4 Data Flow Diagram

Abstract Data Flow Diagram (DFD) is widely used for structured software analysis and design. It is also widespread in the field of business administration. The syntax and semantics of DFD are introduced in this chapter. A structured approach for DFD model development is also discussed.

4.1 Introduction to DFD

The Data Flow Diagram (DFD) is a structured analysis and design method. It is a visual tool to depict logic models and expresses data transformation in a system. DFD includes a mechanism to model the data flow. It supports decomposition to illustrate details of the data flows and functions. DFD cannot present information on operation sequence. Therefore, it is not a process or procedure modeling method.

DFD includes following characteristics: (1) supporting the analysis and requirement stage of system design; (2) a diagramming technique with annotation; (3) describing a network of activities/processes of the target system; (4) allowing for behaviors of parallel and asynchronous ; (5) stepwise refinement through hierarchical decomposition of processes.

4.2 Syntax and Semantics of DFD

The complete data flow analysis includes: Data Flow Diagram, Data Dictionary and Process Specifications [3].

DFD presents a symbol system to describe data flows and a decomposition mechanism to describe a system at various detail levels.

4.2.1 Notations of DFD

The construction elements of DFD are Activity/Process, Data Flow, Data Store and External Entity (Source/Sink).

(1) Activity/process

The notation of activity/process is shown in Fig. 4.1. These two symbols belong to Gane & Sarson notation system and Ward & Mellor notation system, respectively.

- An activity /process is the transformation of data. It accepts data flows as inputs and produces data flows as outputs.
- An activity can be further decomposed to form more detailed sub-process.
- The label of an activity/process should be a verb.
- Activities are linked to process specifications.

The rules for activity/process are listed as follows.

- An activity/process is always internal to a system. How the external entity or system treats data will not be modeled.
- Data stays at rest unless moved by a process.
- Processes cannot consume or create data. That means the process must have at least 1 input data flow (to avoid miracles), at least 1 output data flow (to avoid black holes) and should have sufficient inputs to create outputs (to avoid gray holes).

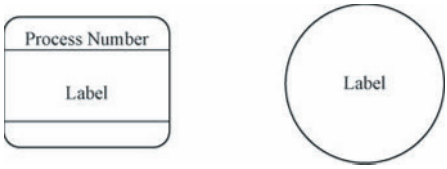

Fig. 4.1. Symbols of activity/process

Logical process models omit any processes that do nothing more than move or route data, thus leaving the data unchanged. Valid processes include those that:

- Perform computations (e.g., calculate grade point average).
- Make decisions (determine availability of ordered products).
- Sort, filter or otherwise summarize data (identify overdue invoices).
- Organize data into useful information (e.g., generate a report or answer a question).
- Trigger other processes (e.g., turn on the furnace or instruct a robot).
- Use stored data (create, read, update or delete a record).

(2) Data flow

A Data flow shows the flow of information. Its symbols are shown in Fig. 4.2.

- A Data flow is a connector element whose two ends link to activities/processes, a Data store, an external entity and so forth.
- It reflects a data transfer but control flows.
- The arrow of data flow shows its direction.
- Data flows can be split / joined.
- The label of a data flow is noun.
- It is specified in the data-dictionary.

The rules for data flow are listed as follows.

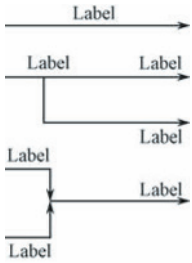

Fig. 4.2. Symbols of data flow

- A data flow means data in motion, moving from one place to another in the system.
- It flows from an external entity (source) to the system.
- It flows from the system to an external entity (sink).
- It flows from an internal symbol to another internal symbol, but always either start or end at a process.

(3) Data store

A data store is the storage for permanent data and presents a placeholder for database/file. The symbols of data store are shown in Fig. 4.3.

Fig. 4.3. Symbols of data store

- Data store is passive.
- It is serviced by a process.
- No activity can be beyond basic retrieval capacity.
- Its label should be a noun.
- It is specified in the data-dictionary and/or with an ERD.

The rules for data store are listed as follows.

- It is internal to the system.
- Data in it keep at rest. That means the data store does not change the status of data by itself.
- It should be included in the system if the system processes transform (store, add, delete, update) the data.
- Every data store on DFD should correspond to an entity on an ERD.
- Data stores can come in many forms such as hanging file folders, computer-based files, note books, and so forth.

(4) External entity (source/sink)

An External Entity provides connection to the system's context. Its symbols are shown in Fig. 4.4.

- It is the origin/destination of external data flows.
- It provides connection to the system's context.
- It is passive and only sends/receives data.
- Its label should be a noun.

- It is specified in the data-dictionary.

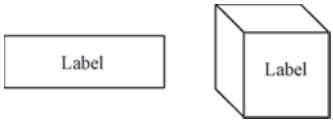

Fig. 4.4. Symbols of External Entity

The rules for External Entity are listed as follows.

- External Entities are external people, systems and data stores.
- They stand outside the system, but interact with the system.
- They (1) receive information from the system, (2) trigger the system into motion, or (3) provide new information to the system.

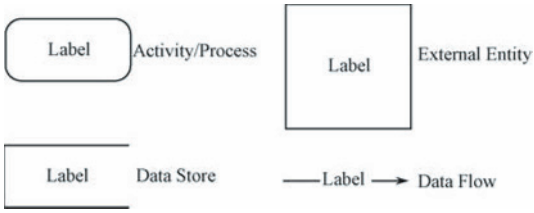

Fig. 4.5. Gane & Sarson DFD notations

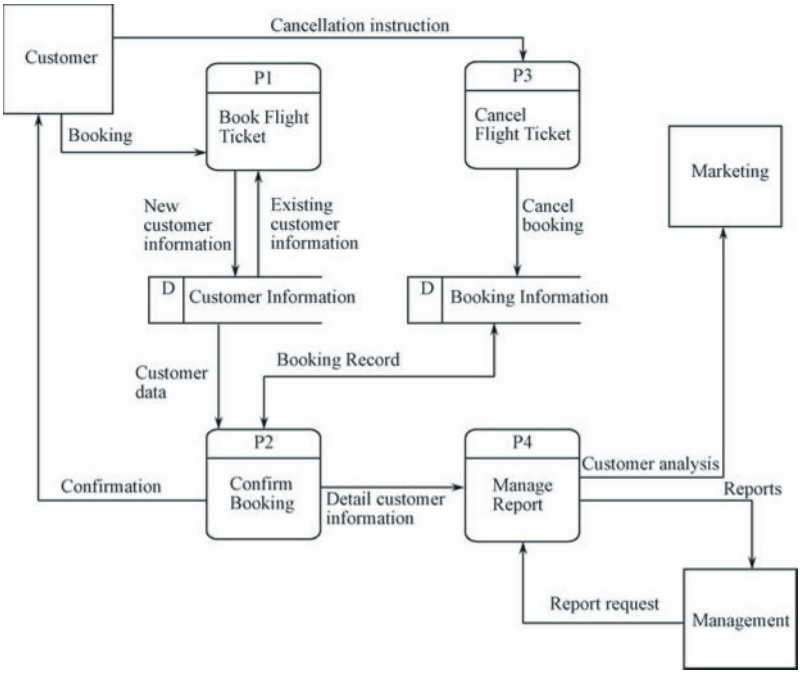

Fig. 4.6. DFD example of flight ticket booking

Gane & Sarson DFD notations are shown in Fig. 4.5 and an example on a flight ticket booking system based on the notation is shown in Fig. 4.6. The

extended notations by Ward and Mellor are shown in Fig. 4.7 and an office environment control system model is shown in Fig. 4.8 and Fig. 4.9.

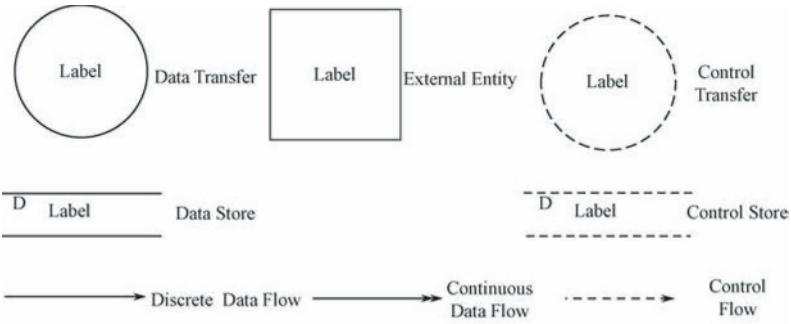

Fig. 4.7. DFD notations extended by Ward and Mellor

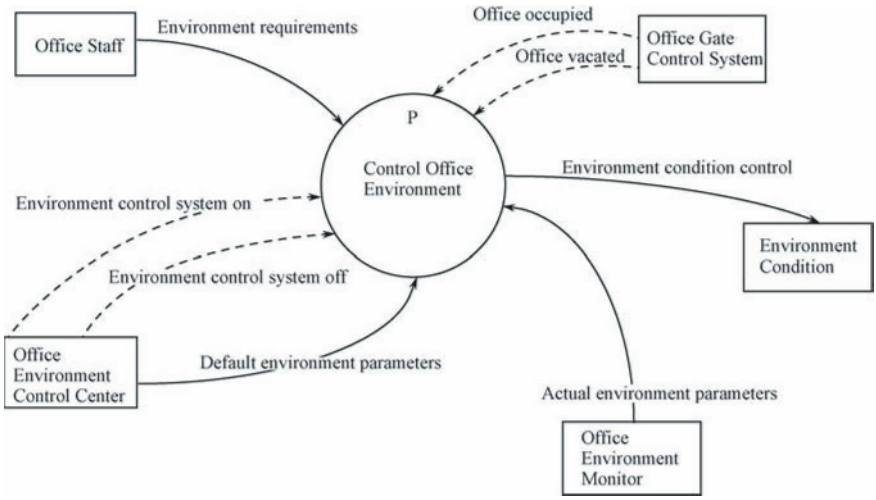

Fig. 4.8. DFD example of an office environment control system

4.2.2 DFD Models Organization

Activities/Processes in DFD can be decomposed. That means certain activities in the parent diagram can be illustrated by more detailed child diagrams. A set of DFDs includes a context diagram, a level-0 diagram and relative child diagrams.

(1) Context diagram

The context diagram is the DFD of the scope of an organizational system that shows the system boundaries, external entities that interact with the system and the major information flows between the external entities and

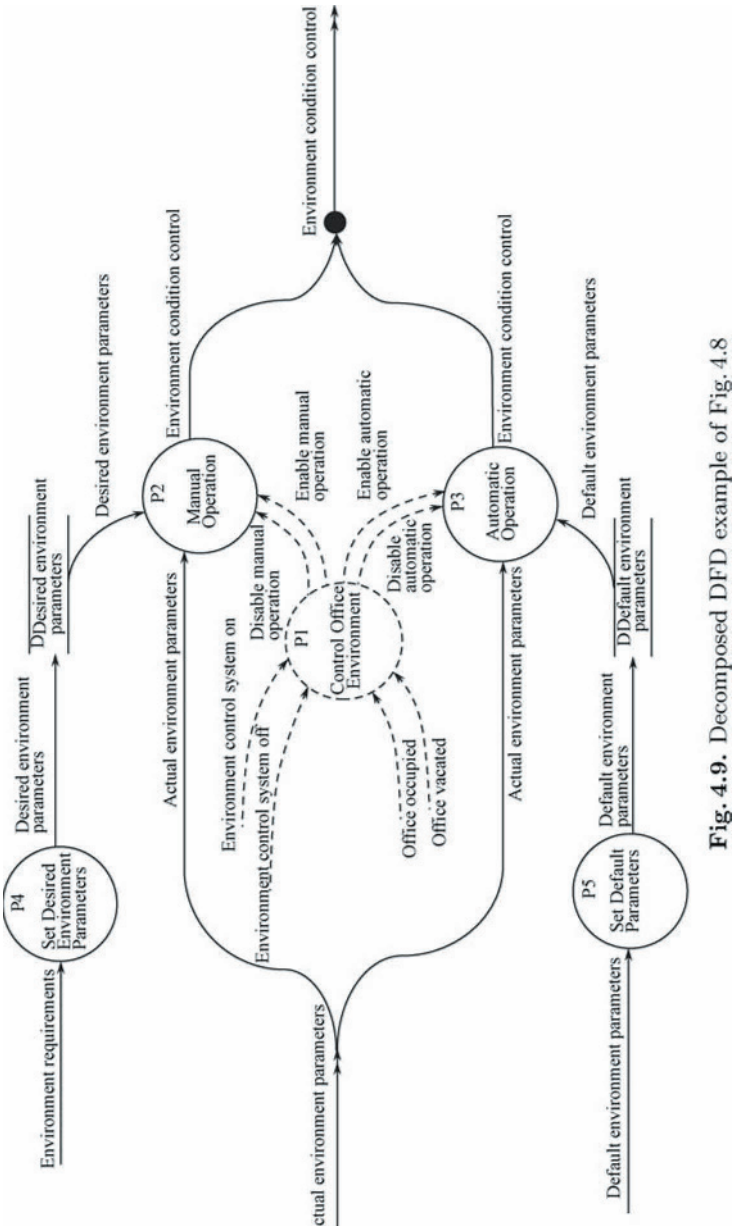

Fig. 4.9. Decomposed DFD example of Fig. 4.8

the system. Fig. 4.10 shows the context diagram of a food ordering system. Fig. 4.11 shows the context diagram of a salary management system.

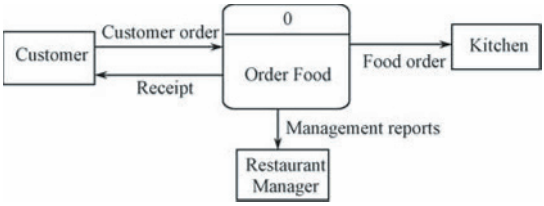

Fig. 4.10. Context diagram of a food ordering system

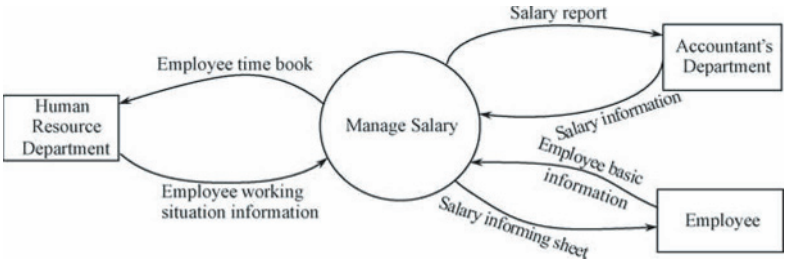

Fig. 4.11. Context diagram of a salary management system

(2) Level-0 diagram

The level-0 diagram is a DFD that represents a system's major processes, data flows and data stores at a high level of detail. Fig. 4.12 shows the level-0 diagram of a food ordering system. Fig. 4.13 shows the level-0 diagram of a salary management system.

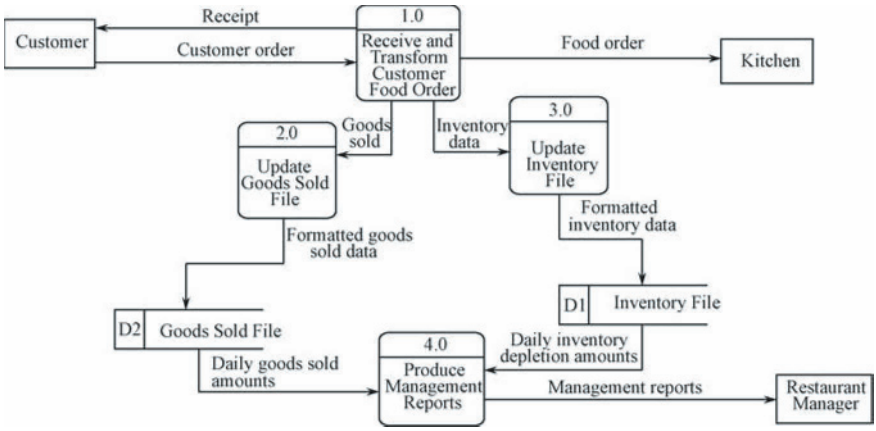

Fig. 4.12. Level-0 diagram of the food ordering system

(3) Decomposed DFD child diagrams

Going on decomposition step by step, level-1 diagrams, level-2 diagrams and so forth are developed. Fig. 4.14 shows the level-1 diagram of the salary

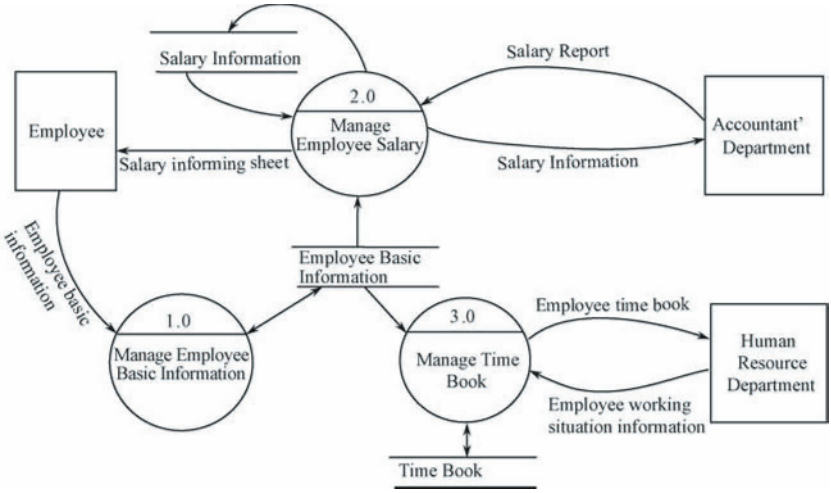

Fig. 4.13. Level-0 diagram of the salary management system

management system. It is the decomposition of Activity 3.0.

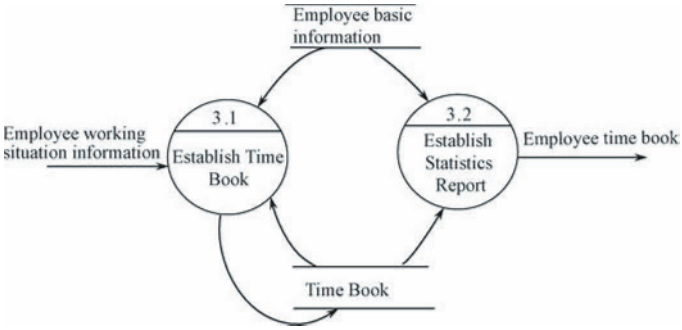

Fig. 4.14. Level-1 diagram of the salary management system

Through decomposition, a tree style models framework is formed. The framework and relative coding rules are shown in Fig. 4.15.

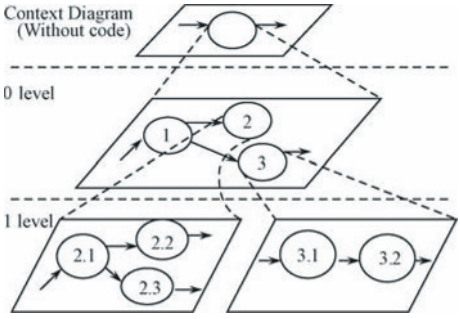

Fig. 4.15. DFD decomposition framework and relative codes

4.2.3 Data Dictionary

The finished DFD should be accompanied with a data dictionary.

A data dictionary includes:

(1) Name: the title of data item, control item, data store or external entity.

(2) Alias: alternative name.

(3) Usage: where and how to use.

(4) Content depiction: symbols system for content depiction.

(5) Additional information: data type, default value, constraint, etc.

4.3 Structured Approach of DFD

DFD is the core technique of Structured Design and Analysis Method and widely used in industries. It includes a formalized procedure and rules to models development.

4.3.1 Modeling Process of DFD

Creating a DFD is a highly iterative process of gradual refinement. The general steps are:

(1) Create a preliminary Context Diagram.

(2) Identify Use Cases, i.e., the ways in which users most commonly use the system.

(3) Create DFD fragments for each use case.

(4) Create a level-0 diagram from fragments.

(5) Decompose to level 1, 2, ...

(6) Go to step (1) and revise as necessary.

(7) Validate the DFDs with users.

4.3.2 Data Flow Diagramming Rules

While developing DFDs, some rules should be obeyed. These rules include general rules and specific rules for DFD symbols, context diagram and decomposition.

(1) General rules

- Inputs to a process are always different than outputs.
- Objects always have a unique name.
- In order to keep the diagram uncluttered, you can repeat data stores and sources/sinks on a diagram.

(2) Rules for activity/process

- No process can have only outputs (a miracle).
- No process can have only inputs (black hole).
- A process has a verb phrase label.

(3) Rules for data store

- Data cannot be moved directly from one store to another.
- Data cannot move directly from an outside source to a data store.
- Data cannot move directly from a data store to a data sink.
- Data store has a noun phrase label.

(4) Rules for external entity

- Data cannot move directly from a source to a sink.
- A source/sink has a noun phrase label.

(5) Rules for data flow

- A data flow has only one direction of flow between symbols.
- A fork means that exactly the same data goes from a common location to two or more processes, data stores or sources/sinks.
- A join means that exactly the same data comes from any two or more different processes, data stores or sources/sinks to a common location.
- A data flow cannot go directly back to the same process it leaves.
- A data flow to a data store means update.
- A data flow from a data store means retrieve or use.
- A data flow has a noun phrase label.

(6) Rules for context diagram

- One process, numbered 0.
- Sources and sinks (external entities) as squares.
- Main data flows depicted.
- No internal data stores are shown. They are inside the system.
- External data stores are shown as external entities.

(7) Rules for process decomposition

- Processes can be decomposed / refined. That means one process can be decomposed into a complete DFD.
- Interface flows must remain consistent. When decomposing a DFD, it is necessary to conserve inputs to and outputs from a process at the next level of decomposition. This is called balancing. As shown in Fig. 4.10, there are one input "Customer order" and three outputs "Receipt", "Food order", "Management reports". In its child diagram as shown in Fig. 4.12, there are the same input and outputs without any modification. Then it can be said that the context diagram in Fig. 4.10 and the level-0 diagram in Fig. 4.12 are balancing (consistent). The decomposition relationship between Fig. 4.16 and Fig. 4.17 is unbalancing. There are one input and one output in Fig. 4.16 while there adds an input "C" in Fig. 4.17.
- Lower level processes, data flows and data stores can be added on.
- Sources and sinks remain on level-1.
- The level-0 can be used as "abstract".
- A data flow can be split into separate data flows on a lower level diagram, as shown in Fig. 4.18.

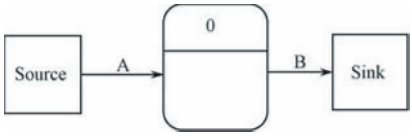

Fig. 4.16. An example of context diagram

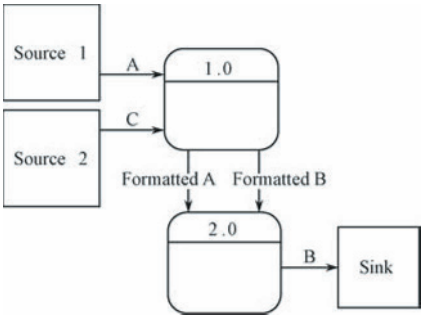

Fig. 4.17. An example of level-0 diagram

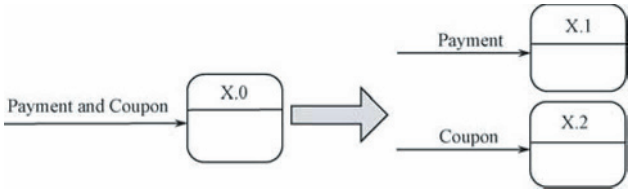

Fig. 4.18. Data flow split

4.4 DFD Modeling Case

A simplified examination management system is selected to illustrate the modeling and analysis process with DFD.

The functionality of the examination management system is declared as follows.

(1) Check applicants' information sheets.

(2) Prepare examinee identification number and send examinee identifications to eligible applicants; transfer the examinees name list to the examination paper appraising office.

(3) Check the examination result report from the examination paper appraising office and determine examinees that pass the examination under certain criterion.

(4) Inform examinees of their examination results.

(5) Analyze examination results and prepare an analysis report.

The context diagram is developed firstly to isolate the system from it environment and illustrate interactions between the system and its users, as shown in Fig. 4.19.

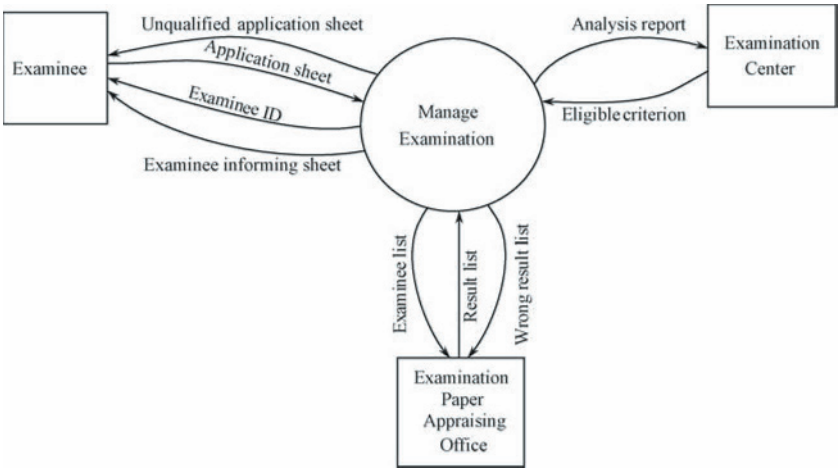

Fig. 4.19. Context diagram of an examination management system

The examination management system has two main functions: "Register Application Sheet" and "Analyze Result". The level-0 diagram is shown in Fig. 4.20.

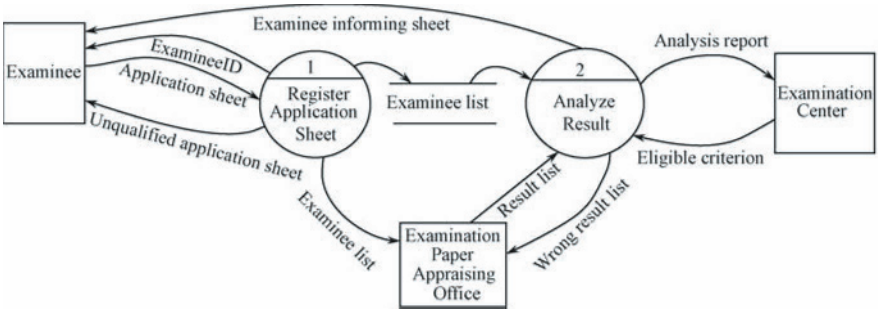

Fig. 4.20. Level-0 diagram of the system

Two activities in the level-0 diagram are decomposed to form two level-1 diagrams, as shown in Fig. 4.21 and Fig. 4.22.

DFD is widely used in industries. For instance, the international standard "ISO/IEC 62264-1 Enterprise – Control System Integration[1].

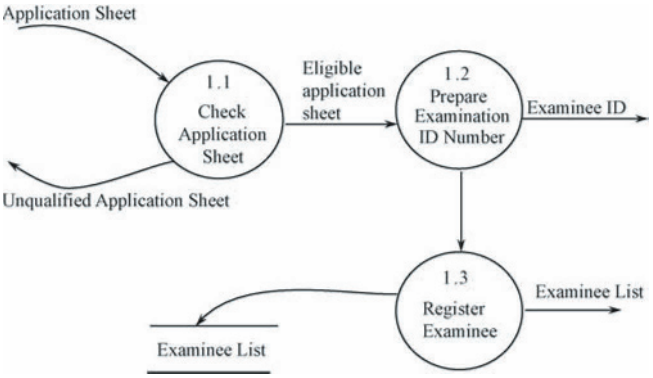

Fig. 4.21. Level-1 diagram: Decomposition of "Register Application Sheet"

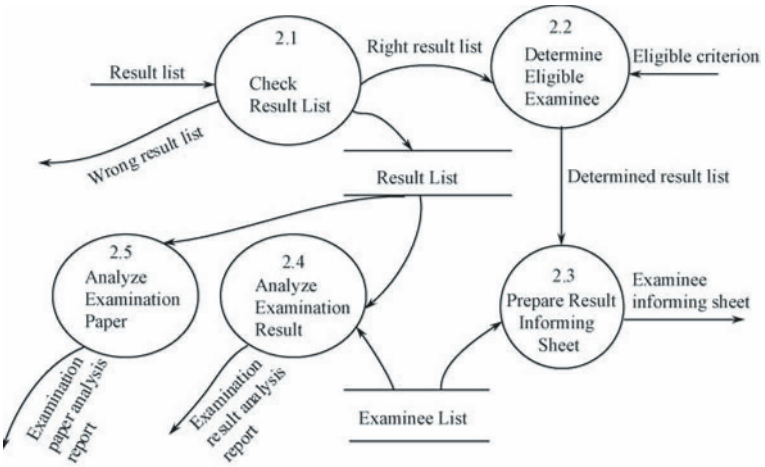

Fig. 4.22. Level-1 diagram: Decomposition of "Analyze Result"

References

[1] ISO TC184 SC5. ISO/IEC 62264-1: Enterprise-Control System Integration Part 1: Models and Terminology. ISO , 2003.

[2] Popkin Software: System Architect 2001 Tutorial. Popkin Software Co., 2001.

[3] Yourdon E. Just Enough Structured Analysis. http://www.yourdon.com/, 2006.