CHAPTER 6 Structuring Requirement: Process Modeling

LEARNING OUTCOMES

By the end of this chapter, you should be able to:

1. Explain the use of Logic Modeling;
2. Use structured English, decision tables and decision trees to analyse, describe and document structured decisions;
3. Describe the guidelines in choosing appropriate logic modeling techniques for different situations; and
4. Explain software development strategies.

INTRODUCTION

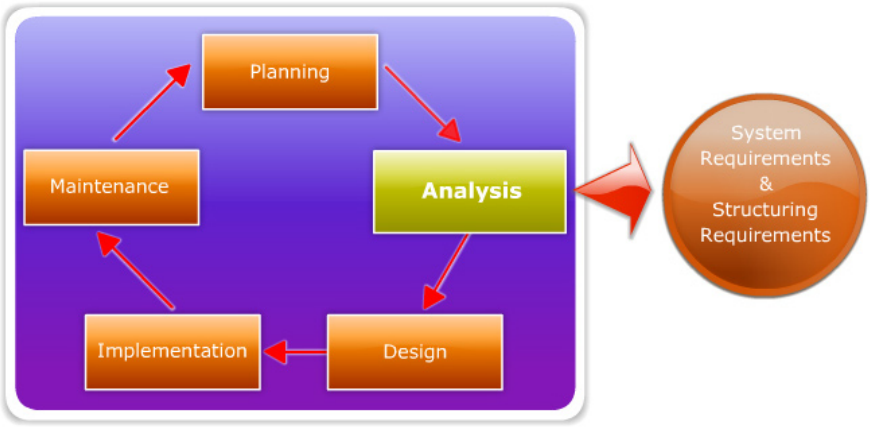
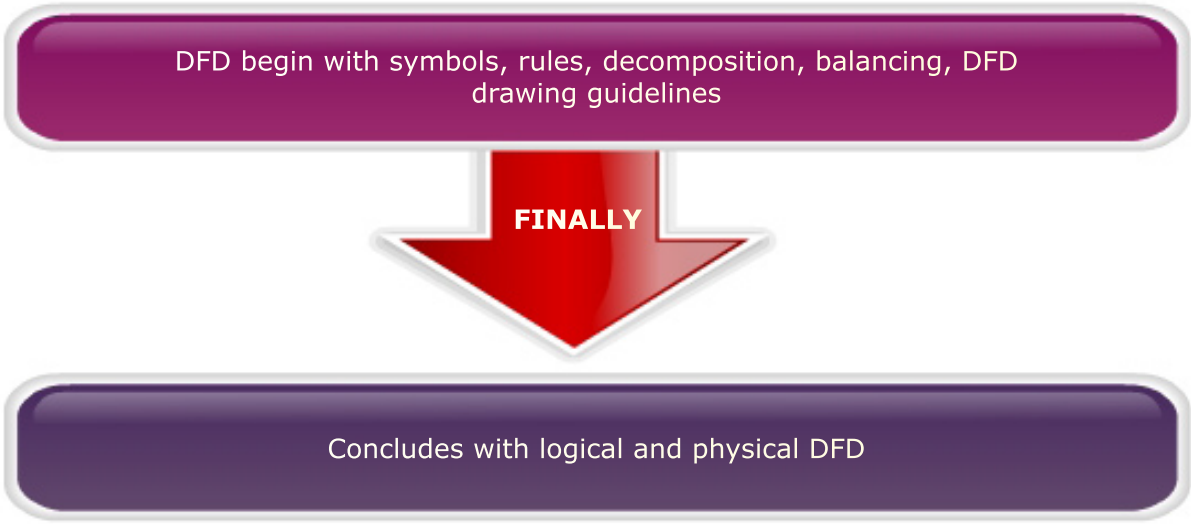


Figure 6.1: System Development Life Cycle (SDLC)

In this chapter, we continue our focus on the systems analysis part of the SDLC, which is highlighted in Figure 6.1 – Structuring Requirements. Note that there are two ways of structuring requirements which are process modeling and logic modeling. This chapter will focus on structuring requirements by modeling the process involved. 'Process Modeling' is a formal way of explaining how a system operates. It shows processes, activities, business functions and data flow among them. Process models can be used to document current systems or proposed systems. Although there are many process modeling techniques, this chapter focuses on one tool that is used to coherently represent the information gathered as part of system requirement - Data Flow Diagrams (DFD).

Data flow diagrams are versatile diagramming tools. It enables you to model how data flow through an information system, the relationships among the data flows and how data come to be stored at specific locations. Data flow diagrams also show the processes that change or transform data.



6.1 DFD SYMBOLS

Through a structured analysis technique called DFD, the systems analyst can put together a graphical representation of data processes throughout the organisation. By using combinations of only four symbols, the systems analyst can create a pictorial depiction of processes that will eventually provide solid system documentation.

Four basic symbols devised by Gane and Sarson (1979) are used to chart data movement on DFD as shown in Figure 6.2 below, a double square, an arrow, a rectangle with rounded corners and a flat rectangle (open on the right side and closed on the left side).

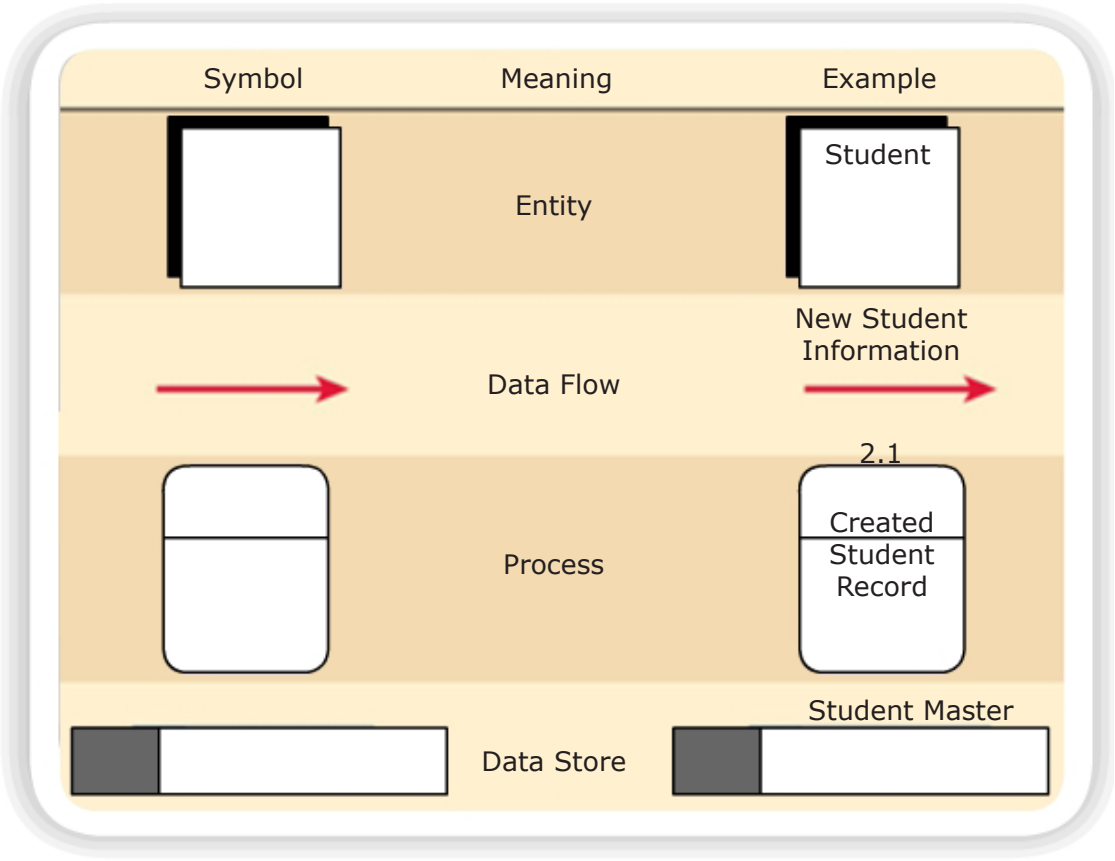


Figure 6.2: DFD symbols, meanings and examples (Source: Adapted from Gane and Sarson ,1979)

An entire system and numerous subsystems can be depicted graphically with these four symbols in combination.

6.1.1 Process

Process refers to an activity, function, or task that is done to achieve organisational objectives whether directly or indirectly. A process can be manual or computerised.

A process receives input data and produces output that has a different content, form, or both. For instance, the process of calculating pay uses two inputs (pay rate and hours worked) to produce one output (total pay). Process can be very simple or complex.

The symbol for a process is a rectangle with rounded corners. The name of the process appears inside the rectangle. The process name identifies a specific function and consists of a verb (and an adjective, if necessary) followed by a singular noun.

Examples of process names are: Register Student, Generate Report, Calculate Commission, and Assign Grade. The noun needs to be short but easily understood. In general, each process performs one activity. Therefore, try to avoid using comma and 'AND' because they show a process performing more than one activity. Processing details are not shown in a DFD.

For example you might have a process named Assign Grade. The process symbol does not reveal the business logic for the Assign Grade process. To document the logic, you need to create a process description, which will be explained later in following topic.

6.1.2 Data Flow

Data Flow consists of data that is moving or flowing as input to a process or as output from a process. The symbol for data flow is a line with an arrowhead.

Example of data flow like Name, Matric No. and Faculty. Data flow is labeled with a name that is made up of a noun, or a combination of a noun and an adjective, if required. Other examples of data flows are Payment, Customer Order, and Confirmed Order.

6.1.3 Data Store

A Data Store is used in DFD to represent data that the system stores. It is a situation in which data is stored to be used by processes inside a system, when needed.

For example,
A doctor stores all patient records to be referred to when a patient comes to get a medical examination. The name of data store is chosen according to the information or data being represented. Therefore, try to use a noun, or a combination of a noun and an adjective, if required. Examples of data stores are *Student File, Customer File, Treatments, Confirmed Orders, and Stock Database.*

In a DFD, the Gane and Sarson symbol for a data store is a flat rectangle that is open on the right side and closed on the left side. A data store must be connected to a process with a data flow. Data flow coming out of a data store shows that information is retrieved from the data store, while data flow going in shows that information is inputted (or record being added) into the data store. Data flowing in and out canrepresent the same data.

There is an exception to the requirement that a data store must have at least one incoming and one outgoing data flow. In some situations, a data store has no input data flow because it only used to store fixed reference data that is not updated by the system.

For example, consider a data store called Tax Table, which contain tax data from internal revenue service. When the company runs its payroll, the Calculate Tax process accesses data from this data store. On a DFD, this would be represented as a one-wayoutgoing data flow from Tax Table data store into the Calculate Tax process.

6.1.4 Entity

External Entity consists of individuals, institutions, units, departments, organisations or information systems that are present outside the system, but are interacting by giving input or getting output from the system. The symbol for entity is a double square, which may be shaded to make it look three-dimensional.

The word ‘external’ means that entities are located outside the system, but entities can be located inside or outside of the organisation. Confusion often occurs when determining entities that are supposed to be external entities. Errors in assigning individuals or objects inside the system as external entities are common. Individuals who execute the process inside the system, like clerks who input data, or workers who arrange inventory, are internal entities. Internal entities are normally stated inside process descriptions.

Examples of internal entities are: Account Executive of the Account Department who inputs employee data to the Human Resource System; and Manager who receives reports from the Customer Order System.

External entities are labeled according to the names that represent them. Other examples of external entity names are Doctor, Customer, University, and Student Information System. External entities show the boundaries of the system and how the system interacts with the outside environment.

For example,

Suppliers who supply raw materials to the inventory system are the external entities that give input data to the system. Customers who receive invoices and products are considered external entities that receive output data from the system.

External entities play two roles - as sources and as sink as shown in Figure 6.3.

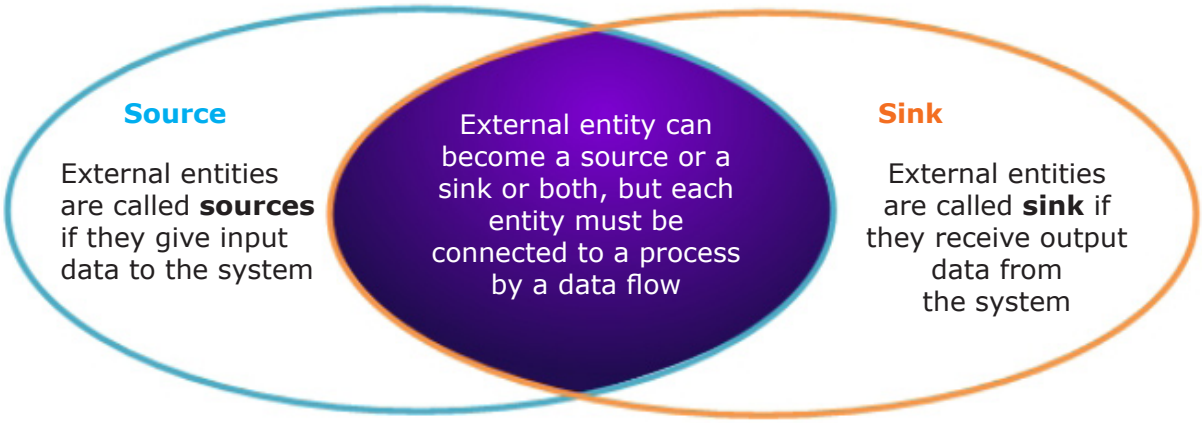


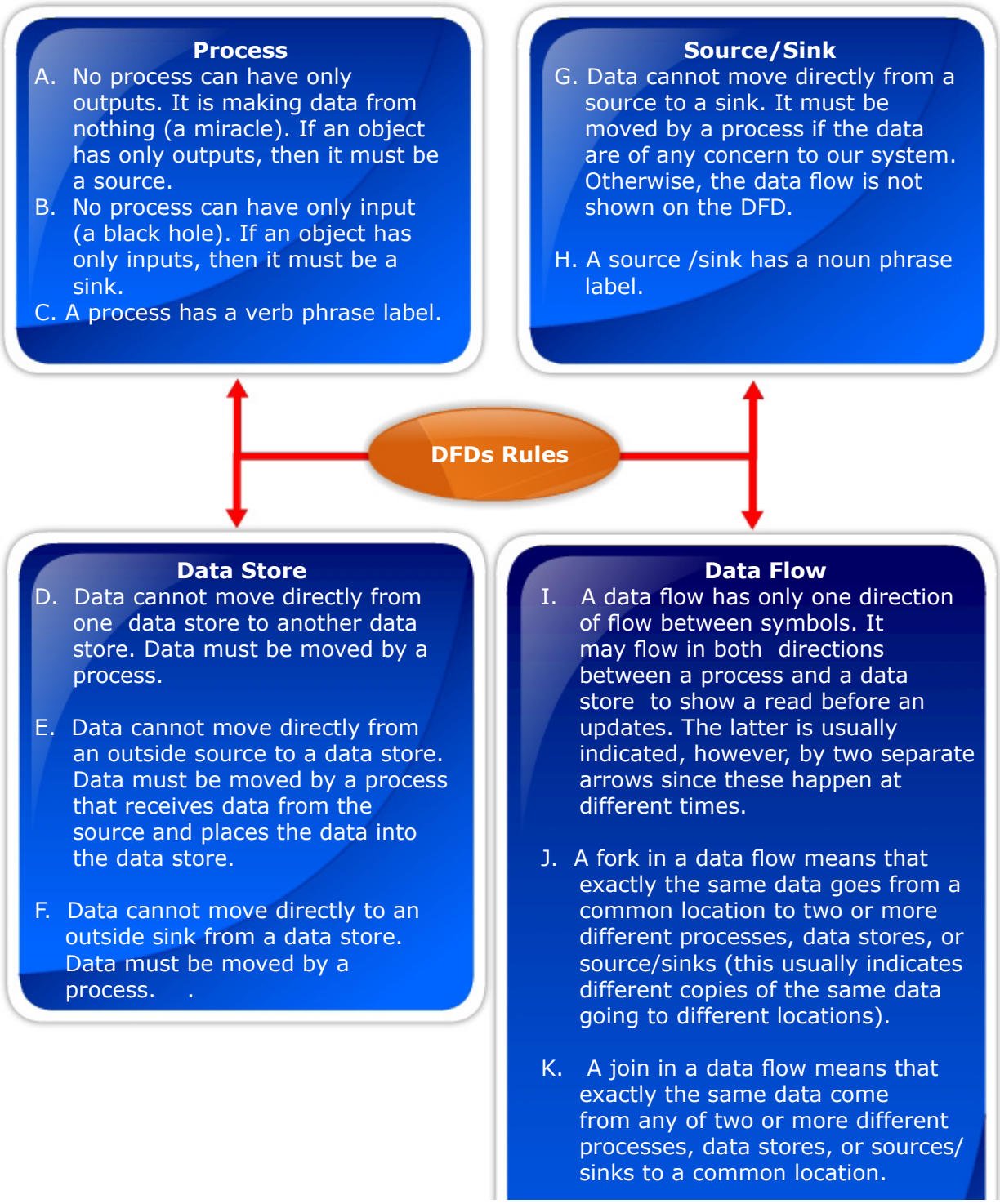
Figure 6.3: Roles of External Entities

ACTIVITY

What is a data-flow diagram? Why do system analysts use data-flow diagrams?

6.2 DFD RULES

There are several guidelines that need to be followed while drawing a DFD. These rules allow you to evaluate DFDs for correctness. The rules for DFDs are listed in Figure 6.4



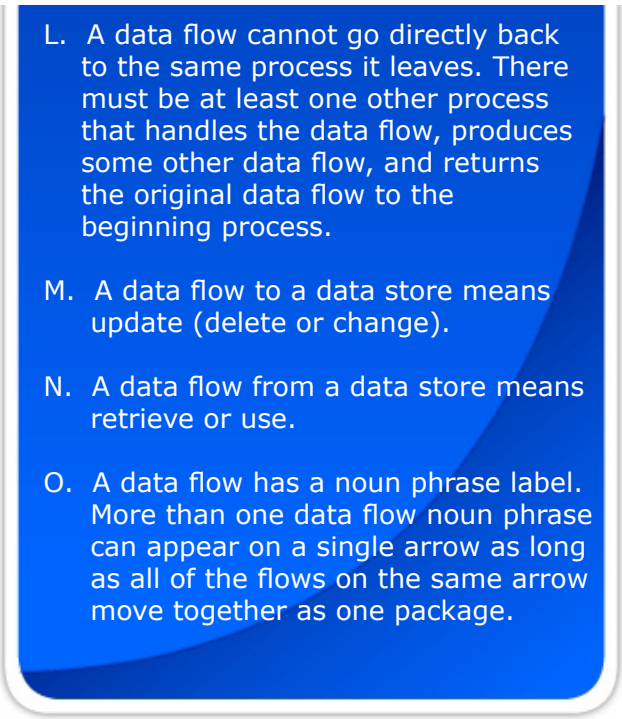


Figure 6.4: DFD rules (Source: Adapted from Celko, 1987)

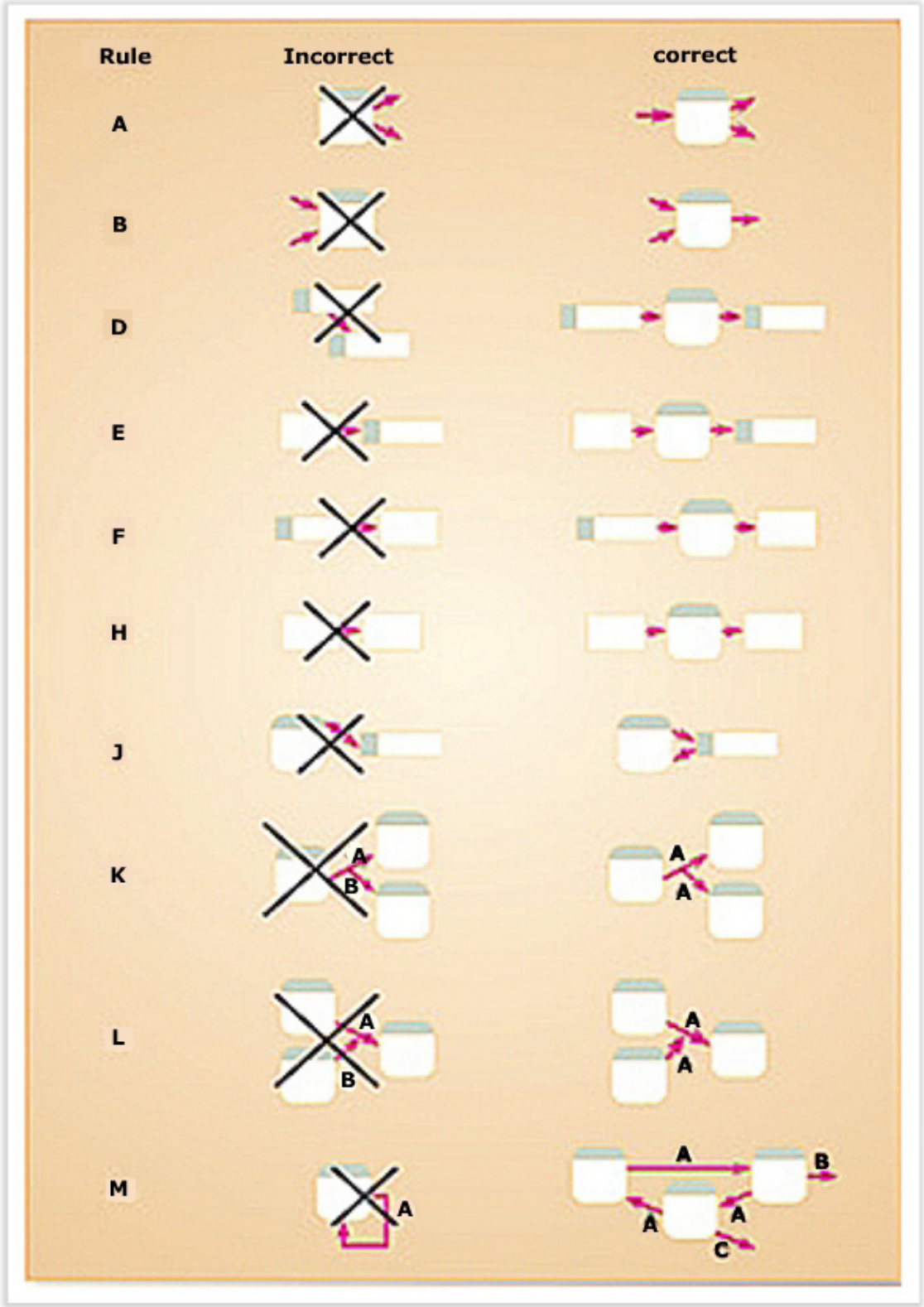
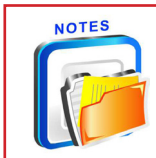


Figure 6.5: Illustrates incorrect and correct ways to draw DFD (Source: Adapted from Celko, 1987)



The rules that prescribe naming conventions (rules C, G, I and P) and those that explain how to interpret data flows in and out of data stores (rules N and O) are not illustrated in Figure 6.5.

Besides the above rules, there are two general guidelines that can be used:

- **Input to a process is different from its output:**

As described above, input and output to a process are different in terms of form, or content, or both. This is because a process aims to change input to output - not to merely transfer data without any change. What may happen is that the same input goes in and then comes out of a process, but the process also produces an additional output (a new data flow) as a result of processing the input - this is fine.

- **DFD symbols contain unique names:**

To avoid confusion and error, DFD symbols are named uniquely. If there are two processes that share the same name, then how can you differentiate between the two? Use a unique reference number. Besides using a name, a process is also labeled with a unique reference number.

6.3 CREATING DFD

The first step in constructing a set of DFDs is to draw a context diagram. Context Diagramis an overview of an organisational system that shows the system boundaries, external entities that interact with the system and the major information flows between the entities and the system. It is the highest level of DFD which represents the entire information system.

To draw a context diagram, you start by placing a single process symbol in the center of the page. The symbol represents the entire information system and you identify it as process 0. Then you place the entities around the perimeter of the process and use data flows to connect the entities to the central process. You do not show any data stores in a context diagram because data stores are internal to the system. How do you determine which entity to put in? By checking the system requirements, you can know the external entities that serve as the sources and sinks for the system.

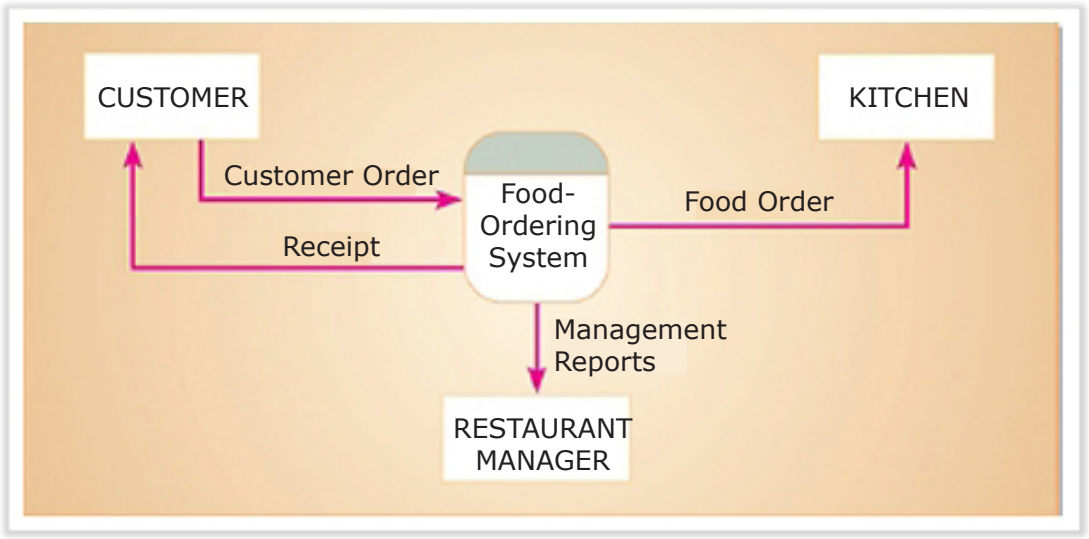


Figure 6.6: Context diagram for Food Ordering System
Source: Adapted from Hoffer et al (2005)

Context diagram example in Figure 6.6 shows the context diagram for a food ordering system. The system is at the center of the diagram. The three entities (Customer, Kitchen and Restaurant Manager) are placed around the central process. Interaction among central process and entities involves four different data flows (Customer Order, Receipt, Food Order and Management Report).

6.3.2 Level 0 Diagram

Context diagram provides most general view of information system and contains a single process symbol. To show details in the process, you have to create level zero diagrams. The level zero data flow diagram is the diagram at the level immediately below the context diagram. It expands the single process on the context diagram to show the major, high-level processes (or functions) within the system.

The analyst must determine which processes are represented by the single process in the context diagram. As you can see in Figure 6.7, we have identified four separate processes. The main processes represent the major functions of the system, and these major functions correspond to actions such as the following:

(a) Capturing data from different sources (e.g. Process 1.0)
(b) Maintain data source (e.g. Process 2.0 and 3.0)
(c) Producing and distributing data to different sinks (e.g. Process 4.0)
(d) High level descriptions of data transformation operations (e.g. Process 1.0)

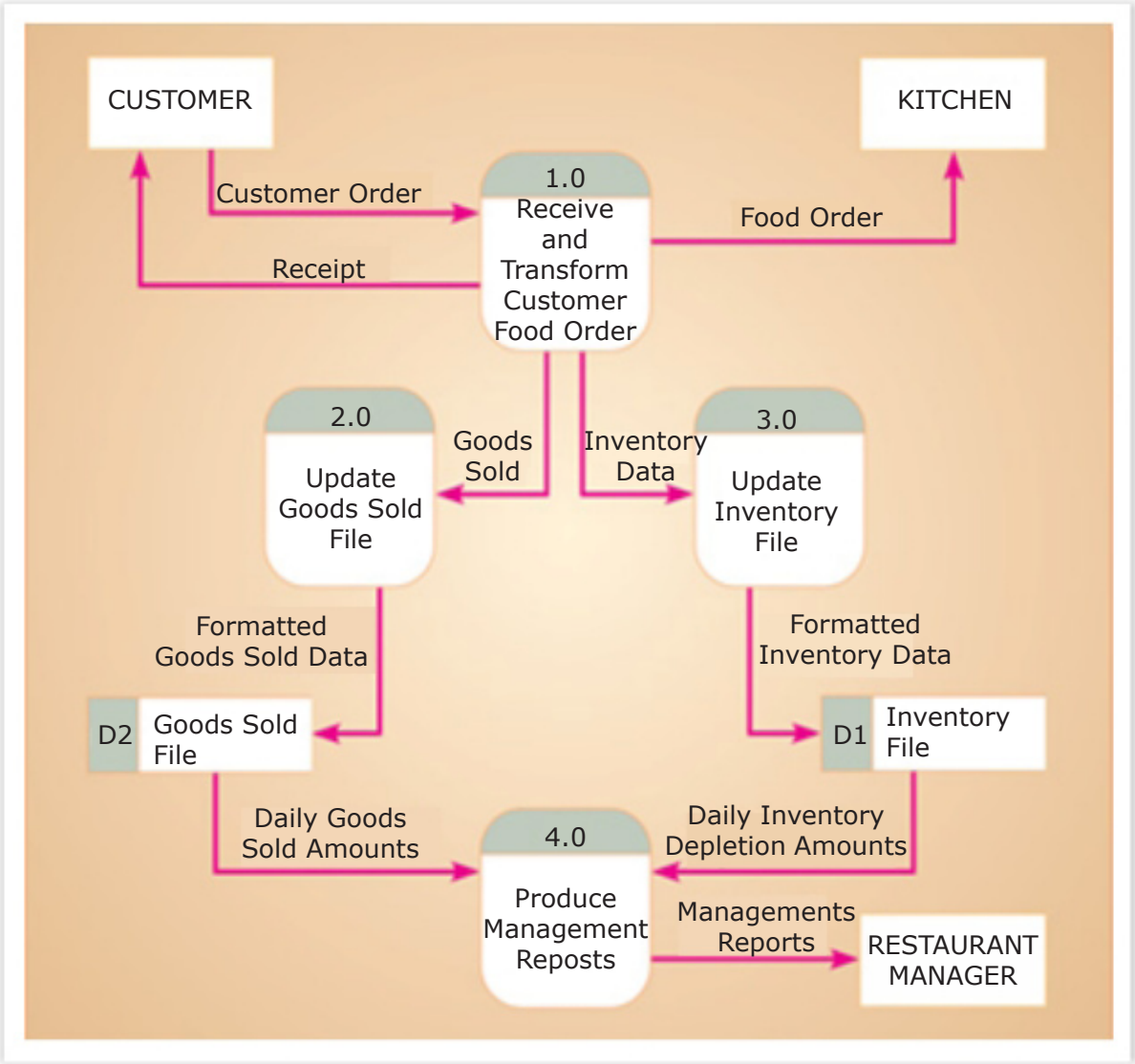


Figure 6.7: level 0 diagram for Food Ordering System
Source: Adapted from Hoffer et al (2005)

When you expand the context diagram into DFD diagram level zero, you must retain all the connections that flow into and out of process 0. It shows in Figure 6.7, the level 0 diagram is expanded to show four processes, two data stores and six new data flows. Notice that the number of each process ends in X.0 which corresponds to the level of the diagram: e.g. 1.0, 2.0, 3.0 etc.

ACTIVITY

What is a context-level data flow diagram? Contrast it to a level 0 DFD.

6.3.3 Lower Level DFDs

(a) Decomposition of DFDs

To create lower level diagrams, you must use decomposing and balancing techniques. **Decomposition** is an iterative process of breaking a system description down into finer and finer detail until functional primitives are identified. **Functional primitive** is the lowest level of a DFD. It is a process that cannot be exploded further. Decomposing also can be called exploding, partitioning and leveling. Figure 6.8 shows the process of DFDs decomposition. **Balancing** maintains consistency amongst a set of DFDs by ensuring that input and output data flows align properly which will be define later in this topic.

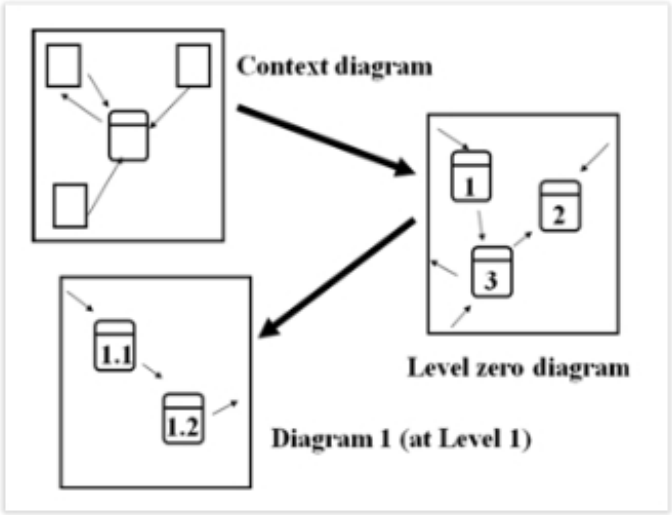


Figure 6.8: Decomposition of DFDs

Let's continue with our example of the Food Ordering System to see how a level 0 DFD can be further decomposed. The first process in Figure 6.7 called Receive and Transform Customer Food Order can be decomposed as shown in Figure 6.9.

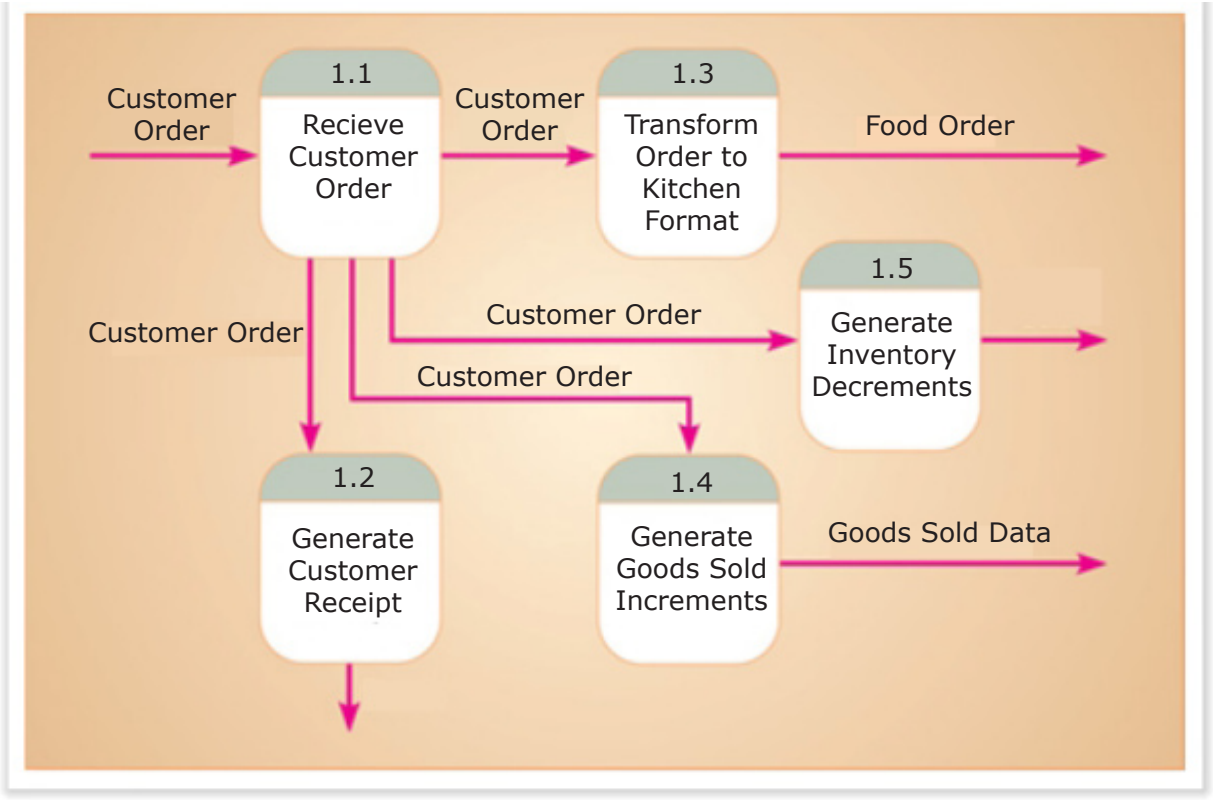


Figure 6.9: Level 1 diagram showing decomposition of Process 1.0 from level 0 diagram in Food Ordering System
Source: Adapted from Hoffer et al (2005)

Note that each of the five processes in Figure 6.9 is labeled as a sub process of Process 1.0. We can say Level 1 processes are numbered 1.1, 1.2, 1.3 and so on. Each Level 1 diagram and diagrams at lower levels (e.g. Level 2, Level 3) show only a part of the processing as the entire system is not being represented.

Although you may include source and sinks, the context and level-0 diagram have shown the sources and sink. If we should decide to decompose Process 2.0, 3.0 or 4.0 in a similar manner, the DFDs we would create also will be in level-1 diagram. In general, a level-n diagram is a DFD that is generated from n nested decompositions from a level-0 diagram. Figure 6.10 show decomposition of DFDs in details.

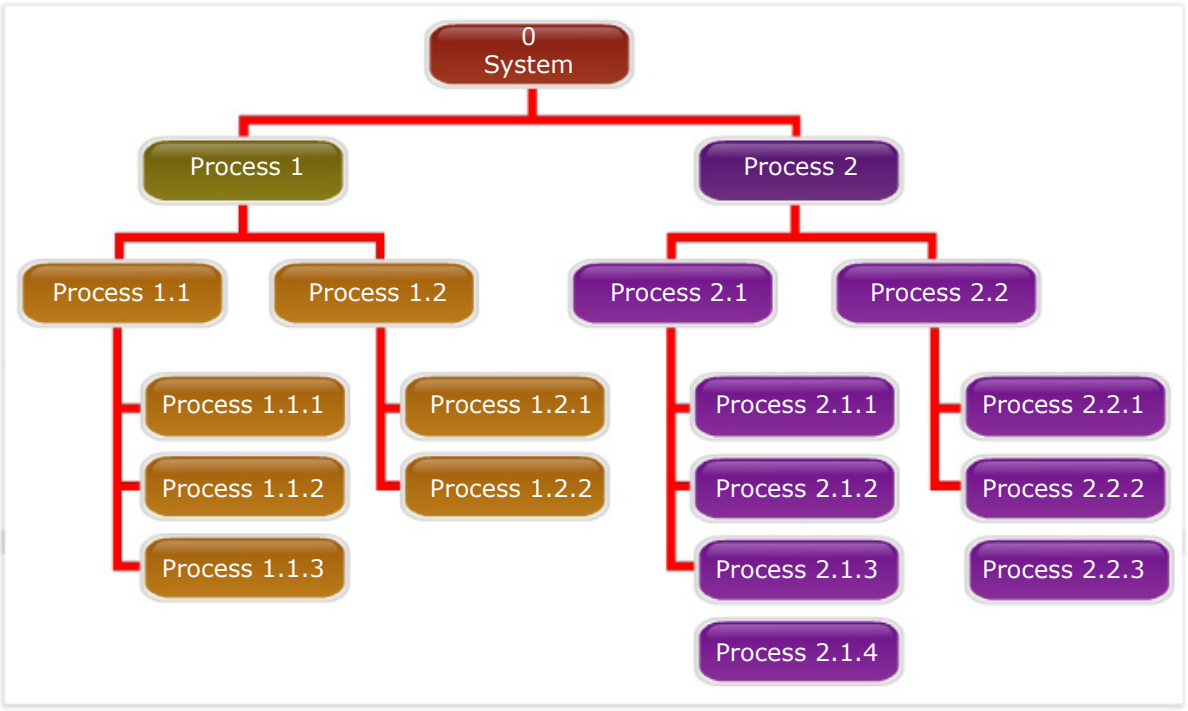


Figure 6.10: Decomposition of DFDs in details

(b) Balancing DFDs

Balancing ensures that the input and output data flows of the higher level DFD are maintained on the lower level DFD. In other words, Balanced means:

Number of inputs to lower level DFD equals number of inputs to associated process of higher-level DFD.

Number of outputs to lower level DFD equals number of outputs to associated process of higher-level DFD.

Figure 6.11 shows one example of what an unbalanced DFD could look like. This is an unbalanced DFD because the process of the context diagram has only one input but the Level-0 diagram has two inputs. These two DFDs are not balanced.

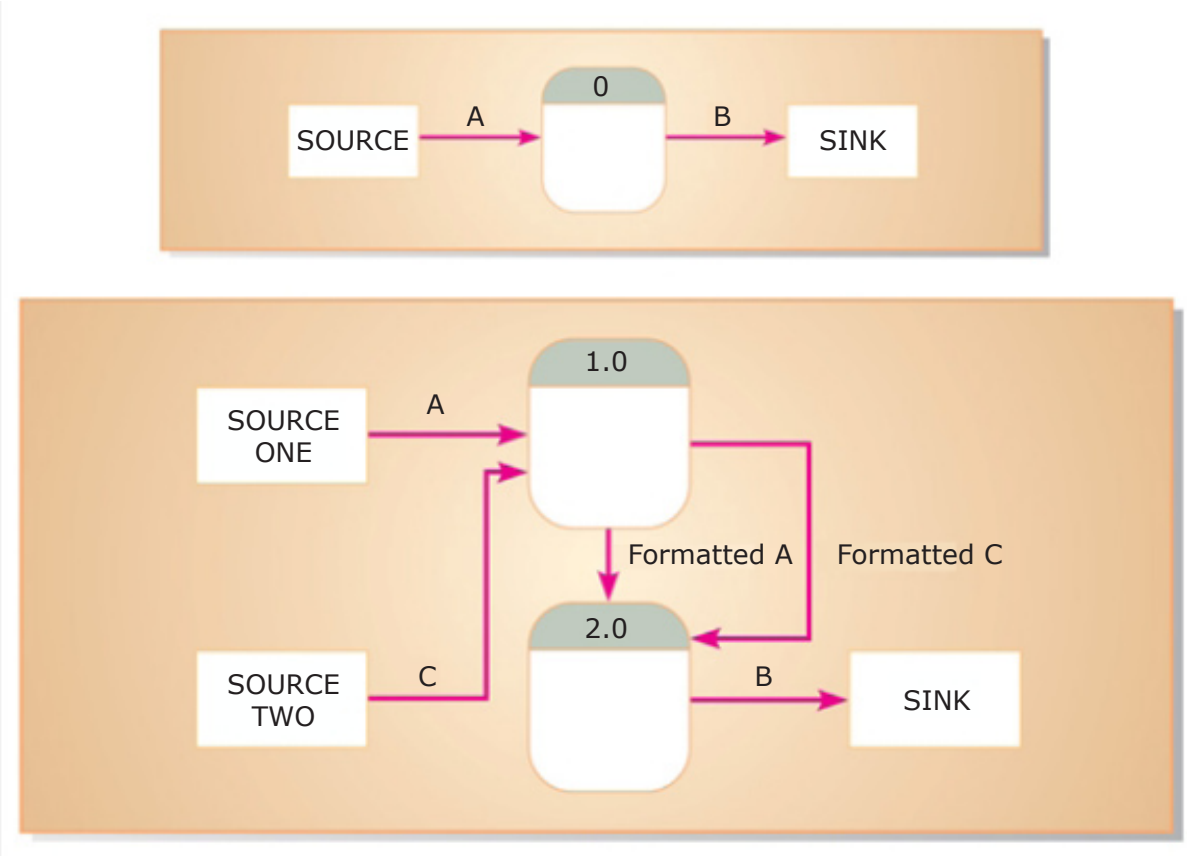
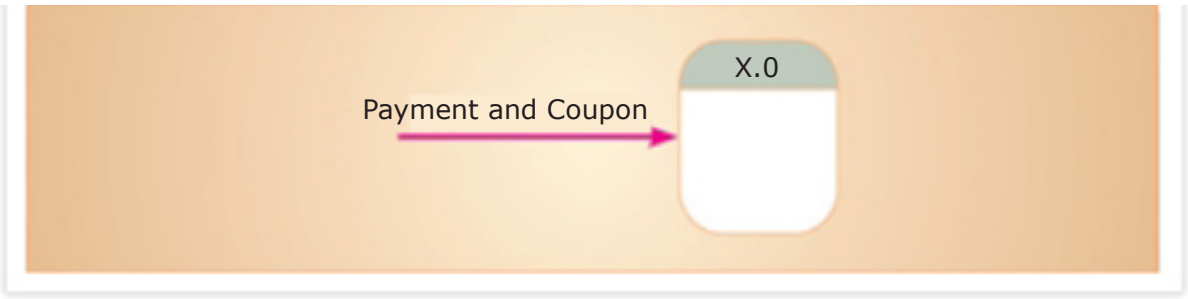


Figure 6.11: Unbalanced set of data flow diagrams
Source: Adapted from Hoffer et al (2005)

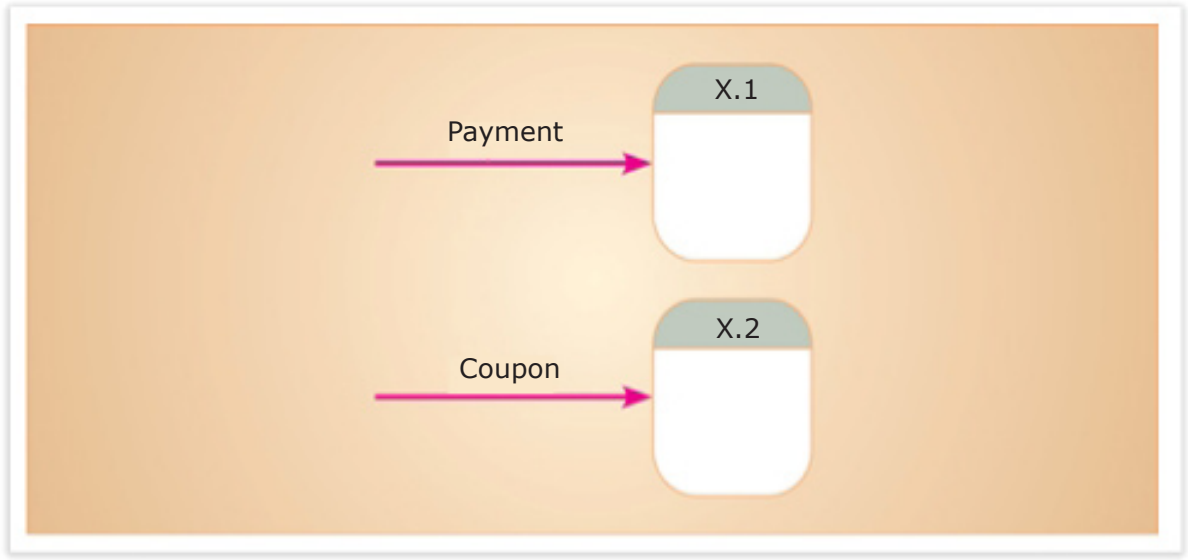
What happened with Figure 6.11 perhaps when drawing level-0 DFD, the analyst realises that the system also needed C in order to compute B. A and C were both drawn in the level-0 DFD, but the analyst forgot to update the context diagram. When making corrections, the analyst should also include SOURCE ONE and SOURCE TWO on the context diagrams. It is very important to keep DFDs balanced from the context diagram all the way through each level of diagram that you create.

A data flow consisting of several sub flows on a level-n diagram can be split apart on a level-n+1 diagram for a process that accepts this composite data flow as input.

For example, consider Figure 6.12. In this figure we see that a composite data flow Payment and Coupon is input to the process. That is the payment and coupon always flow together and input to the process at the same time. Then the process is decomposed into sub processes and each sub process receives one of the components of the composite data flow from the higher level DFD. These diagrams are still balanced because exactlythe same data are included in each diagram.



(a) Composite Data Flow



(b) Disaggregated Data Flow

Figure 6.12: Example of data flow splitting
Source: Adapted from Hoffer et al (2005)

ACTIVITY

What is decomposition? What is balancing? How can you determine if DFDs are not balanced?

6.4 GUIDELINES FOR DRAWING DFD

Learning the guidelines of drawing DFD is important because data flow diagrams have proven to be essential tools for process modeling which is part of the structuring requirements phase. The guidelines that need to be considered when drawing DFD include completeness, consistency, timing, iterative development and primitive DFDs as shown in Figure 6.13.

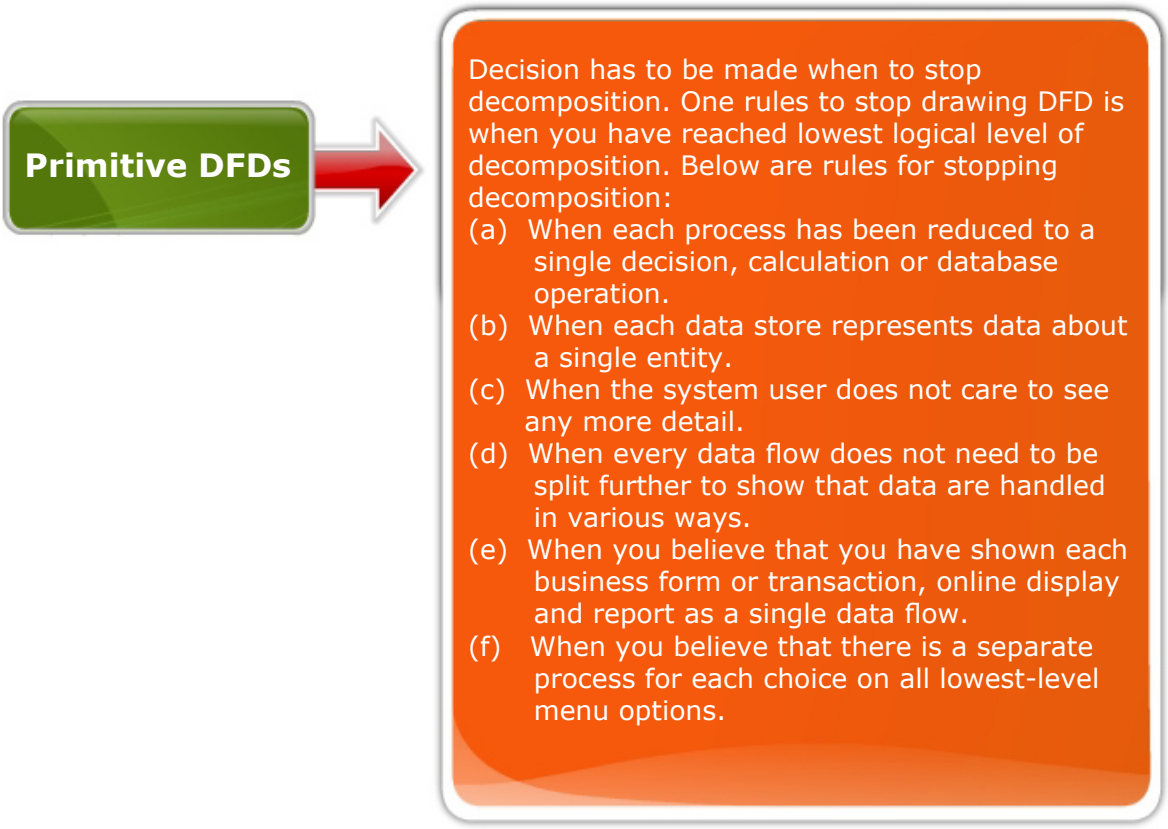


Figure 6.13: Drawing DFDs guidelines

Using the guidelines will help you to create DFDs that are correct, robust and accurate representations of the information system you are modeling. Having mastered the skills of drawing good DFDs, you can now use them to support the analysis process in system development life cycle.

1. Explain what the term DFD consistency means and provide an example.
2. Explain what the term DFD completeness means and provide an example.

6.5 LOGICAL AND PHYSICAL DFD

DFDs are categorised either logical or physical. **Logical DFD** shows what a system is or does. It only shows what processes and data are involved, without concern with how the system will be implemented.

Physical DFD show not only what a system is or does but also how the processes are actually implemented. Physical DFD must be consistent with logical DFD.

For example, a logical DFD for the Airline Reservation System contains a ticket booking process and a ticket issuing process. A physical DFD for this system shows the way both processes are implemented: e.g. the ticket booking process via the telephone, and ticket printing from the computer system.

A logical system does not provide information on whether:

(a) A certain process is implemented manually or by the computer, or
(b) Data is stored on paper files or floppy discs, or
(c) Data is obtained via telephones or e-mails.

Why do you not need to know all these technical aspects? Well, the objective of a logical diagram is to let you understand the processes needed for the business so that you can define the system requirements correctly, and think about the process implementation in detail at a later stage. Process implementation will be defined in detail in the design phase of SDLC by translating the logical models into physical models.

Table 6.1: Show the Comparison Between Physical and Logical DFD

Logical DFDs	Physical DFDs
• Represent what a system must do regardless of how it is implemented • They are technology independent • They show what processing, data movements and data storage must occur in a system • They show the essential aspects of a system	• Represent a particular way of implementing the processes and data in a system • They are technology dependent – they specify particular methods of doing tasks • They show the essential aspects of a system

So, once you have understood the system fairly well, with or without the DFD of the current system, you should then focus on drawing the logical DFD for the new system in great detail, during the analysis phase. This will enable you to produce a complete set of the physical DFD of the new system, based on your implementation plan, during the design phase.

Figure 6.13 shows an example of the logical and physical DFD for a Sale System of a supermarket. Customer brings Items to the Cashier counter. The Cost of each Item is Checked and totaled. Payment is made to the Cashier. Finally, the Customer is given the Receipt. Logical DFD only shows what sales processes are involved without clarifying how the processes are implemented. Through a physical DFD, Customers can know that processes are realised by checking the UPC Code using the Scanner, and Payment is received in the form of Cash or Cheque.

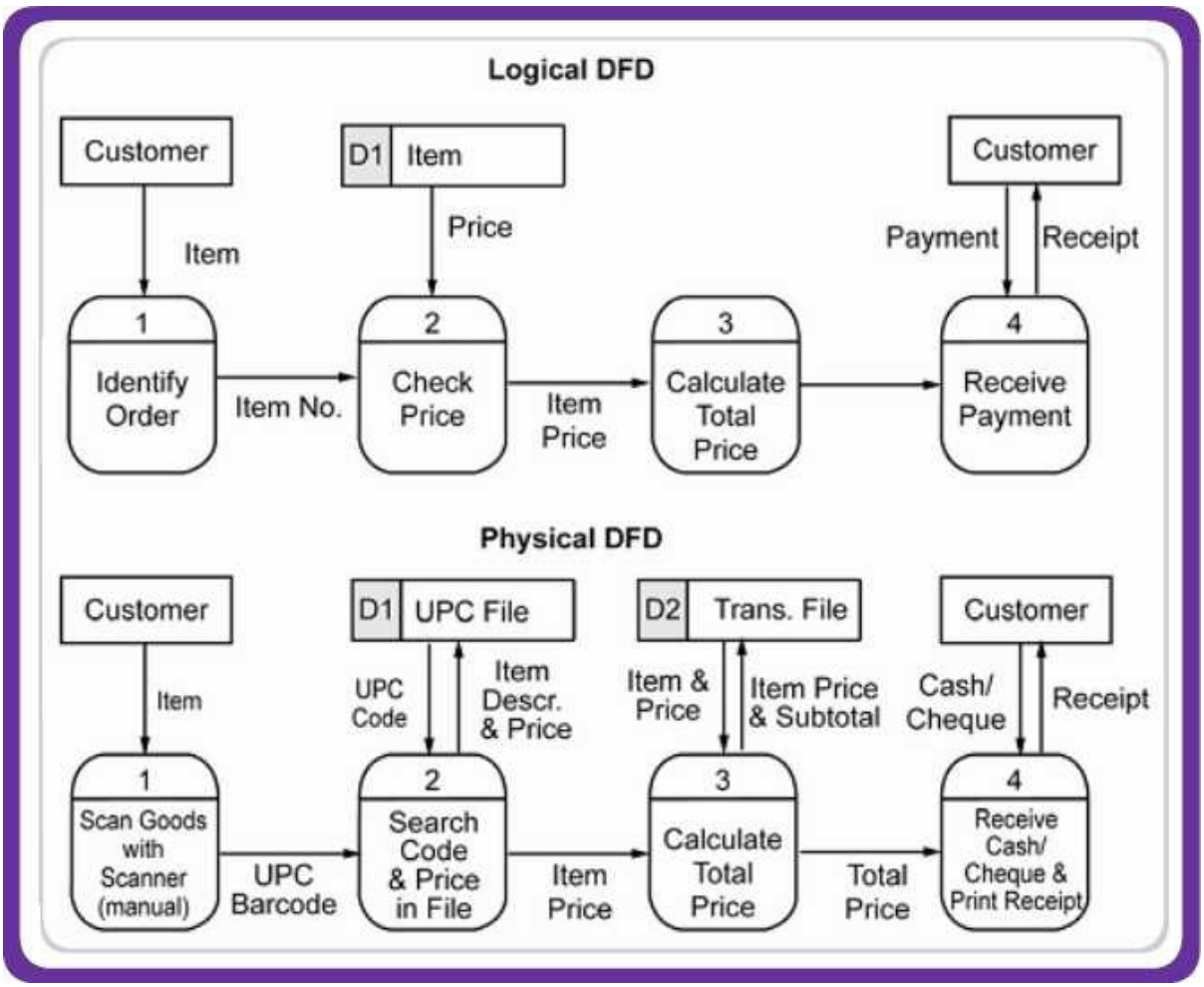


Figure 6.13: Drawing DFDs guidelines

SUMMARY

1. Process modeling can be done in many different ways, but this topic has focused to use data flow diagrams (DFD) as a way to structuring requirement. DFDs are useful to graphically show the movement and transformation of data in the information system. DFDs use four symbols to represent a process model: data flows, data stores, processes and external entity.
2. The general overview of the system can be seen through the Context Diagram as the highest level of DFD. Details of the system are then shown by DFD at lower levels, beginning with Level-0 DFD, followed by Level-1 DFD, Level-2 DFD, and so on until the Primitive DFD. The DFD drawn must take into account he concept of completeness, balancing, consistency to ensure that the diagrams are correct as the system being modeled were timeless. You should be willing to revise DFDs lesson several times for better understanding. Following these guidelines you can produce DFDs that are correct, robust and accurate representations of the information system.

REFERENCES

1. Celko, J. 1987. "I. Data Flow Diagrams." Computer Language 4 (January): 41-43
2. Gane, C., and T. Sarson. 1979. Structured Systems Analysis. Upper Saddle River, NJ: Prentice Hall.
3. Hoffer, J.A., George, J.F. and Valacich (2008) 5th ed., Modern Systems Analysis and
4. Design, Benjamin/Cummings, Massachusetts.
5. Kendall, E.K. and Kendall E. J (2008) 7th ed., Systems Analysis and Design, Upper Saddle
6. River, NJ: Prentice Hall.

KEY TERMS

Balancing

Data flow