing a set of DFDs. This approach works well for complex systems.

Reading

‘How to construct process models’ pp. 336–351 in your text. This reading relates to Objective 8 of this unit. This section provides greater details of the process of constructing DFDs and its linkage to business event modelling.

However, you should be aware that one critical problem of this step-by-step approach is that it disguises the iterative nature of developing DFDs. Typically, you sketch a DFD and find problems in that formulation of a system. You modify it and find difficulties again. You change it again, and so on. What is happening is that, as you express your understanding of a situation under study, your perceptions change, becoming clearer and founded on more reliable information. A DFD is a working tool. It allows you to do analysis by formulating a problem, testing that formulation and reformulating it. This means that you can expect to revise your DFDs a number of times before arriving at a satisfactory representation of a system.

Another important point is that development of DFDs is not a ‘back room’ task. That is, it is not carried out in a laboratory but ideally with a user. As you draw DFDs many questions will arise: how are things done? What policies govern functions? What data is stored? How it is used? What data needs to be on an output flow? As a fact-finding technique, developing a DFD with a user has no equal, and there is the added benefit that the user identifies with the work and understands it. With this trend and philosophy of analysts teaming-up with users in developing information systems has been a driving force for the acceptance and adoption of Joint Applications Development (JAD) methods.

Self-test 5.2

1. What is meant by data flow conservation and synchronization?

2. Write a brief definition of these key terms:

   (a) decomposing a process
   (b) balancing
   (c) elementary process.

3. Explain the relationship between a Data Flow Diagram, a data dictionary and a process logic specifications.
The Hong Kong Sweatshop Company manufactures shirts, trousers, dresses and other clothes. Rolls of fabric are stored in warehouses until required for cutting and sewing into finished garments. Every six months there is a stock take of all rolls of fabric in stock. A computer-based information system is required to assist in this stock taking activity.

The rolls of fabric are stored in warehouses attached to each of the six factories located around Hong Kong. Rolls are normally about 100 metres long, but can vary from 60 to 120 metres. Each roll is examined, measured (to the nearest ten centimetres) and stored until required for cutting and sewing.

Every six months there is a manual stock take of all fabric in the six warehouses. It is not necessary to identify individual rolls (although each does have a unique roll number), just the total metres of each type of fabric at each warehouse. Each warehouse is identified by a single numeric digit. All staff participate in the stock take, which is carried out over a weekend so that stock can be ‘frozen’. Stock sheets are completed and returned to Head Office for analysis. Each stock sheet records up to 40 rolls, each entry containing a fabric code and a number of metres for a particular roll.

There are 120 different fabrics stored at present and potentially any fabric may be stored at any warehouse. Each fabric is identified by a five digit fabric code — the first two digits are a category code (there are fifteen categories: ‘01’ cotton, ‘02’ polyester, ‘03’ nylon etc.) and the remaining three are a sequential number within that category.

The Managing Director requires a computer-based information system to provide the following reports:

- stock valuation report — listing the total metres, price per metre and total value for each fabric type at each warehouse, with warehouse and company totals of value
- fabric by category — listing the total metres and total value of each fabric category (irrespective of warehouse location)

(a) draw a series of Data Flow Diagrams for this system. They should include a context diagram, a Diagram 0 and at least one lower level diagram. You should clearly label each diagram as being context diagram, D0 or lower level.

(b) write Data Dictionary entries for the data stores in this system;

(c) briefly describe all major processes at Diagram 0 level.

In answering this question, you may use any of the recognized systems analysis documentation methods. Annotate your answers with explanatory notes where necessary.
Summary

Systems analysis is used to discover information systems requirements and to specify them in a way that can be communicated unambiguously to both users and systems developers. Users must understand a specification to ensure that it is correct and consistent. Systems developers must understand a specification to translate it into an effective technological system. As such we have a common basis for this communication, it comes in the form of data flow diagrams and its companion process logic specifications, and data dictionary. In this Unit 5, you have learned how to construct and document them.

Your next unit, Unit 6, describes data modelling techniques. These techniques examine the underlying data structure of a system.
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.

Systems analysis and design


Glossary

This glossary provides brief definitions of the main technical terms you encountered in Unit 5. The definitions given here are offered in addition to, or in place of those found in your textbook.

**Black hole:** A black hole is a process where all flows connected with it are input flows.

**Conditions:** Representing simple rules or decision logic, this can be expressed using constructs of IF [condition] THEN [actions...].

**Data dictionary:** A data dictionary is a collection of data structures that specifies data stores and data flows that were identified in a DFD.

**Data flow conservation:** Data flow conservation states that flows should represent only relevant or usable data needed by and generated from the process.

**Data flow diagrams (DFD):** A tool that depicts the flow of data through a system and the work or processing performed by that system.

**Data flow:** Represents an input of data to a process, or the output of data (or information) from a process. A data flow is also used to represent the creation, deletion, or updating of data in a file or database.

**Data modelling:** A technique for organising and documenting a system’s data. Data modelling is sometimes called database modelling because a data model is usually implemented as a database. It is sometimes called information modelling.

**Data store:** A data store is essentially data at rest. It can be some temporary storage of output from a process or more permanently a data-file/database for input and output.

**Data structures:** Specific arrangements of data attributes that define the organization of a single instance of data flow.

**Decision table:** A decision table is a tabular form of presentation that specifies a set of conditions and their corresponding actions.

**Elementary process:** A process in a DFD that is not further exploded into a child DFD.

**External agents:** External agents are people/systems that are outside the boundary of the system but do interact with the processes in terms of providing input or receiving outputs from the system processes.

**External entities:** See External agents.

**Gray hole:** A gray hole reflect a process in which the inputs are insufficient or unbalanced to produce the output.

**Hierarchical partitioning:** See Process decomposition.
**Iteration:** An iteration structure specifies that a set of actions should be repeated based on some stated condition.

**Joint Application Development (JAD):** Uses highly organized and intensive workshops to bring together owners, users, analysts, designers, and builders to jointly define and design systems.

**Logical models:** Show what a system is or does. They are implementation independent; that is, they depict the system independent of any technical implementation. As such, logical models illustrate the essence of the system. Popular synonyms include essential model, conceptual model, and business model.

**Miracle:** A miracle is a process where all flows connected with it are output flows.

**Network modelling:** A technique for documenting the geographic structure of a system. It is also a diagrammatic technique used to document the shape of a business or information system in terms of its business locations.

**Physical models:** Show not only what a system is or does, but also how the system is physically and technically implemented. They are implementation dependent because they reflect technology choices and the limitations of those technology choices. Synonyms include implementation model and technical model.

**Primitive process:** See *Elementary process*.

**Process bubble:** See *Process*.

**Process decomposition:** It is the act of breaking a system into its component subsystems, processes, and sub-processes. Each level of abstraction reveals more or less detail as desired about the overall system.

**Process modelling:** The technique for organising and documenting the structure and flow of data through a system’s process and/or the logic, policies, and procedures to be implemented by a system’s processes.

**Process:** A process is a construct that represents a process within a system. It is usually identified as a round-cornered rectangle or simply a circle.

**Sequence:** This is a series of steps of action in structured English that typically starts with action.

**Structured English:** It is a language and syntax, based on the relative strengths of structured programming and natural English, for specifying the underlying logic of elementary processes on process models such as data flow diagrams.

**Synchronization:** Balancing of data flow diagrams at different levels of detail to preserve consistency and completeness of the models. Synchronization is a quality assurance technique.

**Systems approach:** By systems approach, a large problem (system) is broken down into more manageable components.
Answer key for self-test questions

Self-test 5.1

1 The difference between logical and physical models lies in their dependencies on implementation.

Logical models show what a system is or does. They are implementation-independent, that is, they depict the system independent of any technical implementation. Physical models show not only what a system is or does, but also how the system is physically and technically implemented. They are implementation-dependent because they reflect technology choices and the limitations of those technology choices.

2 Processes are often made up of many sub-processes. Under systems approach, process models are decomposable, which means, large and complicated processes can be broken down into smaller but manageable components.

3 Context diagram — graphic model of an information system that shows a flow of data and information between a system and external entities with which it interacts, to establish a context or setting for that system.

Diagram 0 — graphic system documentation and specification model that uses a symbol vocabulary to identify main processing functions, data flows, external entities and data storage points. It is the expanded level of the context diagram.

4 For Figure 5.3,
   • External agents: Customer, Comic publisher.
   • Process: Peak Comic Rental System.
   • Data Flows: New member form, New member card, Comics, Member card and Payments, Comic returns, Order, Comics and Bill, Payments.

For Figure 5.4,
   • External agents: Customer, Comic publisher.
   • Processes: New member process, Comic rental, Purchasing.
   • Data Flows: New member card, New member form, Customer data, Comics and receipt, Member card and Payments, Comic returns, Customer status, Rental transaction, Comic availability, Comic information, New comic entries, Order, Comics and Bill, Payments.
   • Data stores: Customer DB, Comic library.
For Figure 5.5,

- **External agent**: Customer.
- **Processes**: Comic check-out, Return process.
- **Data Flows**: Receipts, Payment, Member card, Comics, Comic returns, Comic price, Comic info, Customer status, Rental transaction, Customer-comic rental.
- **Data stores**: Customer DB, Comic library.

5 Expanded level 1 diagram of New Member Process:

**Expanded level 1 diagram of Purchasing Process:**
Self-test 5.2

1 Data flow conservation states that flows should represent only relevant or usable data needed by and generated from the process.

Synchronization is the balancing of data flow diagrams at different levels of detail to preserve consistency and completeness of the models. Synchronization is a quality assurance technique.

2 Decomposing a process — to expand a process in a DFD to a more detailed level for further scrutiny (to produce a child or lower-level DFD).

Balancing — to correlate parent and child elements of a DFD in terms of flows in and out and functions accomplished. A check for consistency.

Elementary process — a process in a DFD that is not further exploded into a child DFD.

3 All flows and stores on a Data Flow Diagram are described in alphabetical order in a data dictionary. All elementary processes on a DFD are described in process logic specifications. All data that are used in process logic specifications are described in a data dictionary.

4 (a) Context diagram
Diagram 0

1. Issue stock sheets
   - employees
   - completed-stock-sheet

2. Process stock sheets
   - fabric stock

3. Produce reports
   - report-request
   - stock-valuation
   - fabric-by-category

Managing Director

(b) stock-valuation-report
   {warehouse-number
     |fabric-code
     |fabric-metres
     |fabric-unit-price
     |fabric-value
   
   warehouse-value
   company-value

fabric-by-category-report
   {fabric-category
     |fabric-category-metres
     |fabric-category-value

DFD 3 produce reports

report-request

3.1 Produce stock val.
   - stock-valuation
   - fabric stock

3.2 Produce fabric by cat.
   - fabric-by-category
   - fabric stock
(c) Process 1: Issue Stock Sheets
   Every six months
   Send copies of stock sheets to warehouses
   Receive stock-take Request

Process 2: Process Stock Sheets
   Key Stock Sheets
   Sort Stock Records [on warehouse-number, fabric-code]
   For each fabric code
      Accumulate metres stored in this warehouse
      Write Fabric Stock file

Process 3: Produce Report
   Receive Report Request
   Print Store Valuation Report
   Print Fabric-by-Category Report
Unit 5

Processing modelling
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

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.

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

Overview  
Introduction  
Objectives  
Concept of process modelling  
Data Flow Diagrams (DFDs)  
  Modelling constructs  
  The systems approach  
  Process decomposition  
Design principles in DFDs  
Common DFD modelling errors  
Detail specifications for DFD  
  Process logic specification  
  Structured English  
  Decision table  
  Data flow and data store descriptions  
  Data structure constructs  
Constructing a set of DFDs  
Summary  
References  
  Systems analysis and design  
Glossary  
Answer key for self-test questions