Bringing British Education to You

www.nccedu.com

Information Systems Analysis

Topic 2:

Hard Approaches to the Analysis of Information Systems

V1.0

Objectives

- Define the term hard approach to systems analysis
- Identify business situations where a hard approach to systems analysis might be appropriate
- Define and explain the abbreviation SSADM
- Identify the advantages, disadvantages of SSADM
- Define and explain the abbreviation DFD
- Define and explain terminology associated with DFDs
- Illustrate the use of DFDs
- Construct DFDs
- Provide solutions to business problems using DFDs

Hard Approach to Information Systems Analysis

- A ***Hard Approach*** to Information Systems Analysis refers to taking a highly structured approach to the analysis of Information Systems.

- This approach follows a logical sequence of steps and adheres to rules, guidelines and standards.

When a Hard Approach to Systems Analysis Might be Appropriate

- It is particularly appropriate to use when working on large, complex information systems, such as government systems.

- It can also be used for smaller-scale business information systems projects.

Examples of Hard Approach Methodologies

- Structured Systems Analysis and Design Methodology (SSADM)
- Prototyping
- Joint Application Design (JAD)
- Rapid Analysis and Design (RAD)
- Dynamic Systems Development Method (DSDM)
- Scrum
- Agile methodology

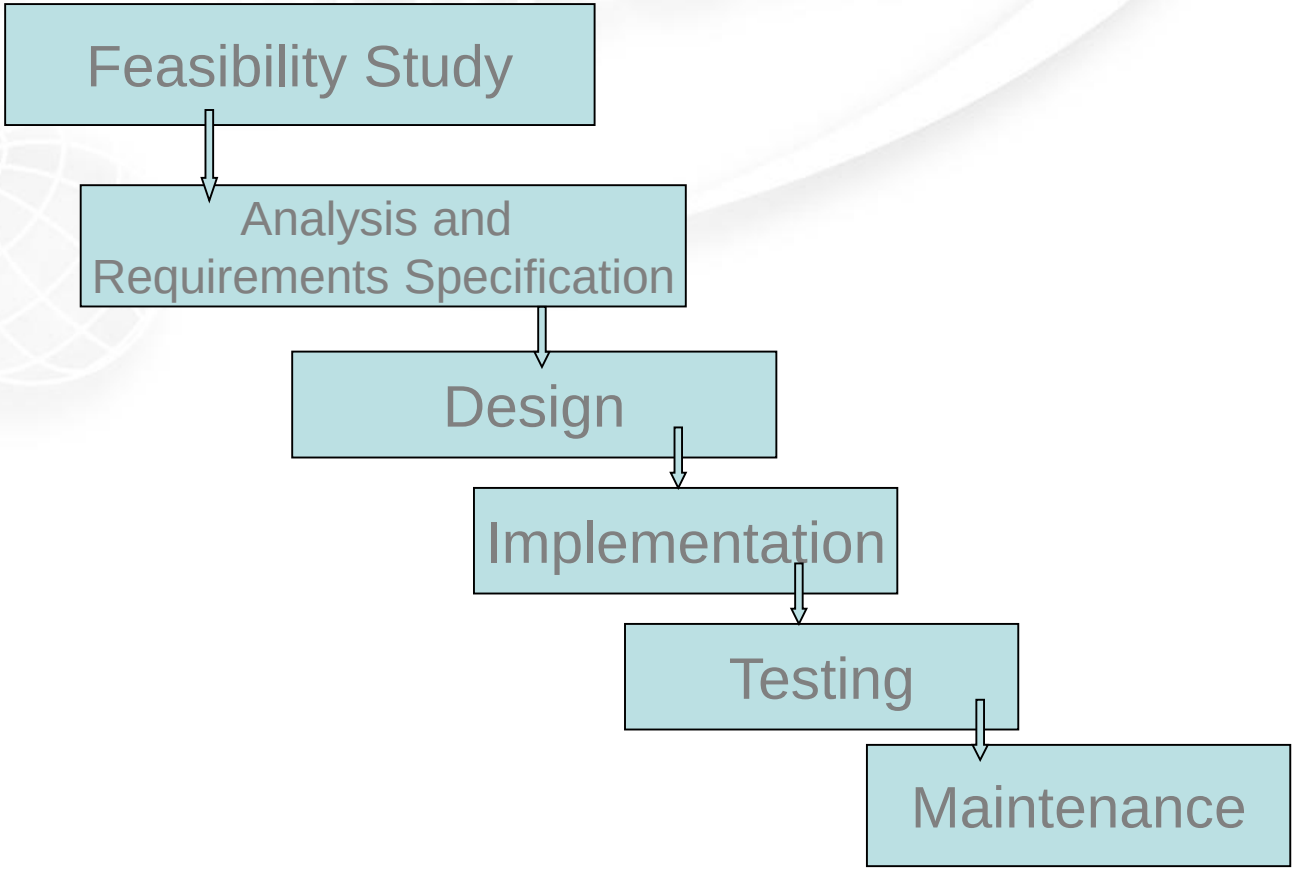
SSADM Views of a System

- SSADM can be used to look in detail at three views of a system:
  - ***The Process View*** describes the processes (functions) carried out by an information system, how data is moved around the system and how it changes as it is processed.
  - ***The Data View*** describes the data and information the system uses.
  - ***The Event View*** describes the events that set the processes running and the effect of external events on the data.

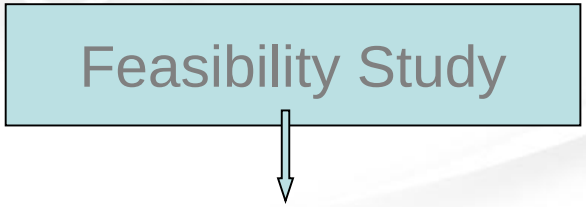
Stages of SSADM

- If it is undertaken thoroughly, SSADM can help to produce well-documented and accurate information systems.
- The following stages are worked through sequentially:
  - Feasibility Study
  - Analysis and Requirements Specification
  - Design
  - Implementation
  - Testing
  - Maintenance

SSADM (Waterfall Method)



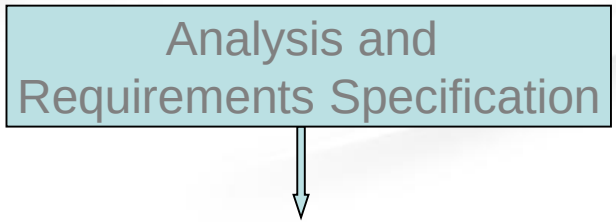
Feasibility Study



- Examines and determines whether a project is technically, financially and socially ***feasible***

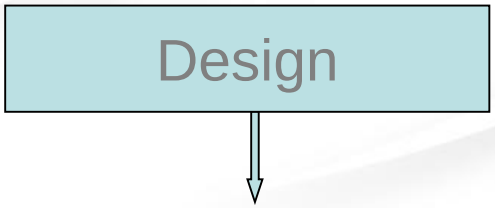
- Determines whether the project is cost-effective; for example, a ***cost-benefit analysis*** is undertaken

Analysis and Requirements Specification



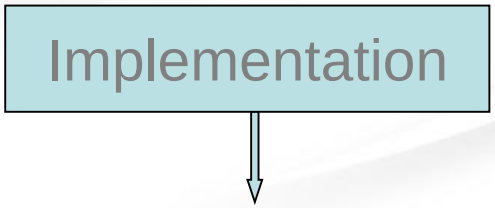
- All aspects of the system are analysed, e.g. the hardware and the software.
- The requirements are then defined, e.g. the technical specification is produced.

Design



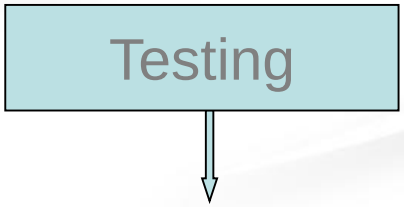
- All aspects of the system are designed, e.g. the software is coded.

Implementation



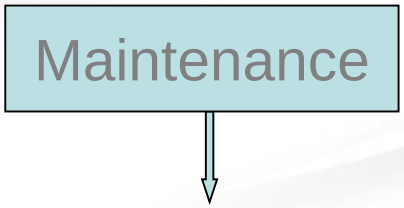
- The system is introduced to the organisation either directly, in a phased changeover, or running in parallel with the existing system until the new system is working successfully.

Testing



- All aspects of the system are tested to ensure robustness and reliability.

Maintenance



- The system must be maintained regularly to ensure robustness and reliability.

Advantages of SSADM

- Each step of the Waterfall method needs to be completed before progression onto the next one. This aims to ensure that all procedures associated with each step are undertaken.
- It is easy to measure progress by referring to the objectives defined for each step.
- It ensures thorough planning and scheduling.

Disadvantages

- There is a lack of flexibility, e.g. if the requirements are not specified correctly or change later in the project, it can be expensive to repeat the requirements stage or it may not be possible to return to this stage.
- There is often limited user involvement as this method tends to concentrate on the technical requirements.
- A project can often take longer to deliver than other methods that allow stages in a project to be repeated, e.g. the Agile methodology (referred to in Topic 6).

SSADM Techniques

- SSADM uses three techniques to provide different views of the same system:
  - ***Logical Data Modelling*** illustrates the structure of the data, e.g. entity types, entity attributes and the relationships between the entities.
  - ***Data Flow Modelling*** illustrates the flow of data in and out of the system and the data processing.
  - ***Entity/Event Modelling*** illustrates the way in which data in the system changes over time by events acting on entities.
- Each technique is cross-referenced against the others to ensure accuracy of detail.

Data Flow Modelling (DFD)

- Identifies, models and documents how data moves around an information system; how data enters and leaves the system; what changes the data and where the data is stored:
  - ***Processes***: activities that transform data from one form to another
  - ***Data stores***: where data is stored temporarily or permanently
  - ***External entities***: outside the system boundary, they show where data comes from (its ***source***) or where data is sent to (its ***sink***) e.g. people, organisations
  - ***Data flows***: the movement of data to or from a process

DFD - 1

- A DFD refers to a ***Data Flow Diagram***.

- A DFD can be used to illustrate:
  - the functions that an information system performs
  - the flow of data into it
  - the flow of data within it
  - the flow of data out of it

DFD - 2

- A DFD can be used to document the following information:
  - what processing is done: when, how, where and by whom
  - what data is required for the processing: for what purpose, by whom and by when

DFD - 3

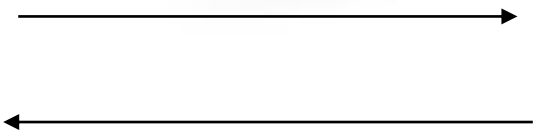
- Data Flow Diagrams can describe an information system in different ways:
  - ***Current Physical DFD:*** what the current system does
  - ***Current Logical DFD:*** how it does what it does, e.g. the processing
  - ***Required Logical DFD:*** what it should do e.g. the processing requirements of the proposed system
  - ***Required Physical DFD:*** how it should do what it needs to do

Advantages of DFDs

- They are usually simple to construct and are easy to understand.
- They also illustrate the ***boundary*** of a system.
- They can be constructed to represent an information system at different levels of detail:
  - 1st level illustrates an overview of the whole system
  - 2nd level more detail of 1st level
  - 3rd level more detail of 2nd level, etc.
- Therefore a complex system can be broken down into smaller diagrams (***sub-processes***) – this is described as ***decomposition.***

Data Flow Notation

- The flow of data is shown as an arrowed line with the arrowhead showing the direction of flow:



- Each data flow should be uniquely identified by a descriptive name, e.g. Payment

Data Flow Guidelines

- Data flows are allowed between external entities and processes but are not allowed between external entities and data stores.
- Data flows from external entities must flow into processes.
- Data flows to external entities must flow from processes.
- Processes and data stores must have inputs and outputs.
- Inputs to data stores only flow from processes.
- Outputs from data stores only flow to processes.

Process Notation

- May be shown as a rectangular box, a circle or an oval

- It must be given a name in the bottom section describing what the process does, e.g. issues receipt.
- An identifying number can be put in the top left corner, e.g. 1
- The top right hand side refers to the location of the process or the people responsible for it, e.g. Data Entry Personnel.

Process Guidelines

- Data flows are allowed between processes and external entities
- Data flows are allowed between processes and data stores
- Data flows are allowed between different processes

Data Store Notation

- Can be shown by an open-ended box with a descriptive name, e.g. Orders

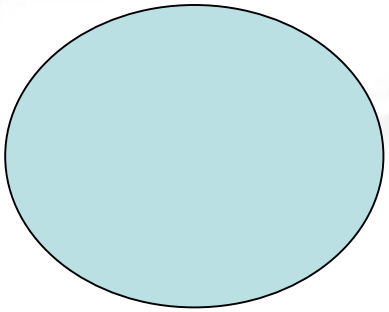
- It is also given a reference number prefixed by a letter:
  - D indicates a permanent computer file
  - M indicates a manual file
  - T indicates a file that is deleted after processing
  - e.g. T1

Data Store Guidelines

- Data flows are allowed between data stores and processes.
- Data flows are not allowed between data stores and external entities or between one data store and another.
- Data stores require a process to initiate communication of information.

External Entity Notation

- Can be shown as a circle



- It should be given a descriptive name, e.g. Supplier

External Entity Guidelines

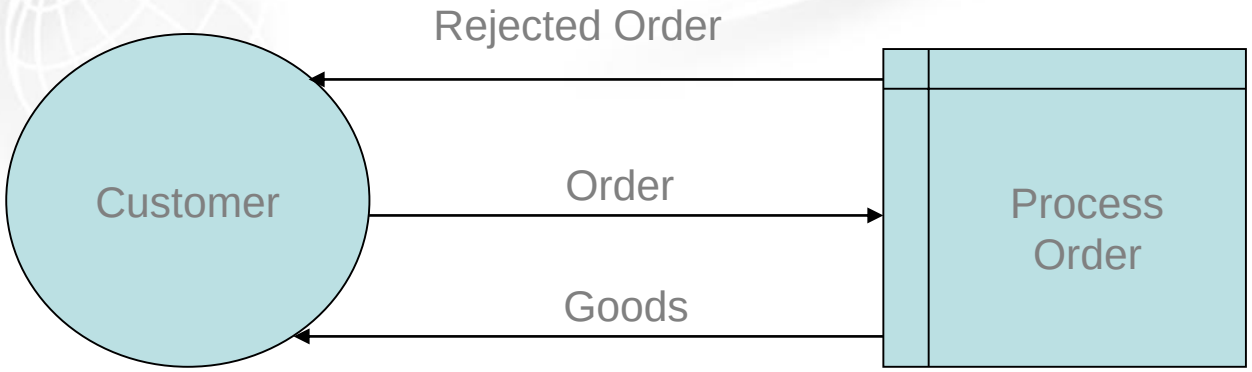
- External entities may be duplicated to avoid crossing data flow lines.
- A stripe is drawn across the left hand corner of any duplicated entities.
- A lowercase letter can also be added to each external entity to identify them.

Constructing a DFD

- Meaningful names should be given to processes, data flows, data stores and external entities.
- Each process should be numbered.
- The notation should be consistent.
- Guidelines must be followed.
- Complex DFDs must be avoided.
  - To avoid complex DFDs, use decomposition.

A Basic DFD

- Where would a data store be added here?



Summary

This topic covers

- An overview of:
  - types of hard approach methodologies
  - structured systems analysis and design methodology, tools and techniques
  - advantages and disadvantages of structured systems analysis and design methodologies
  - the purpose and potential of dataflow diagrams

References

- SmartDraw (2011). Data Flow Diagrams. [Available Online] http://www.smartdraw.com/resources/tutorials/data-flow-diagrams/
- Freetutes.com (2011). DFD Example - General Model Of Publisher's Present Ordering System. [Available Online] http://www.freetutes.com/systemanalysis/sa5-dfd-ordering-system.html