Data Flow Diagrams (DFDs)

Specific Instructional Objectives

At the end of this lesson the student will be able to:

- Identify the activities carried out during the structured analysis phase.
- Explain what a DFD is.
- Explain why constructing DFDs are important in arriving at a good software design.
- Explain what a data dictionary is.
- Explain the importance of data dictionary.
- Identify whether a DFD is balanced.

Structured Analysis

Structured analysis is used to carry out the top-down decomposition of a set of high-level functions depicted in the problem description and to represent them graphically. During structured analysis, functional decomposition of the system is achieved. That is, each function that the system performs is analyzed and hierarchically decomposed into more detailed functions. Structured analysis technique is based on the following essential underlying principles:

- Top-down decomposition approach.
- Divide and conquer principle. Each function is decomposed independently.
- Graphical representation of the analysis results using Data Flow Diagrams (DFDs).

Data Flow Diagram (DFD)

The DFD (also known as a bubble chart) is a hierarchical graphical model of a system that shows the different processing activities or functions that the system performs and the data interchange among these functions. Each function is considered as a processing station (or process) that consumes some input data and produces some output data. The system is represented in terms of the input data to the system, various processing carried out on these data, and the output data generated by the system. A DFD model uses a very limited number of primitive symbols [as shown in fig. 5.1(a)] to represent the functions performed by a system and the data flow among these functions.

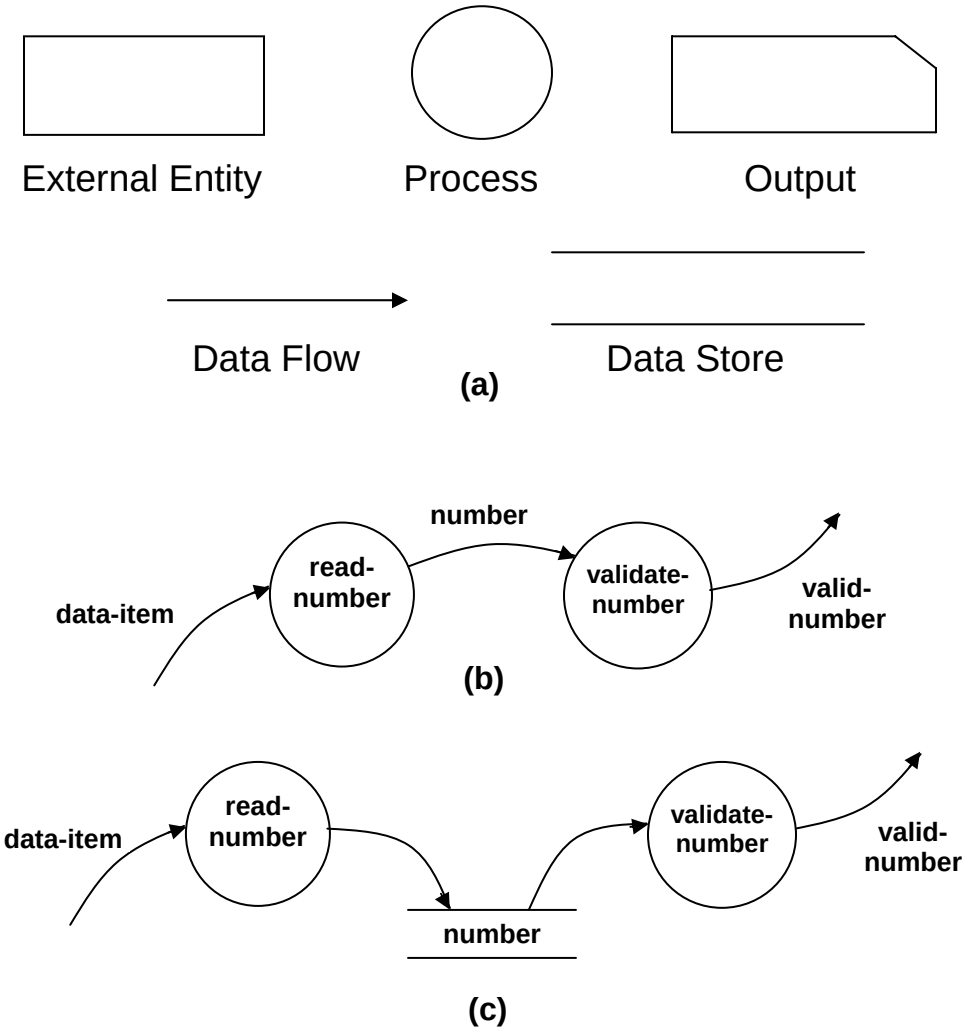


Fig. 5.1 (a) Symbols used for designing DFDs
(b), (c) Synchronous and asynchronous data flow

Here, two examples of data flow that describe input and validation of data are considered. In Fig. 5.1(b), the two processes are directly connected by a data flow. This means that the 'validate-number' process can start only after the 'read-number' process had supplied data to it. However in Fig 5.1(c), the two processes are connected through a data store. Hence, the operations of the two bubbles are independent. The first one is termed 'synchronous' and the second one 'asynchronous'.

Importance of DFDs in a good software design

The main reason why the DFD technique is so popular is probably because of the fact that DFD is a very simple formalism – it is simple to understand and use. Starting with a set of high-level functions that a system performs, a DFD model

hierarchically represents various sub-functions. In fact, any hierarchical model is simple to understand. Human mind is such that it can easily understand any hierarchical model of a system – because in a hierarchical model, starting with a very simple and abstract model of a system, different details of the system are slowly introduced through different hierarchies. The data flow diagramming technique also follows a very simple set of intuitive concepts and rules. DFD is an elegant modeling technique that turns out to be useful not only to represent the results of structured analysis of a software problem, but also for several other applications such as showing the flow of documents or items in an organization.

Data dictionary

A data dictionary lists all data items appearing in the DFD model of a system. The data items listed include all data flows and the contents of all data stores appearing on the DFDs in the DFD model of a system. A data dictionary lists the purpose of all data items and the definition of all composite data items in terms of their component data items. For example, a data dictionary entry may represent that the data **grossPay** consists of the components regularPay and overtimePay.

grossPay = regularPay + overtimePay

For the smallest units of data items, the data dictionary lists their name and their type. Composite data items can be defined in terms of primitive data items using the following data definition operators:

- **+**: denotes composition of two data items, e.g. **a+b** represents data a and **b**.
- **[,,]**: represents selection, i.e. any one of the data items listed in the brackets can occur. For example, **[a,b]** represents either **a** occurs or **b** occurs.
- **()**: the contents inside the bracket represent optional data which may or may not appear. e.g. **a+(b)** represents either **a** occurs or **a+b** occurs.
- **{}**: represents iterative data definition, e.g. **{name}5** represents five **name** data. **{name}*** represents zero or more instances of **name** data.
- **=**: represents equivalence, e.g. **a=b+c** means that **a** represents **b** and **c**.
- **/* */**: Anything appearing within **/*** and ***/** is considered as a comment.

Example 1: Tic-Tac-Toe Computer Game

Tic-tac-toe is a computer game in which a human player and the computer make alternative moves on a 3×3 square. A move consists of marking previously unmarked square. The player who first places three consecutive marks along a straight line on the square (i.e. along a row, column, or diagonal) wins the game. As soon as either the human player or the computer wins, a message congratulating the winner should be displayed. If neither player manages to get three consecutive marks along a straight line, but all the squares on the board are filled up, then the game is drawn. The computer always tries to win a game.

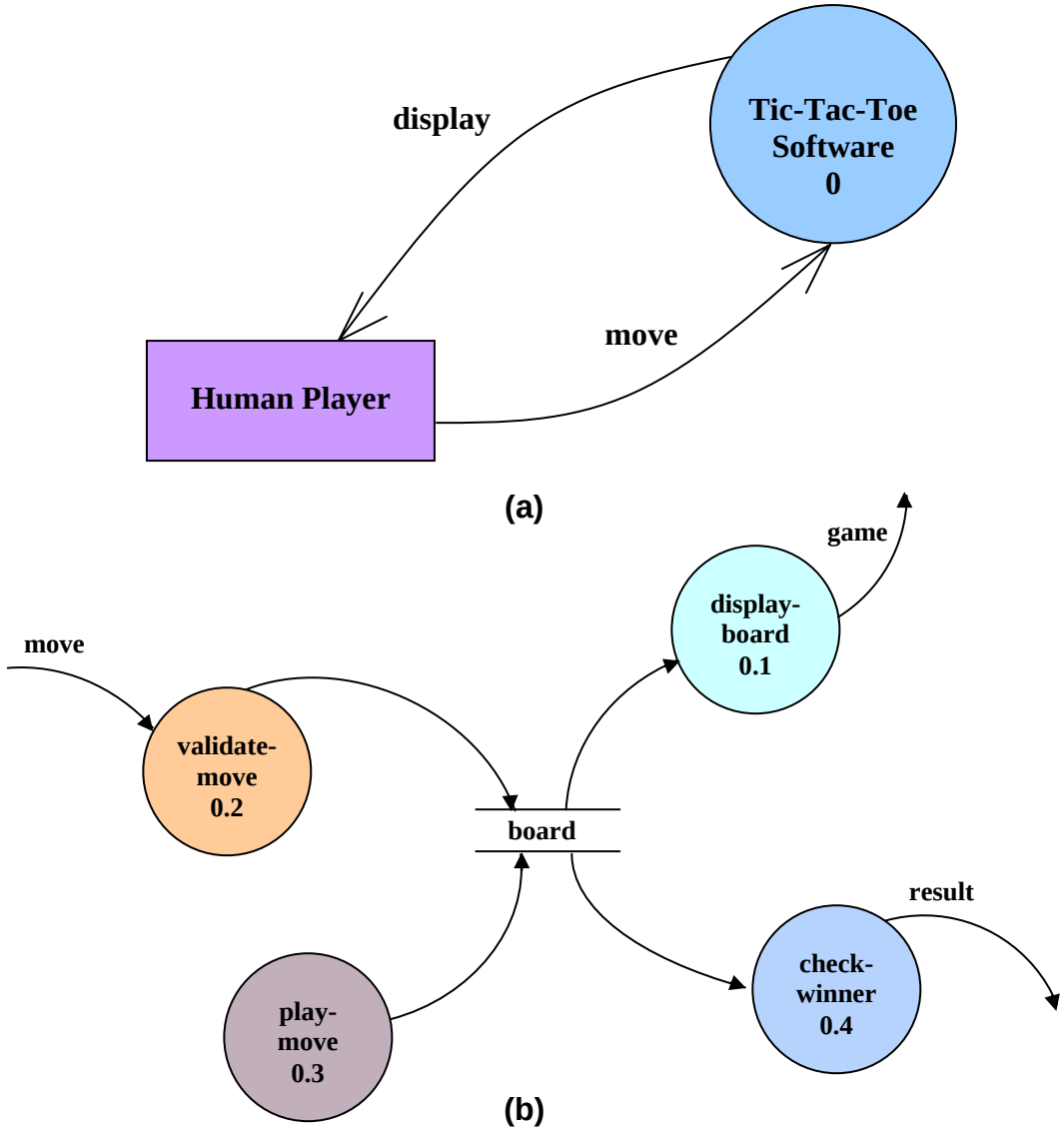


Fig 5.2 (a), (b) Level 0 and Level 1 DFD for Tic-Tac-Toe game described in Example 1

It may be recalled that the DFD model of a system typically consists of several DFDs: level 0, level 1, etc. However, a single data dictionary should capture all the data appearing in all the DFDs constituting the model. Figure 5.2 represents the level 0 and level 1 DFDs for the tic-tac-toe game. The data dictionary for the model is given below.

Data dictionary for the DFD model in Example 1

```
move:          integer /*number between 1 and 9 */
display:       game+result
game:          board
board:         {integer}9
result:        [“computer won”, “human won” “draw”]
```

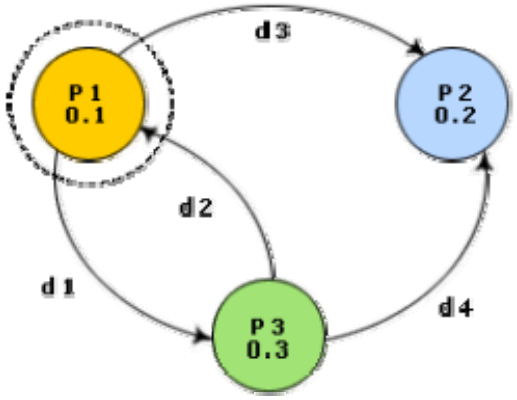
Importance of data dictionary

A data dictionary plays a very important role in any software development process because of the following reasons:

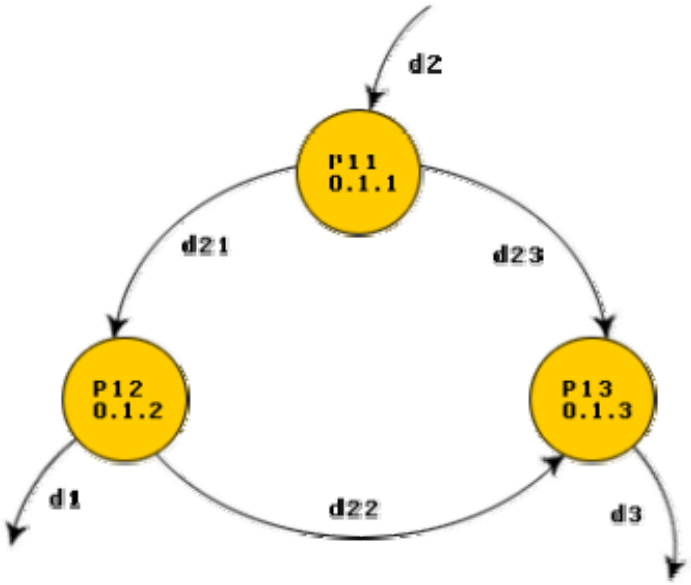
- A data dictionary provides a standard terminology for all relevant data for use by the engineers working in a project. A consistent vocabulary for data items is very important, since in large projects different engineers of the project have a tendency to use different terms to refer to the same data, which unnecessary causes confusion.
- The data dictionary provides the analyst with a means to determine the definition of different data structures in terms of their component elements.

Balancing a DFD

The data that flow into or out of a bubble must match the data flow at the next level of DFD. This is known as balancing a DFD. The concept of balancing a DFD has been illustrated in fig. 5.3. In the level 1 of the DFD, data items d1 and d3 flow out of the bubble 0.1 and the data item d2 flows into the bubble 0.1. In the next level, bubble 0.1 is decomposed. The decomposition is balanced, as d1 and d3 flow out of the level 2 diagram and d2 flows in.



(a) Level 1 DFD



(b) Level 2 DFD

Fig. 5.3: An example showing balanced decomposition