

UNIT-2

Modelling Tools For System Analyst

Goals:

- Analysts use various tools to understand and describe the information system. One of the ways is using structured analysis.
- **What is Structured Analysis?** Structured Analysis is a development method that allows the analyst to understand the system and its activities in a logical way. It is a systematic approach, which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification which can be easily understandable by user.
- **It has following attributes:**
 - It is graphic which specifies the presentation of application.
 - It divides the processes so that it gives a clear picture of system flow.
 - It is logical rather than physical i.e., the elements of system do not depend on vendor or hardware.
 - It is an approach that works from high-level overviews to lower-level details.

Role Of Data In Bussiness

Structured Analysis Tools During Structured Analysis, various tools and techniques are used for system development.

Modeling System Functions: ex. Data flow Diagram

Modeling Stored Data: ex. Entity Relation Diagram

Modeling Program Structure: ex. Program Flow Chart

Modeling Time: ex. Gantt Chart





Data Flow Diagram

- A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its *process* aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).
- A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel
- Data Flow Diagrams (DFD) or Bubble Chart It is a technique developed by Larry Constantine to express the requirements of system in a graphical form.
- It shows the flow of data between various functions of system and specifies how the current system is implemented.
- It is an initial stage of design phase that functionally divides the requirement specifications down to the lowest level of detail.
- Its graphical nature makes it a good communication tool between user and analyst or analyst and system designer.
- It gives an overview of what data a system processes, what transformations are performed, what data are stored, what results are produced and where they flow.

Basic Elements of DFD/ Components of DFD

DFD is easy to understand and quite effective when the required design is not clear and the user wants a notational language for communication. However, it requires a large number of iterations for obtaining the most accurate and complete solution.

The following table shows the symbols used in designing a DFD and their significance:

• Symbol Name	Symbol	Meaning
Square		Source or Destination of Data
Arrow		Data flow
Circle		Process transforming data flow
Open Rectangle		Data Store

External Entity

- A source or destination of data which is external to the system. E.g supplier, customer etc.

Data process

- A data process transforms data values. Here flow of data is transformed. E.g. Verify credits, updates inventory file.
- You can make a distinction between the following types of processes:

Data store

- A data store stores data passively for later access. A data store responds to requests to store and access data. It does not generate any operations. A data store allows values to be accessed in an order different from the order in which they were generated.
- Input flows indicate information or operations that modify the stored data such as adding or deleting elements or changing values. Output flows indicate information retrieved from the store; this information can be an entire value or a component of a value.

Data flow

- A data flow moves data between processes or between processes and data stores. As such, it represents a data value at some point within a computation and an intermediate value within a computation if the flow is internal to the diagram. This value is not changed.
- The names of input and output flows can indicate their roles in the computation or the type of the value they move. Data names are preferably nouns. The name of a typical piece of data, the data aspect, is written alongside the arrow.

Physical and Logical DFDs :--

- The DFDs that show “What is going on” instead of “How it is going on” is a logical DFD. DFDs that show how things happen and which are the actual physical components involved are known as physical DFDs. Logical DFDs help to get a clear idea of what the system has to achieve without getting into details like who is going to do it? How one is going to do it? Etc. But physical models are easier to visualize. Hence analyst begins with physical DFD before converting it to logical DFD.



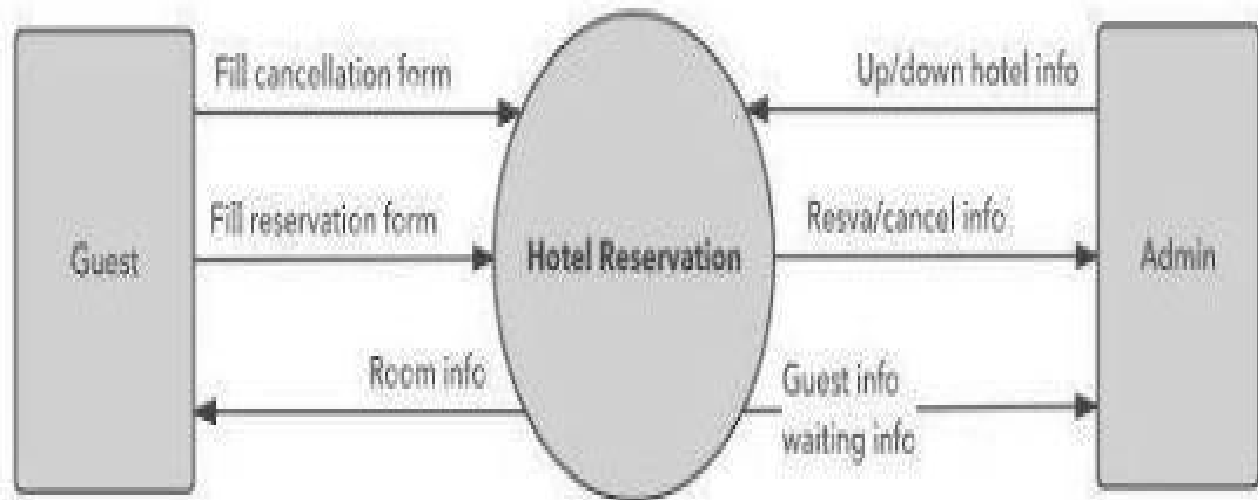
Physical DFD	Logical DFD
Data flow names include the implementation facts as names, numbers, media, timing etc.	Data flow names describe the data they contain. They do not refer to the form or document on which they reside.
Process names include the name of the processor i.e. person, department, computer system etc.	Process names describe the work done without referring to e.g. Account Receivable, Order processing etc.
Data Stores identify their computer and manual implementation.	Physical location of data stores is irrelevant. Many times, the same data store may be shared by subsystems and processes.
This is more realistic and implementation oriented. The PDFD are more detailed in nature.	As the name suggests, this is more logical in format. This is more abstract than PDFD and less dependent on implementation steps.

Levels OF DFD

✓ Context Diagram /0- level DFD

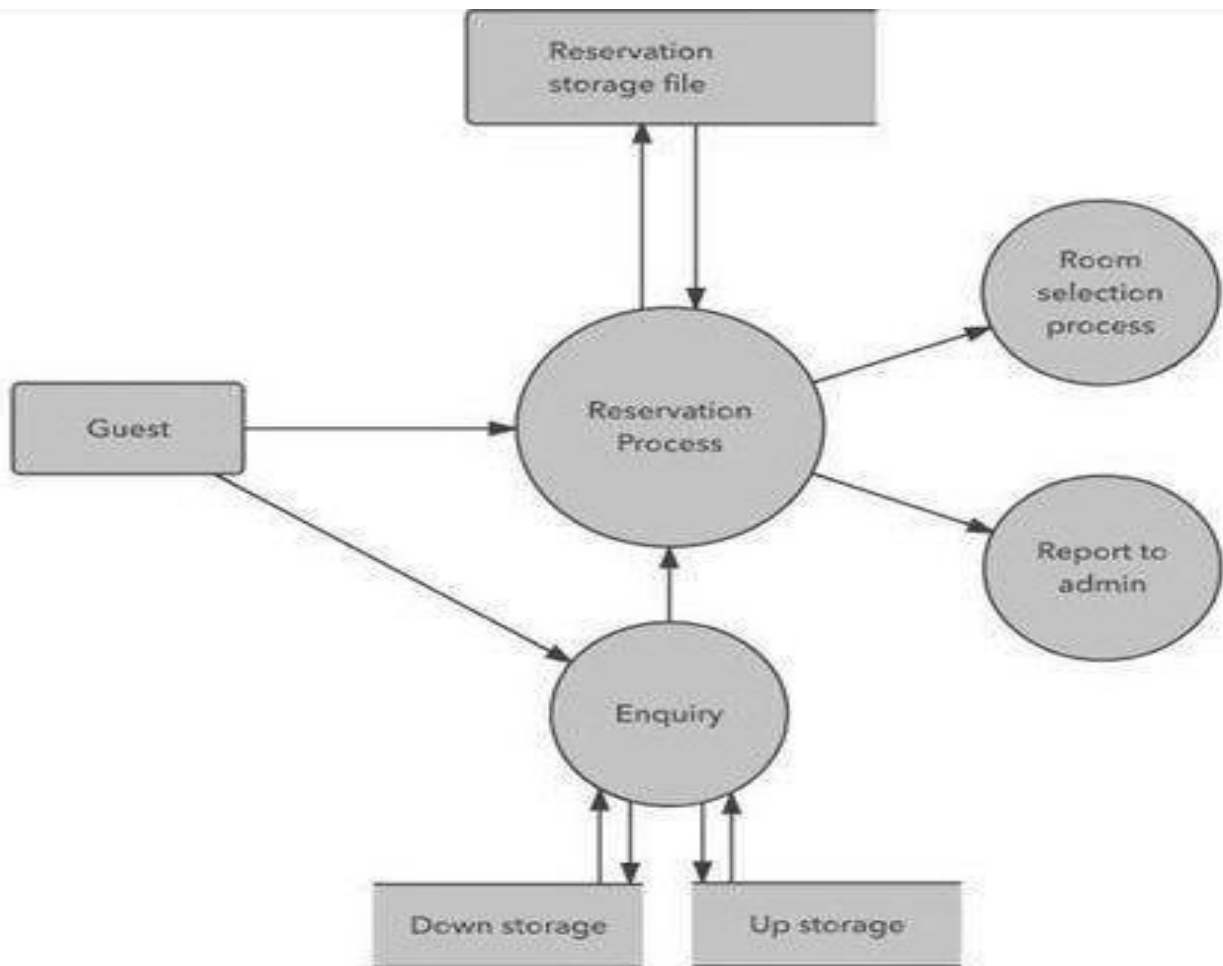
A context diagram helps in understanding the entire system by one DFD which gives the overview of a system. It starts with mentioning major processes with little details and then goes onto giving more details of the processes with the top-down approach.

The context diagram of Hotel Reservation is shown below. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.



Level 1 - The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information.

- It provides a more detailed view of the Context Level Diagram.
- Here, the main functions carried out by the system are highlighted as we break into its sub-processes.



SSADM: Structured Systems Analysis & Design Method

- SSADM (Structured Systems Analysis & Design Method) is a widely-used computer application development method in the UK, where its use is often specified as a requirement for government computing projects. It is increasingly being adopted by the public sector in Europe. SSADM is in the public domain, and is formally specified in British Standard BS7738.
- SSADM divides an application development project into modules, stages, steps, and tasks, and provides a framework for describing projects in a fashion suited to managing the project. SSADM's objectives are to:
 - ✓ Improve project management & control

- ✓ Make more effective use of experienced and inexperienced development staff
- ✓ Develop better quality systems
- ✓ Make projects resilient to the loss of staff
- ✓ Enable projects to be supported by computer-based tools such as computer-aided software engineering systems
- ✓ Establish a framework for good communications between participants in a project

SSADM uses a combination of three techniques:

- **Logical Data Modeling** -- the process of identifying, modeling and documenting the data requirements of the system being designed. The data is separated into *entities* (things about which a business needs to record information) and *relationships* (the associations between the entities).
- **Data Flow Modeling** -- the process of identifying, modeling and documenting how data moves around an information system. Data Flow Modeling examines *processes* (activities that transform data from one form to another), *data stores* (the holding areas for data), *external entities* (what sends data into a system or receives data from a system, and *data flows* (routes by which data can flow).
- **Entity Behavior Modeling** -- the process of identifying, modeling and documenting the events that affect each entity and the sequence in which these events occur.

SSADM's steps, or stages

1. **Feasibility Study**-- the business area is analyzed to determine whether a system can cost effectively support the business requirements.
2. **Requirements Analysis**-- the requirements of the system to be developed are identified and the current business environment is modeled in terms of the processes carried out and the data structures involved.
3. **Requirements Specification**-- detailed functional and non-functional requirements are identified and new techniques are introduced to define the required processing and data structures.

4. **Logical System Specification**-- technical systems options are produced and the logical design of update and enquiry processing and system dialogues.
5. **Physical Design** -- a physical database design and a set of program specifications are created using the logical system specification and technical system specification.

SSADM is supported by a number of CASE tool providers.

CASE TECHNOLOGY

- **Computer aided software engineering (CASE)** is the implementation of computer facilitated tools and methods in software development.
- CASE is used to ensure a high-quality and defect-free software. CASE ensures a check-pointed and disciplined approach and helps designers, developers, testers, managers and others to see the project milestones during development.
- CASE can also help as a warehouse for documents related to projects, like business plans, requirements and design specifications.
- One of the major advantages of using CASE is the delivery of the final product, which is more likely to meet real-world requirements as it ensures that customers remain part of the process.
- CASE illustrates a wide set of labor-saving tools that are used in software development.
- It generates a framework for organizing projects and to be helpful in enhancing productivity.

CASE TOOL

CASE refers alternatively to computer-aided system engineering or computer-aided software engineering. Objective of CASE tools is to automate key aspects of the entire system-development process, from beginning to end. CASE tools are essential for system analysis and it improve quality of the final result.

CASE Components: -

CASE tools generally include five Components:-

1. Diagramming Tools
2. Information Repository
3. Interface Generators
4. Code Generators
5. Management Tools

1. Diagramming Tools: - Diagramming tools **support analysis and documentation** of application requirements. They include the capabilities to **produce data flow diagrams**, data structure diagrams, and program structure charts. They support the **capability to draw diagrams and charts, and to store the details internally**. When changes must be made, the nature, of the change is described to the system, which can then redraw the entire diagram automatically.

2 Centralized information Repository: -

The capture, analysis, processing, and distribution of all systems information is aided by a centralized information repository or data dictionary. The dictionary contains the details of system components, such as data items, data flows, and processes.

3 Interface Generators :- System interfaces are the means by which users interact with an application, both to enter information and data or to receive information. Interface generators provide the capability to prepare demonstration of user interfaces. Like creation of system menus, presentations screens and report layouts.

4 Code Generators :- Code generators **automate the preparation of computer software**. They incorporate methods that allow the conversion of system specifications into executable source code. The best generators will **produce approximately 75 percent of the source code** for an application. The rest must be written by hand.

5 Management Tools: -

CASE systems also **help project managers in maintaining efficiency and effectiveness** throughout application development process. This CASE component assists development managers in the scheduling of analysis and design activities and the **allocation of resources to different project activities**.

Why Are CASE Tools Important?

Advantages

1. Increased Speed.

CASE Tools provide automation and reduce the time to complete many tasks, especially those involving diagramming and associated specifications. Estimates of improvements in productivity after application range from 35% to more than 200%.

2. Increased Accuracy.

CASE Tools can provide ongoing debugging and error checking which is very vital for early defect removal, which actually played a major role in shaping modern software. (Brathwaite)

The importance of early defect removal is very important. Less effort and time are consumed if corrections are made at an early stage, such as the design stage. As the system grows larger, it becomes difficult to modify. Error identification becomes harder. In some cases, restraining could be involved and that could incur cost in terms of time and effort. Managing and communicating problems among large teams can be difficult to manage, time consuming, and costly.

3. Reduced Lifetime Maintenance

As a result of better design, better analysis and automatic code generation, automatic testing and debugging overall systems quality improves. There is better documentation also. Thus, the net effort and cost involved with maintenance is reduced. (Brathwaite) Also, more resources can be devoted to new systems development. CASE provides reengineering tools which can be very important because they make this process more efficient, less time consuming, and less expensive by discovering older parts of the system which can be reused.

4. Better Documentation

By using CASE Tools, vast amounts of documentation are produced along the way. Most tools have revisions for comments and notes on systems development and maintenance.

5. Programming in the hands of non programmers

With the increased movement towards object oriented technology and client server bases, programming can also be done by people who don't have a complete programming background. It would be important to understand the logic of the program and possess the ability to analyze the organization of the program and all the intricate details involved in generating good software. (Brathwaite) By using the lower case tools, it could be possible to develop a software from the initial design and analysis phase.

6. Intangible Benefits

CASE Tools can be used to allow for greater user participation, which can lead to better acceptance of the new system. This can reduce the initial learning curve. (Brathwaite)

LIMITATIONS

1. Tool Mix

It is important to make an appropriate selection of tool mix to get cost advantage. CASE integration and data integration across all platforms is also very important. The ability to share the results of work done on one CASE tool with another CASE tool is perhaps the most important type of CASE integration.

2. Cost

CASE is not cheap! Infact, most firms engaged in software development on a small scale do not invest in CASE tools because they think that the benefits of CASE are justifiable only in the in the development of large systems. The cost of outfitting every systems developer with a preferred CASE tool kit can be quite high. Hardware and systems, software, training and consulting are all factors in the total cost equation.

3. Learning Curve

In most cases, programmer productivity may fall in the initial phase of implementation, because users need time to learn the technology. In fact, a CASE consulting industry has evolved to support uses of CASE tools..