SOFTWARE ENGINEERING
LECTURE -10
29/01/21

STRUCTURED ANALYSIS

Structured analysis is used to carry out the top-down decomposition of a set of high-level functions present 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

*(1) [ ] → External entity → ~~function symbol / process or bubble~~.

An external entity such as librarian, user etc. The external entities are essentially those physical entities, external to the s/w system which interact with the system by inputting data to the system.

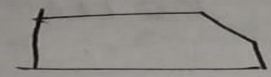
→ like human user, external s/w, H/w and another application sys.

function symbol / Process / bubble are used to represent function.

Data flow symbol →

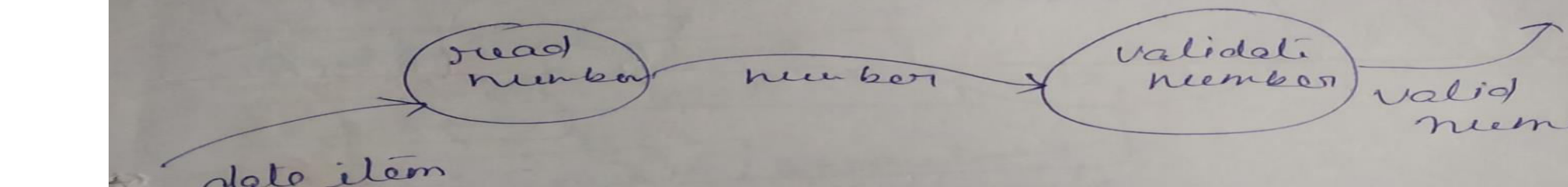
Data Store Symbol → Data Store Symbol represent the stored data, The direction of the data flow arrow shows whether data is being read from or written into a data store.

put Symbal → The output symbal is use for output data storing

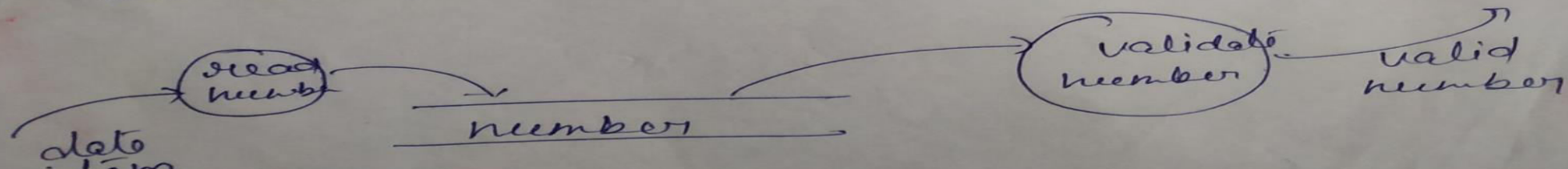


Synchronous and asynchronous operations

If two bubbles are directly connected by data flow arrow, then they are synchronous. This means they operate at the same speed.



If Two bubbles are connected through data store, the the speed of operation of the bubbles are independent.



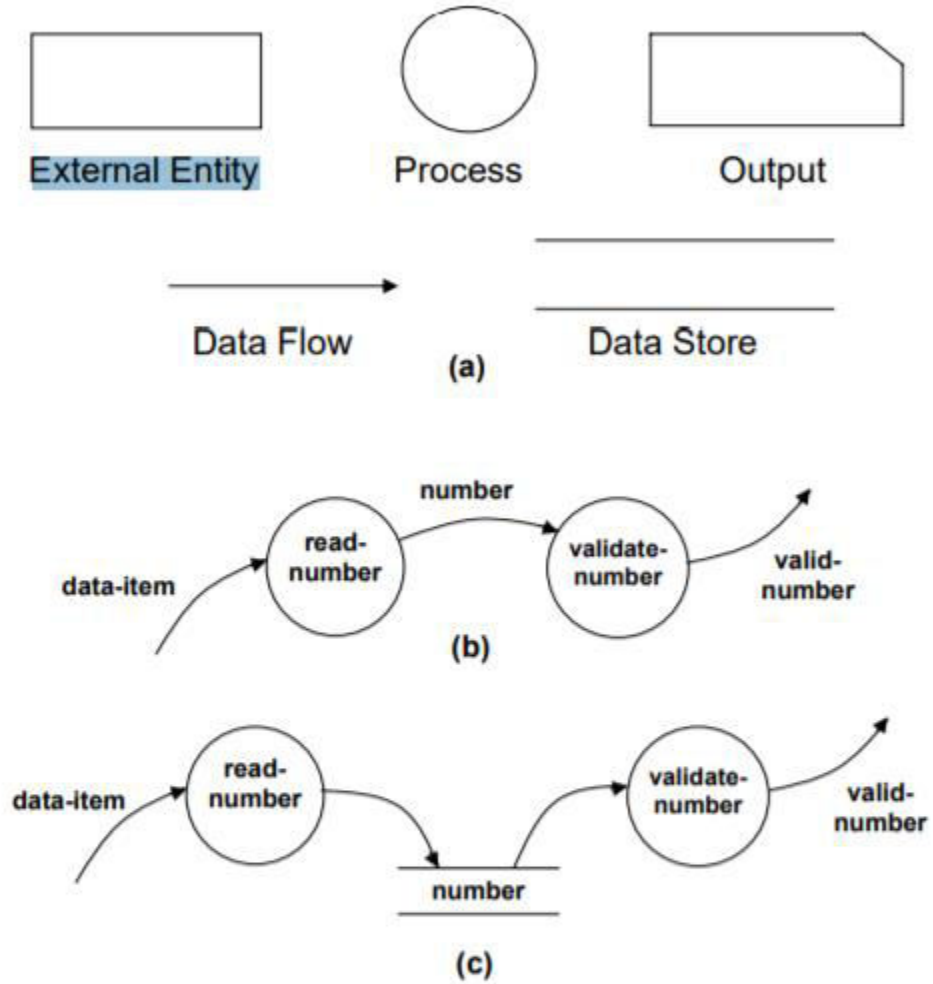


Fig. 5.1 (a) Symbols used for designing DFDs
(b), (c) Synchronous and asynchronous data flow

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'.

numbering of bubbles

Contact level – or zero level = 0

1 " = 0·1 , 0·2 , 0·3

2 " = 0·2·1 0·2·2, 0·2

It is popularized in 1970

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.

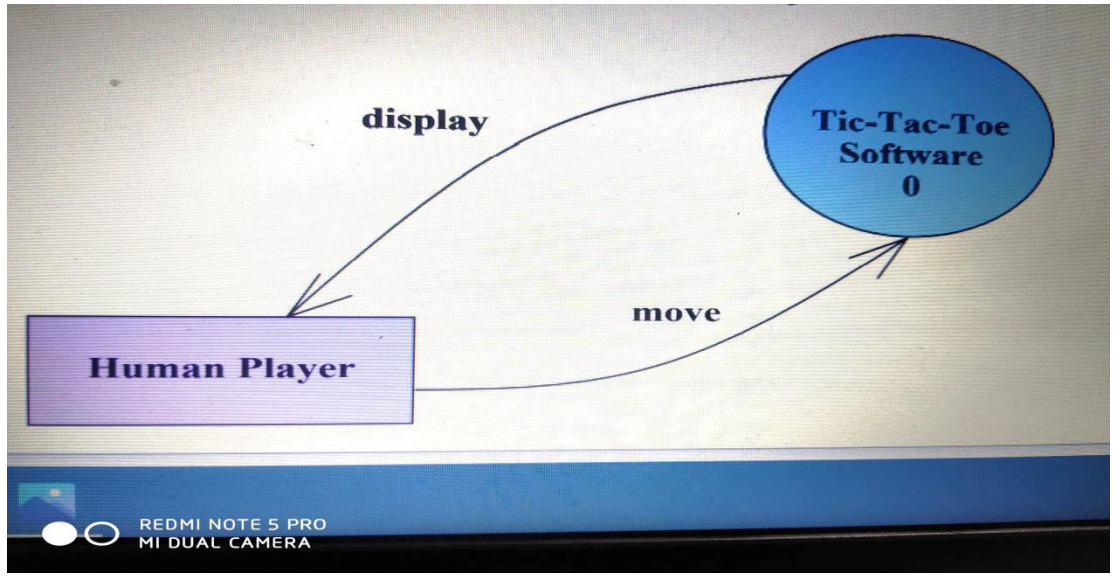
(): the contents inside the bracket represent optional data which may or may not appear. e.g. a+(b) represents either a occurs or a+b occurs.

{}: 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.



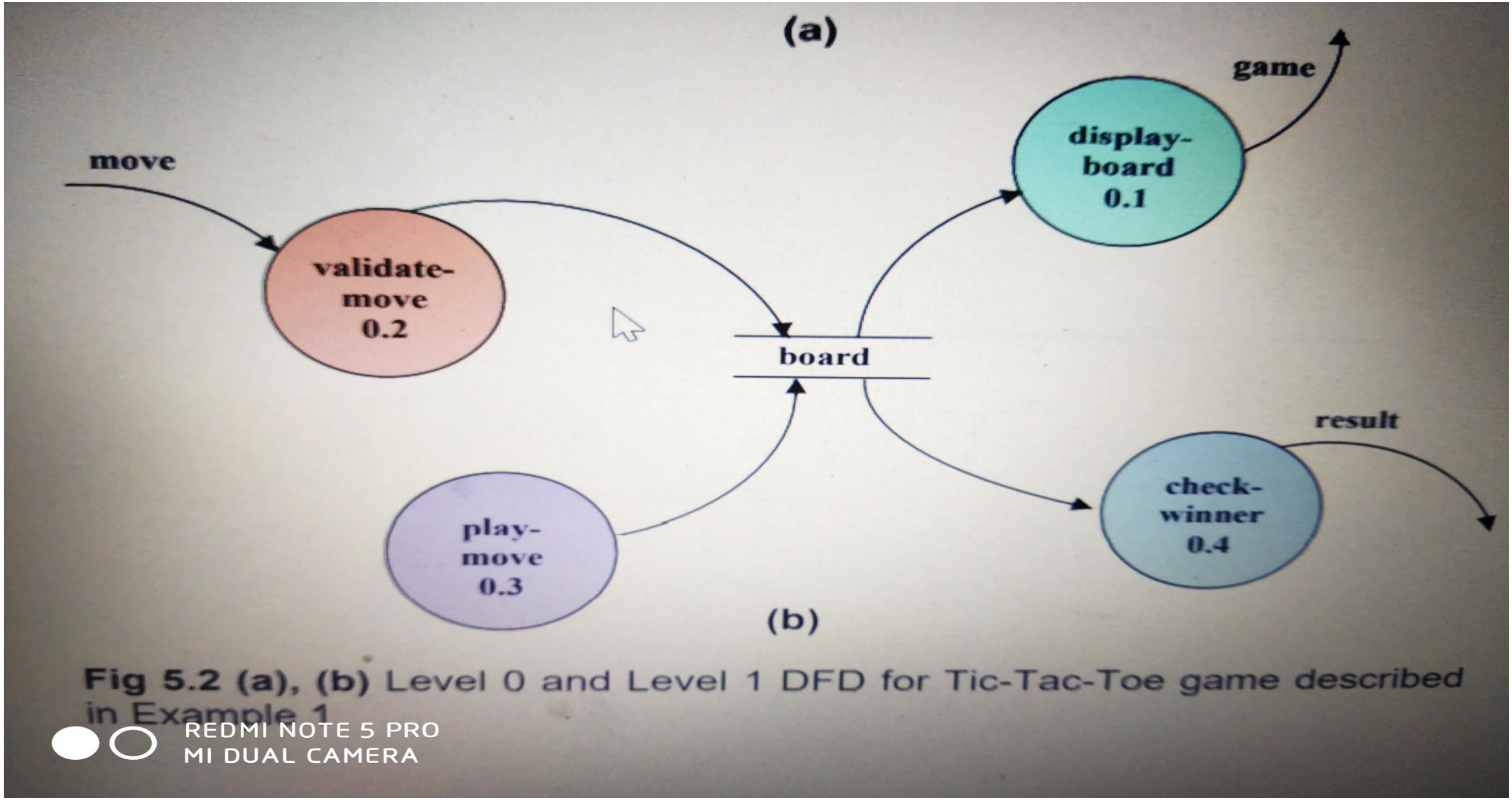


Fig 5.2 (a), (b) Level 0 and Level 1 DFD for Tic-Tac-Toe game described in Example 1

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-tactoe game. The data dictionary for the model is given below. Data dictionary for the DFD model in Example 1

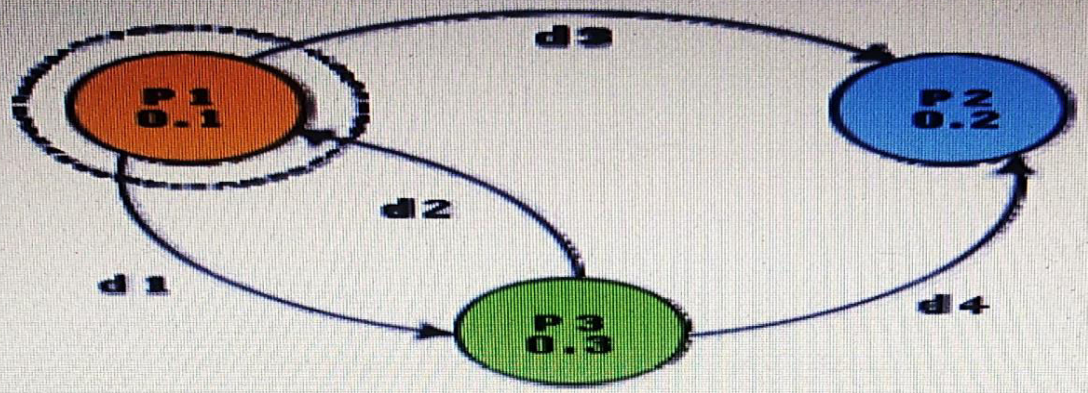
move:	integer /*number between 1 and 9 */
display:	game+result
game:	board
board:	{integer}9
result:	[“computer won”, “human won” “draw”]

IMPORTANCE OF DATA DICTIONARY

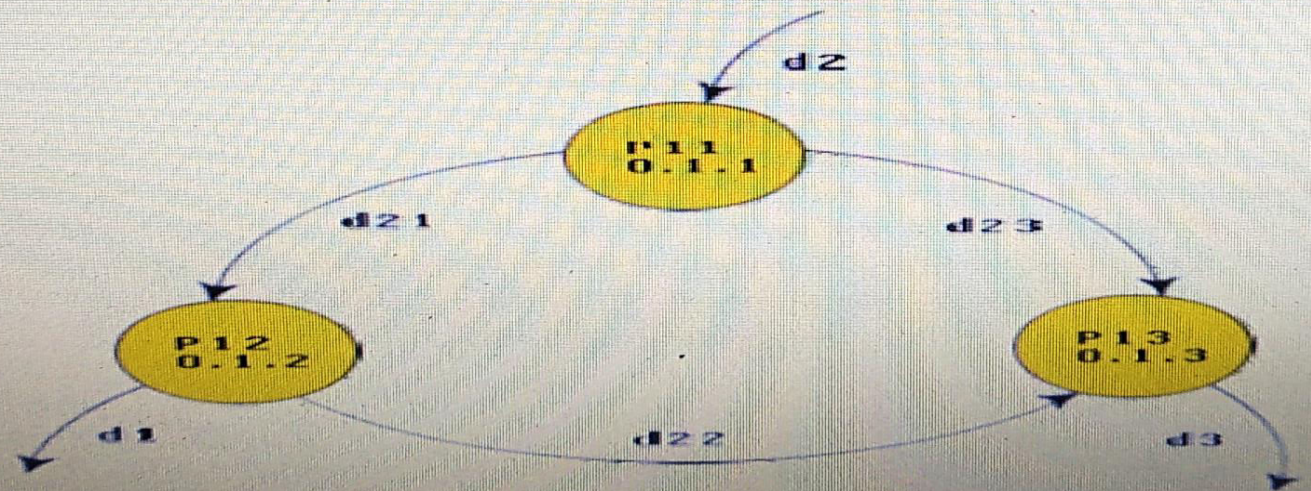
A data dictionary plays a very important role in any software development process because of the following reasons: • A data dictionary provides a standard terminology for all relevant data for use by the engineers working in a project. A consistent vocabulary for data items is very important, since 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

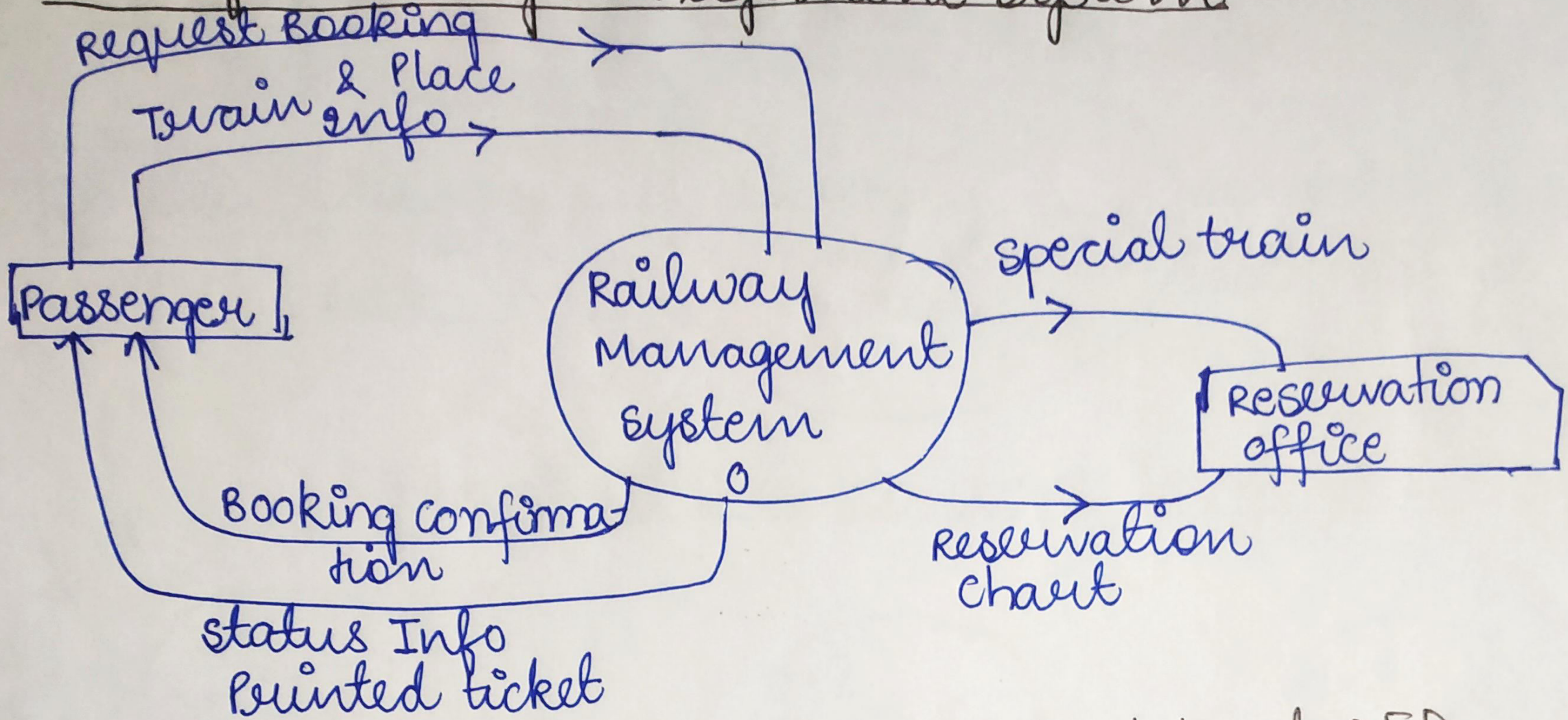
CONTEXT DIAGRAM

The context diagram is the most abstract data flow representation of a system. It represents the entire system as a single bubble. This bubble is labeled according to the main function of the system. The various external entities with which the system interacts and the data flow occurring between the system and the external entities are also represented. The data input to the system and the data output from the system are represented as incoming and outgoing arrows. These data flow arrows should be annotated with the corresponding data names. The name 'context diagram' is well justified because it represents the context in which the system is to exist, i.e. the external entities who would interact with the system and the specific data items they would be supplying the system and the data items they would be receiving from the system. The context diagram is also called as the level 0 DFD.

To develop the context diagram of the system, it is required to analyze the SRS document to identify the different types of users who would be using the system and the kinds of data they would be inputting to the system and the data they would be receiving the system. Here, the term "users of the system" also includes the external systems which supply data to or receive data from the system.

The bubble in the context diagram is annotated with the name of the software system being developed (usually a noun). This is in contrast with the bubbles in all other levels which are annotated with verbs. This is expected since the purpose of the context diagram is to capture the context of the system rather than its functionality

DFD of Railway Management System-



0 Level DFD or context level DFD

RMS Calculating Software. A software system called RMS calculating software would read three integral numbers from the user in the range of -1000 and +1000 and then determine the root mean square (rms) of the three input numbers and display it. In this

example, the context diagram (fig. 5.4) is simple to draw. The system accepts three integers from the user and returns the result to him.

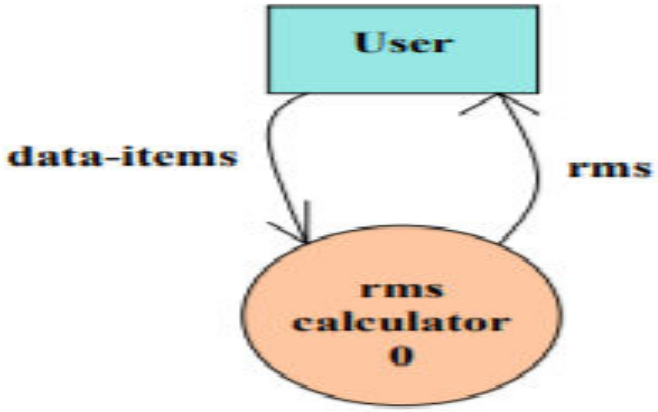


Fig. 5.4: Context Diagram

Example#2: Tic-Tac-Toe Computer Game

The problem is described in Lesson 5.1(Example 1). The level 0 DFD shown in Figure 5.2(a) is the context diagram for this problem.

DFD model of a system

A DFD model of a system graphically depicts the transformation of the data input to the system to the final result through a hierarchy of levels. A DFD starts with the most abstract definition of the system (lowest level) and at each higher level DFD, more details are successively introduced. To develop a higher-level DFD model, processes are decomposed into their sub-processes and the data flow among these sub-processes is identified.

Level 1 DFD:- To develop the level 1 DFD, examine the high-level functional requirements. If there are between 3 to 7 high-level functional requirements, then these can be directly represented as bubbles in the level 1 DFD. We can then examine the

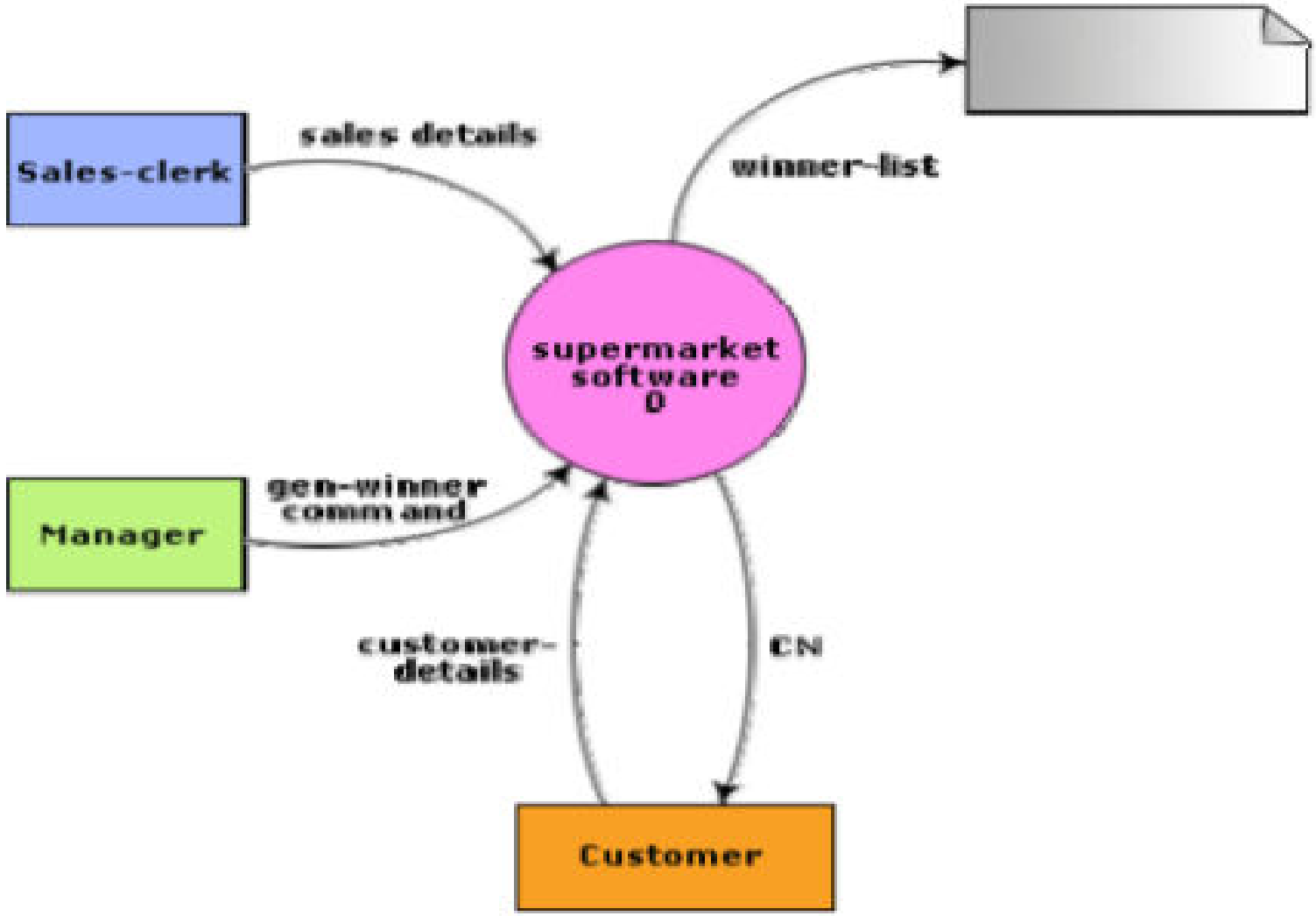
input data to these functions and the data output by these functions and represent them appropriately in the diagram. If a system has more than 7 high-level functional requirements, then some of the related requirements have to be combined and represented in the form of a bubble in the level 1 DFD. Such a bubble can be split in the lower DFD levels. If a system has less than three high-level functional requirements, then some of them need to be split into their sub-functions so that we have roughly about 5 to 7 bubbles on the diagram.

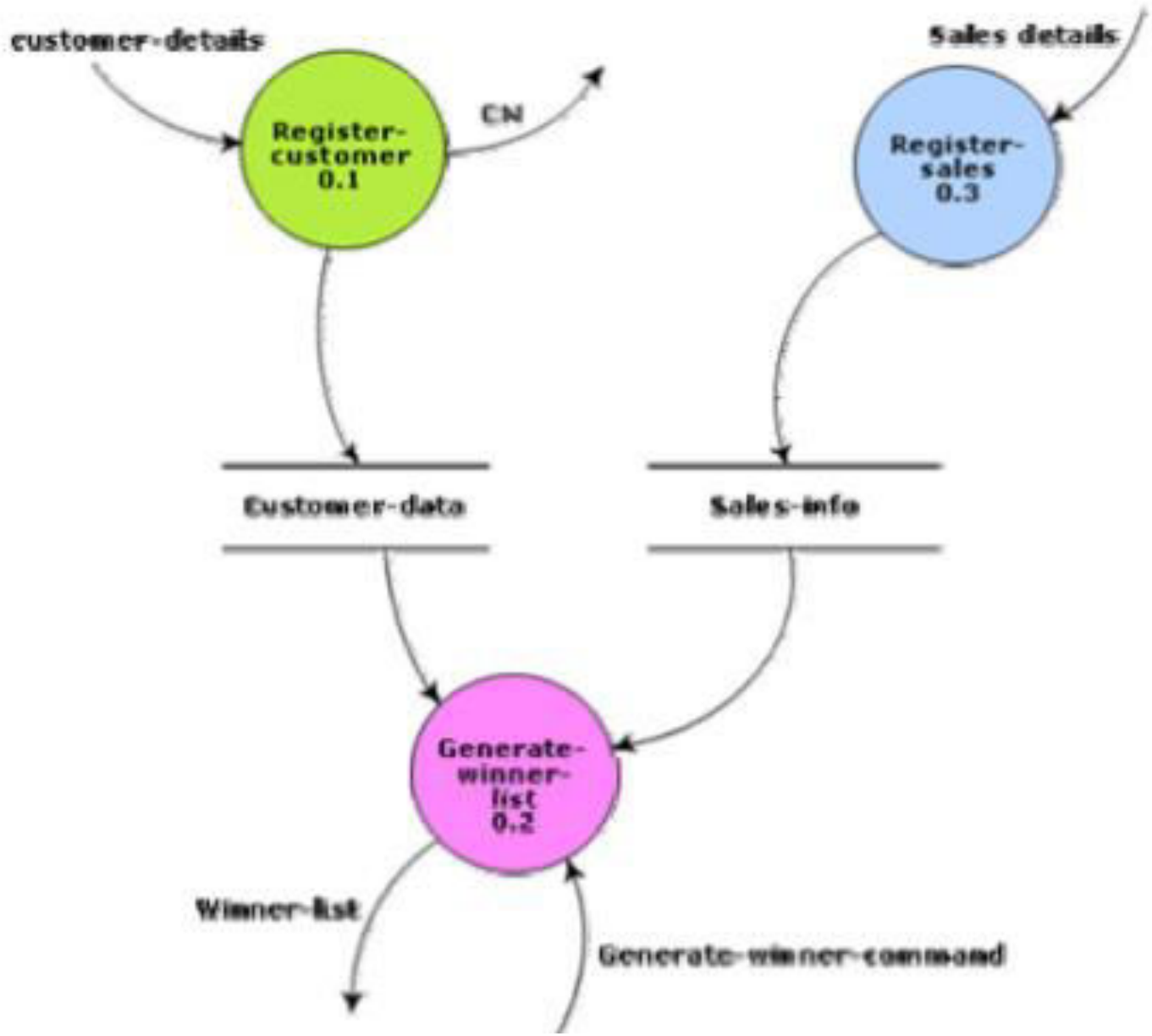
Decomposition:- Each bubble in the DFD represents a function performed by the system. The bubbles are decomposed into sub-functions at the successive levels of the DFD. Decomposition of a bubble is also known as factoring or exploding a bubble. Each bubble at any level of DFD is usually decomposed to anything between 3 to 7 bubbles. Too few bubbles at any level make that level superfluous. For example, if a bubble is decomposed to just one bubble or two bubbles, then this decomposition becomes redundant. Also, too many bubbles, i.e. more than 7 bubbles at any level of a DFD makes the DFD model hard to understand. Decomposition of a bubble should be carried on until a level is reached at which the function of the bubble can be described using a simple algorithm.

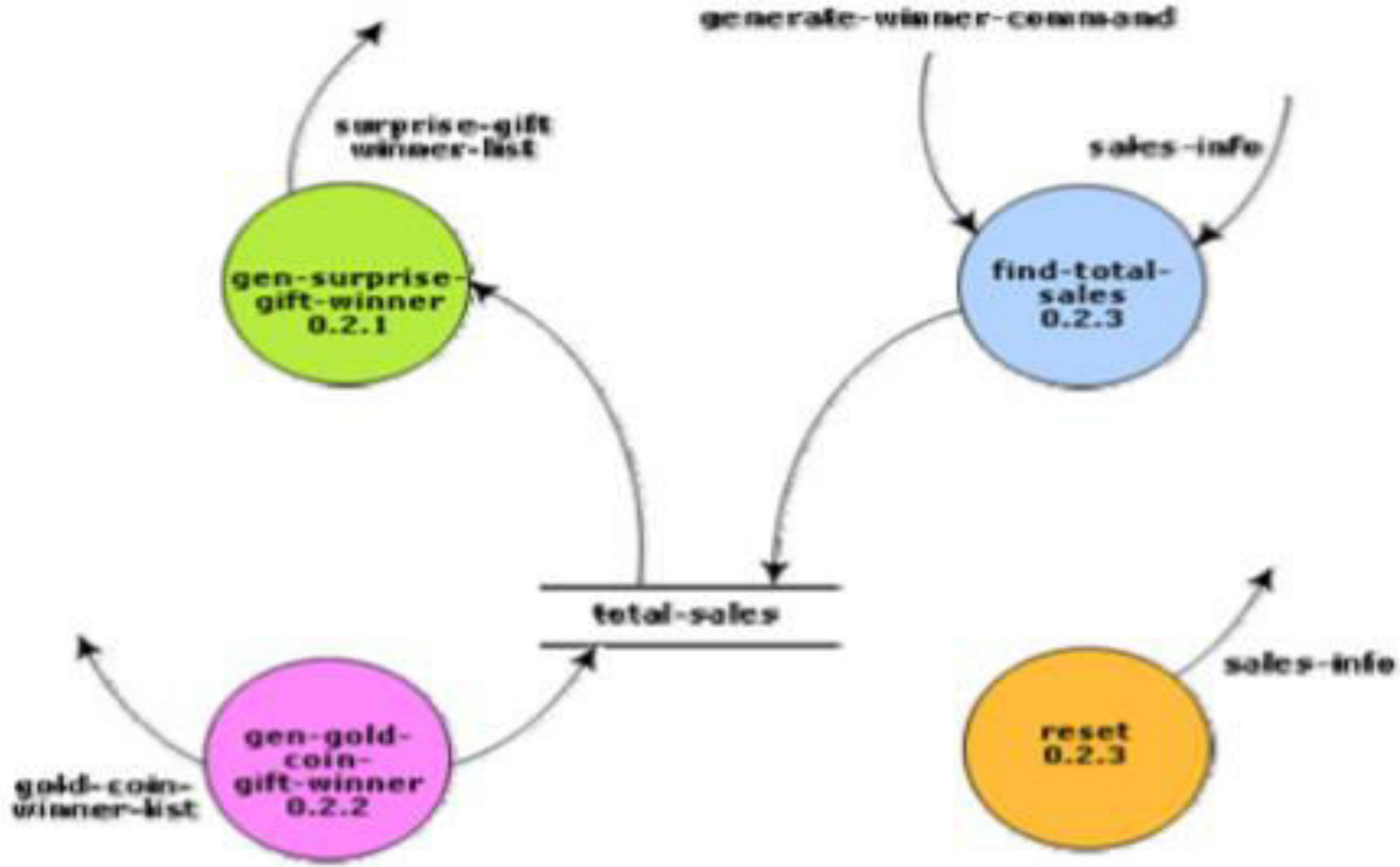
Numbering of Bubbles:- It is necessary to number the different bubbles occurring in the DFD. These numbers help in uniquely identifying any bubble in the DFD by its bubble number. The bubble at the context level is usually assigned the number 0 to indicate that it is the 0 level DFD. Bubbles at level 1 are numbered, 0.1, 0.2, 0.3, etc, etc. When a bubble numbered x is decomposed, its children bubble are numbered x.1, x.2, x.3, etc. In this numbering scheme, by looking at the number of a bubble we can unambiguously determine its level, its ancestors, and its successors.

Example:- A supermarket needs to develop the following software to encourage regular customers. For this, the customer needs to supply his/her residence address, telephone number, and the driving license number. Each customer who registers for this scheme is assigned a unique customer number (CN) by the computer. A customer can present his CN to the check out staff when he makes any purchase. In this case, the value of his purchase is credited against his CN. At the end of each year, the supermarket intends to award surprise gifts to 10 customers who make the highest total purchase over the year. Also, it intends to award a 22 caret gold coin to every customer whose purchase exceeded Rs.10,000. The entries against the CN are the reset on the day of every year after the prize winners' lists are generated.

The context diagram for this problem is shown in fig. 5.5, the level 1 DFD in fig. 5.6, and the level 2 DFD in fig. 5.7.




customer-details
Register-customer 0.1
CN
Sales details
Register-sales 0.3
Customer-data
Sales-info
Generate-winner-list 0.2
Winner-list
Generate-winner-command


generate-winner-command
surprise-gift winner-list
sales-info
gen-surprise-gift-winner 0.2.1
find-total-sales 0.2.3
total-sales
gold-coin-winner-list
gen-gold-coin-gift-winner 0.2.2
reset 0.2.3
sales-info

Commonly made errors while constructing a DFD model Although DFDs are simple to understand and draw, students and practitioners alike encounter similar types of problems while modelling software problems using DFDs. While learning from experience is powerful thing, it is an expensive pedagogical technique in the business world. It is therefore helpful to understand the different types of mistakes that users usually make while constructing the DFD model of systems

- **Many beginners commit the mistake of drawing more than one bubble in the context diagram. A context diagram should depict the system as a single bubble**
- **Many beginners have external entities appearing at all levels of DFDs. All external entities interacting with the system should be represented only in the context diagram. The external entities should not appear at other levels of the DFD**
- **It is a common oversight to have either too less or too many bubbles in a DFD. Only 3 to 7 bubbles per diagram should be allowed, i.e. each bubble should be decomposed to between 3 and 7 bubbles**

- **Many beginners leave different levels of DFD unbalanced.**
- **A common mistake committed by many beginners while developing a DFD model is attempting to represent control information in a DFD. It is important to realize that a DFD is the data flow representation of a system, and it does not represent control information. For an example mistake of this kind:**

- Consider the following example. A book can be searched in the library catalog by inputting its name. If the book is available in the library, then the details of the book are displayed. If the book is not listed in the catalog, then an error message is generated. While generating the DFD model for this simple problem, many beginners commit the mistake of drawing an arrow (as shown in fig. 5.10) to indicate the error function is invoked after the search book. But, this is a control information and should not be shown on the DFD.

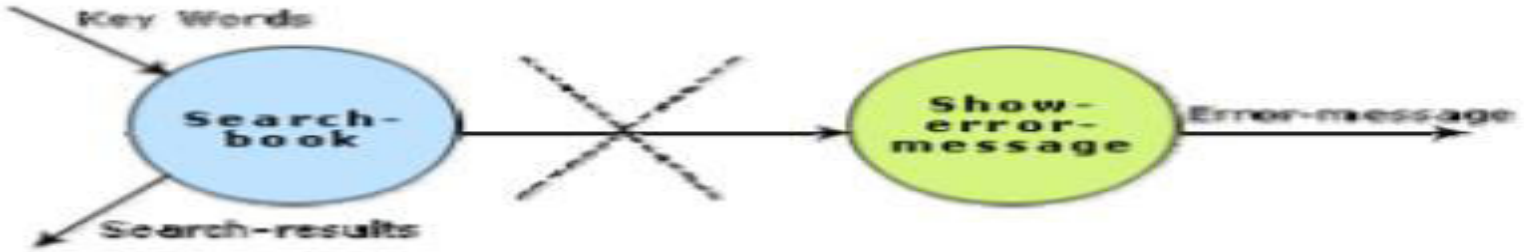


Fig. 5.10: Showing control information on a DFD - incorrect

- Another error is trying to represent when or in what order different functions (processes) are invoked and not representing the conditions under which different functions are invoked.
- If a bubble A invokes either the bubble B or the bubble C depending upon some conditions, we need only to represent the

data that flows between bubbles A and B or bubbles A and C and not the conditions depending on which the two modules are invoked.

- **A data store should be connected only to bubbles through data arrows. A data store cannot be connected to another data store or to an external entity.**
- **All the functionalities of the system must be captured by the DFD model. No function of the system specified in its SRS document should be overlooked.**
- **Only those functions of the system specified in the SRS document should be represented, i.e. the designer should not assume functionality of the system not specified by the SRS document and then try to represent them in the DFD.**
- **Improper or unsatisfactory data dictionary.**
- **The data and function names must be intuitive. Some students and even practicing engineers use symbolic data names such a, b, c, etc. Such names hinder understanding the DFD model.**

Shortcomings of a DFD model

DFD models suffer from several shortcomings. The important shortcomings of the DFD models are the following:

- **DFDs leave ample scope to be imprecise. In the DFD model, the function performed by a bubble is judged from its label. However, a short label may not capture the entire functionality of a bubble. For example, a bubble named find-book-position has only intuitive meaning and does not specify several things, e.g. what happens when some input information are missing or are incorrect. Further, the find-bookposition bubble may not convey anything regarding what happens when the required book is missing.**

- **Control aspects are not defined by a DFD. For instance, the order in which inputs are consumed and outputs are produced by a bubble is not specified. A DFD model does not specify the order in which the different bubbles are executed. Representation of such aspects is very important for modeling real-time systems.**

.

- **The method of carrying out decomposition to arrive at the successive levels and the ultimate level to which decomposition is carried out are highly subjective and depend on the choice and judgment of the analyst. Due to this reason, even for the same problem, several alternative DFD representations are possible. Further, many times it is not possible to say which DFD representation is superior or preferable to another one**

- **The data flow diagramming technique does not provide any specific guidance as to how exactly to decompose a given function into its subfunctions and we have to use subjective judgment to carry out decomposition**.