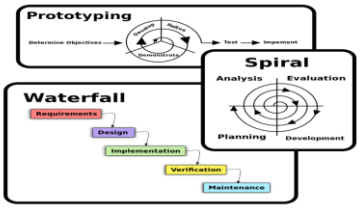
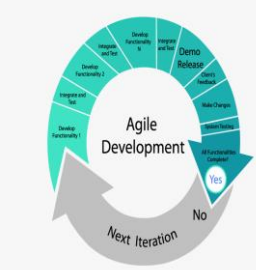
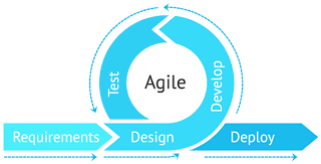


Software Engineering Requirements Modelling



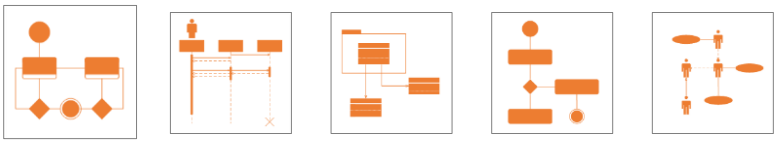


Outline

- Requirements Analysis
  - Domain Analysis
  - Analysis Rules of Thumb
- Requirements Modelling Approaches:
  - Scenario Based
  - Information Based
  - Flow Oriented Strategies.

The requirement analysis model

- Describe what the customer wants built
- Establish the foundation for the software design
- Provide a set of validation requirements
- System information, system functions, system behavior



3

The requirement analysis model action

- The requirements modeling action results in one or more of the following types of models:
  - Scenario-based models of requirements from the point of view of various system “actors”
  - Data models that depict the information domain for the problem
  - Class-oriented models that represent object-oriented classes (attributes and operations) and the manner in which classes collaborate to achieve system requirements
  - Flow-oriented models that represent the functional elements of the system and how they transform data as it moves through the system
  - Behavioral models that depict how the software behaves as a consequence of external “events”

4

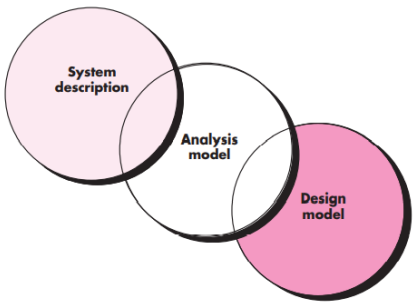
Overall Objectives and Philosophy

- Throughout requirements modeling, your primary focus is on what, not how.
  - What user interaction occurs in a particular circumstance?
  - what objects does the system manipulate?
  - what functions must the system perform?
  - what behaviors does the system exhibit?
  - what interfaces are defined?
  - what constraints apply?
- Complete specification of requirements may not be possible at this stage.
- The customer may be unsure of precisely what is required for certain aspects of the system.
- The developer may be unsure that a specific approach will properly accomplish function and performance.

5

The requirement analysis model objective

1. to describe what the customer requires,
2. to establish a basis for the creation of a software design, and
3. to define a set of requirements that can be validated once the software is built.



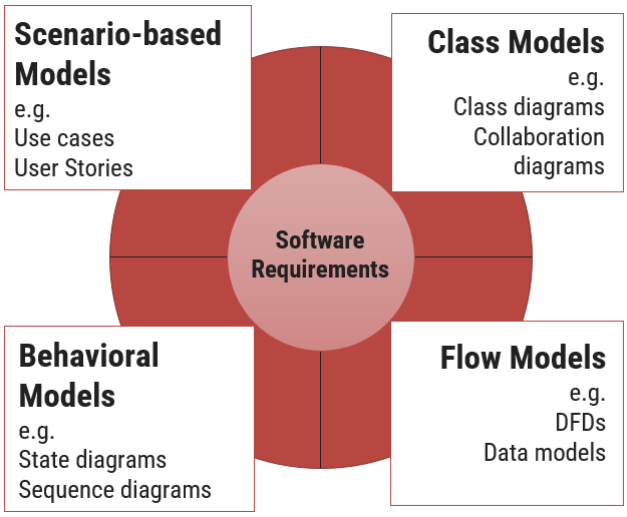
6

Analysis rule of Thumb

- Make **sure all points** of view are **covered**
- Every **element** should **add value**
- **Keep** it **simple**
- **Maintain** a high level of **abstraction**
- **Focus** on the **problem domain**
- **Minimize** system **coupling**
- **Model** should **provides value to all stakeholders**

7

Elements of the Requirements Model

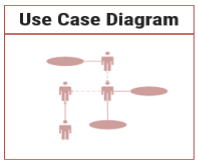


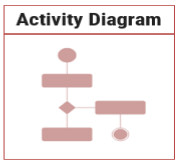
8

Elements of the Requirements Model

Scenario-based elements

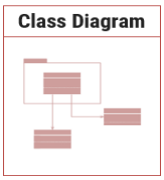
- Describe the system from the user's point of view using scenarios that are depicted (stated) in use cases and activity diagrams





Class-based elements

- Identify the domain classes for the objects manipulated by the actors, the attributes of these classes, and how they interact with one another; which utilize class diagrams to do this.

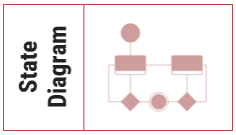


9

Elements of the Requirements Model

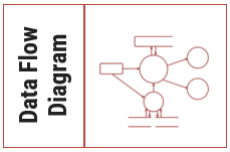
Behavioral elements

- Use state diagrams to represent the state of the system, the events that cause the system to change state, and the actions that are taken as a result of a particular event. This can also be applied to each class in the system.



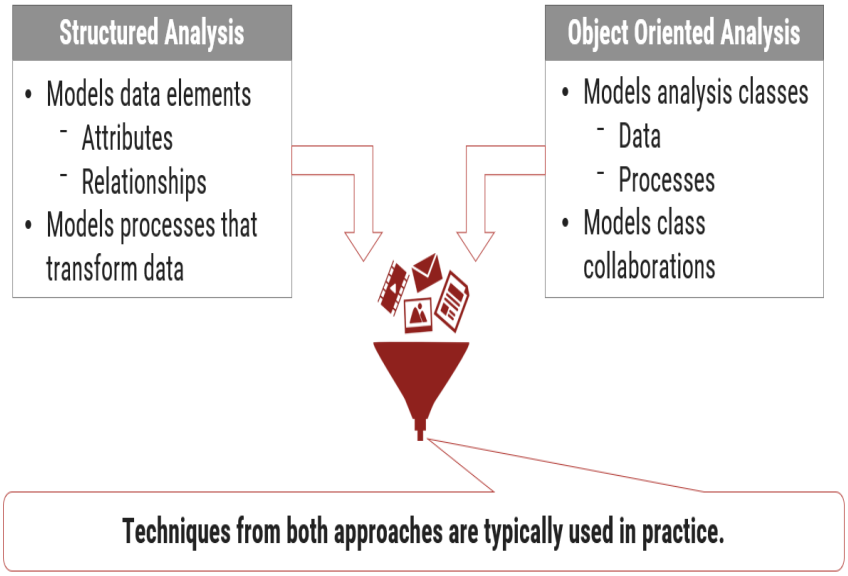
Flow-oriented elements

- Use data flow diagrams to show the input data that comes into a system, what functions are applied to that data to do transformations, and what resulting output data are produced.



10

Analysis Modeling Approaches



11

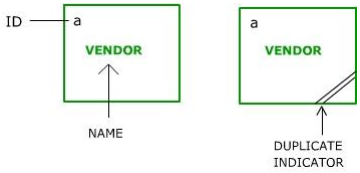
Flow-Oriented Modeling

- Although the data flow diagram (DFD) and related diagrams and information are not a formal part of UML, they can be used to complement UML diagrams and provide additional insight into system requirements and flow.
- The DFD takes an input-process-output view of a system. That is, data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software.
- The DFD is presented in a hierarchical fashion. That is, the first data flow model (sometimes called a level 0 DFD or context diagram) represents the system as a whole.
- Subsequent data flow diagrams refine the context diagram, providing increasing detail with each subsequent level

12

Notations of DFD

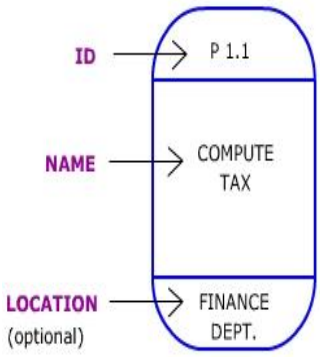
- ***External agents*:** They are external to the system, but interact with the system. They must be drawn at level 0, but need not be drawn at level 2 onwards. Duplicates are to be identified. They must be given meaningful names.



13

Notations of DFD

- ***Process*:** They indicate information processing activity. They must be shown at all levels, At level 0, only a single process, depicting the system is shown. On subsequent levels, the number of processes should be limited to 7 ± 2. No duplicates are allowed.



PROCESS ID:

A HIERARCHIC NUMBER
e. g., 1, 1.1, 1.1.1
1.2 1.2.1

NAME
- STRONG VERG (e. g., COMPUTE, CREATE) AND OBJECT ONLY. NO SUBJECT.
- AVOID WEAK VERBS (e. g., MAINTAIN, PROCESS)

LOCATION (optional)
WHERE PERFORMED (e. g., PURCHASE DEPT, FINANCE DEPT)

14

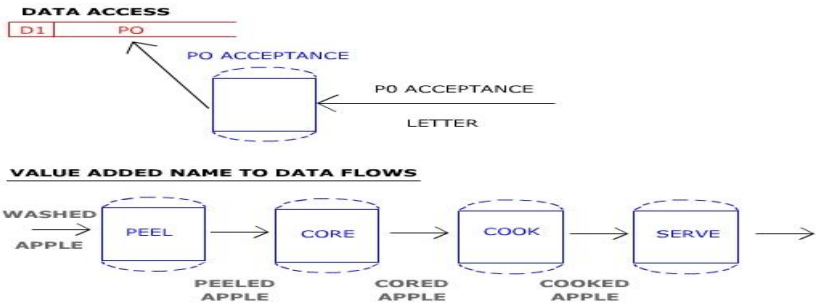
Notations of DFD

- ***Data Stores*:** They are used to store information. They are not shown at level 0. All data stores should be shown at level 1. Duplicates must be indicated.

15

Notations of DFD

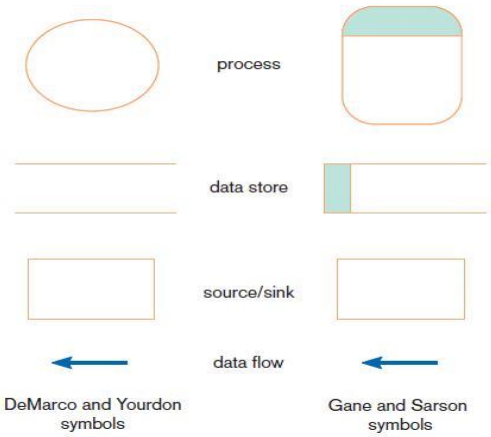
- ***Data Flows*:** They indicate the flow of information. They must be shown at all levels and meaningful names must be given.



- *Examples*: Customer places sales orders. The system checks for availability of products and updates sales information
- Company receives applications. Checks for eligibility conditions. Invites all eligible candidates for interview. Maintains a list of all candidates called for interview. Updates the eligibility conditions as and when desired by the management

16

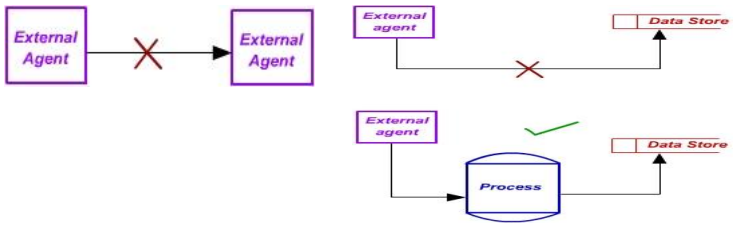
Definitions and Symbols



17

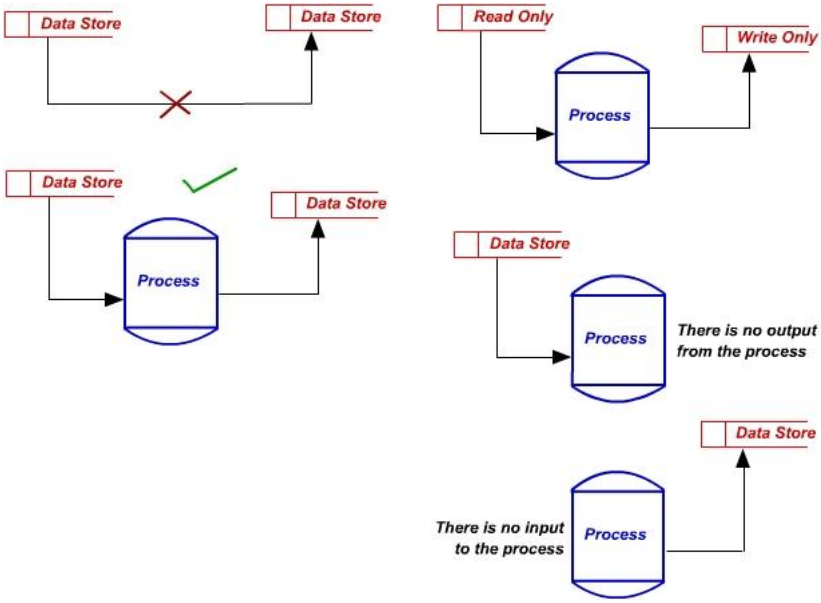
Guideline for DFD

1. the level 0 data flow diagram should depict the software/system as a single bubble;
2. primary input and output should be carefully noted;
3. refinement should begin by isolating candidate processes, data objects, and data stores to be represented at the next level;
4. all arrows and bubbles should be labelled with meaningful names;
5. information flow continuity must be maintained from level to level,
6. one bubble at a time should be refined.



18

Guideline for DFD



19

Rules for DFD

- **Context Diagram (Level 0)**
  - The major information flows between the entities and the system.
  - A context diagram addresses only one process.
- **Level 1**
  - Level 1 DFD, must balance with the context diagram it describes.
  - Input going into a process are different from outputs leaving the process.
  - Data stores are first shown at this level.

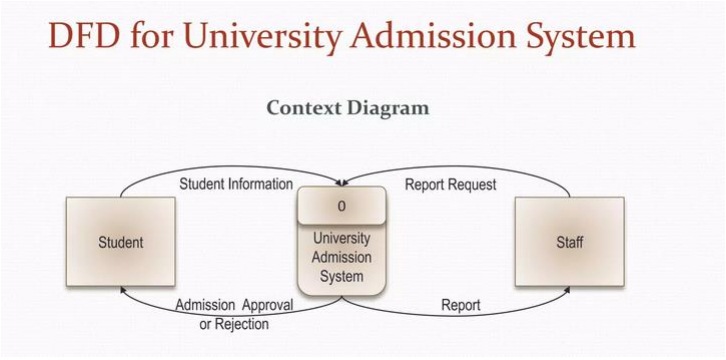
20

Rules for DFD

- **Level 2**
  - Level 2 DFD must balance with the level 1 it describes.
  - Input going into a process are different from outputs leaving the process.
  - Continues to show data stores.
- **Numbering**
  - On level 1 processes are numbered 1,2,3,…
  - On level 2 processes are numbered x.1,x.2,x.3… where x is the number of the parent level 1 process.
  - Number is used to uniquely identify process not to represent any order of processing
  - Data store numbers usually D_1, D_2, D_3,…

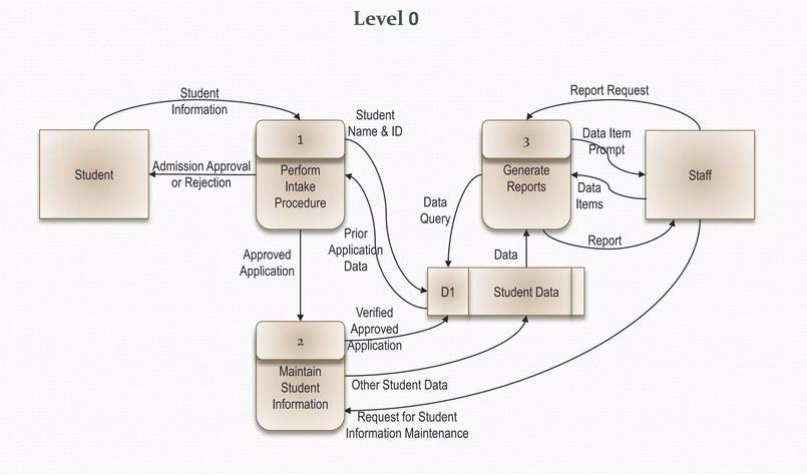
21

Example

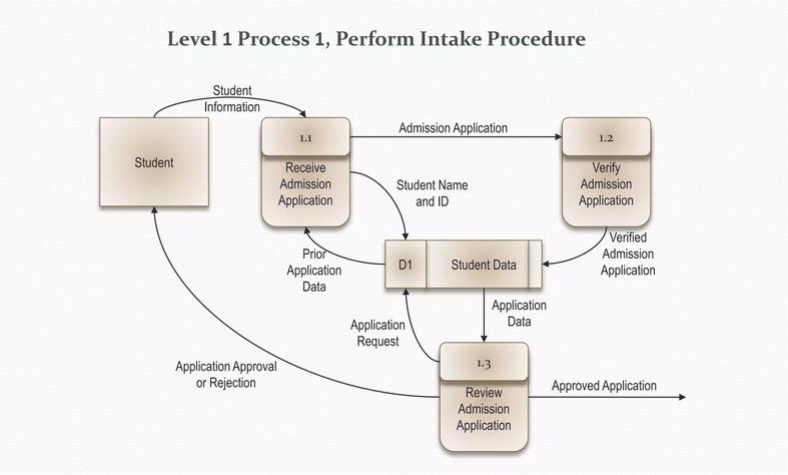


22

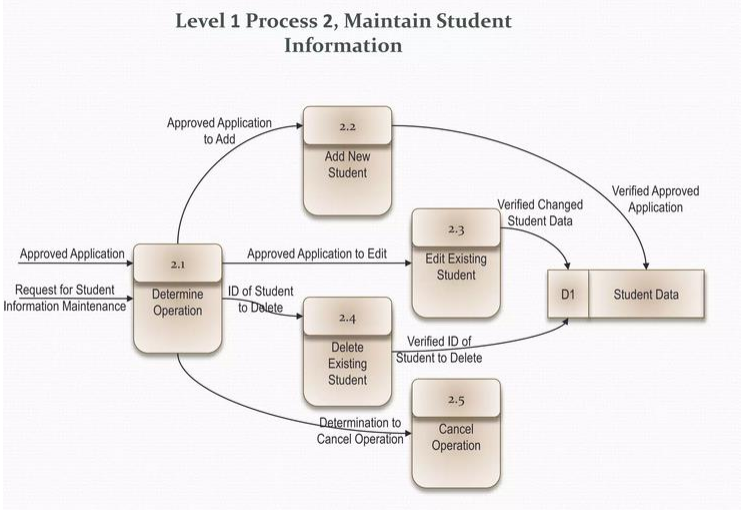
Example



Example



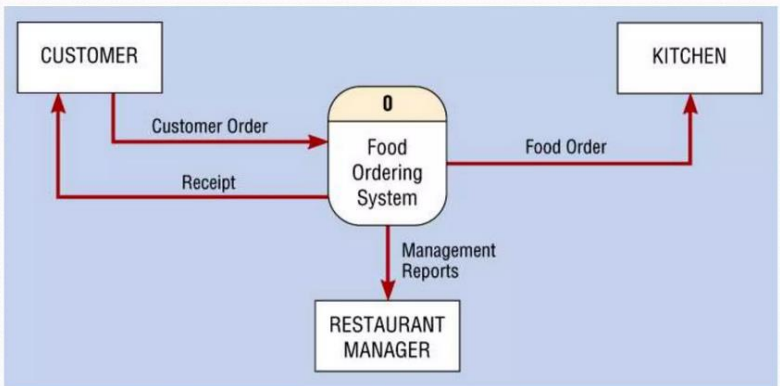
Example



25

Example

- 1 process represents the entire system.
- Data arrows show input and output.
- Data Stores NOT shown. They are within the system.



26

Layers of DFD abstraction for course registration system