CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Object Oriented Software Engineering

LECTURE 8

Today's Outline:

Building Analysis Model:

- Requirement Analysis
- Data modelling Concepts
- Flow Oriented Modelling

Building the Analysis Model

- **What is it?**
- The analysis model, actually a set of models, is the first technical representation of a system. The written word is a wonderful vehicle for communication, but it is not necessarily the best way to represent the requirements for computer software.
- Analysis Modeling uses a combination of text and diagrammatic forms to depict requirements for data, function, and behavior in a way that is relatively easy to understand, straightforward to review for correctness, completeness, and consistency.

Building the Analysis Model

- **Who does it?**
- A software engineer (called as analyst) builds the model using requirements elicited from the customer.
- **Why is it important?**
- To validate software requirement, you need to examine them from a number of different points of view. Analysis modeling represents requirements in multiple ‘dimensions’.

Building the Analysis Model

- **What is the work product?**
- A wide array of diagrammatic forms may be chosen for the analysis model. Each of these representations provides a view of one or more of the model elements.

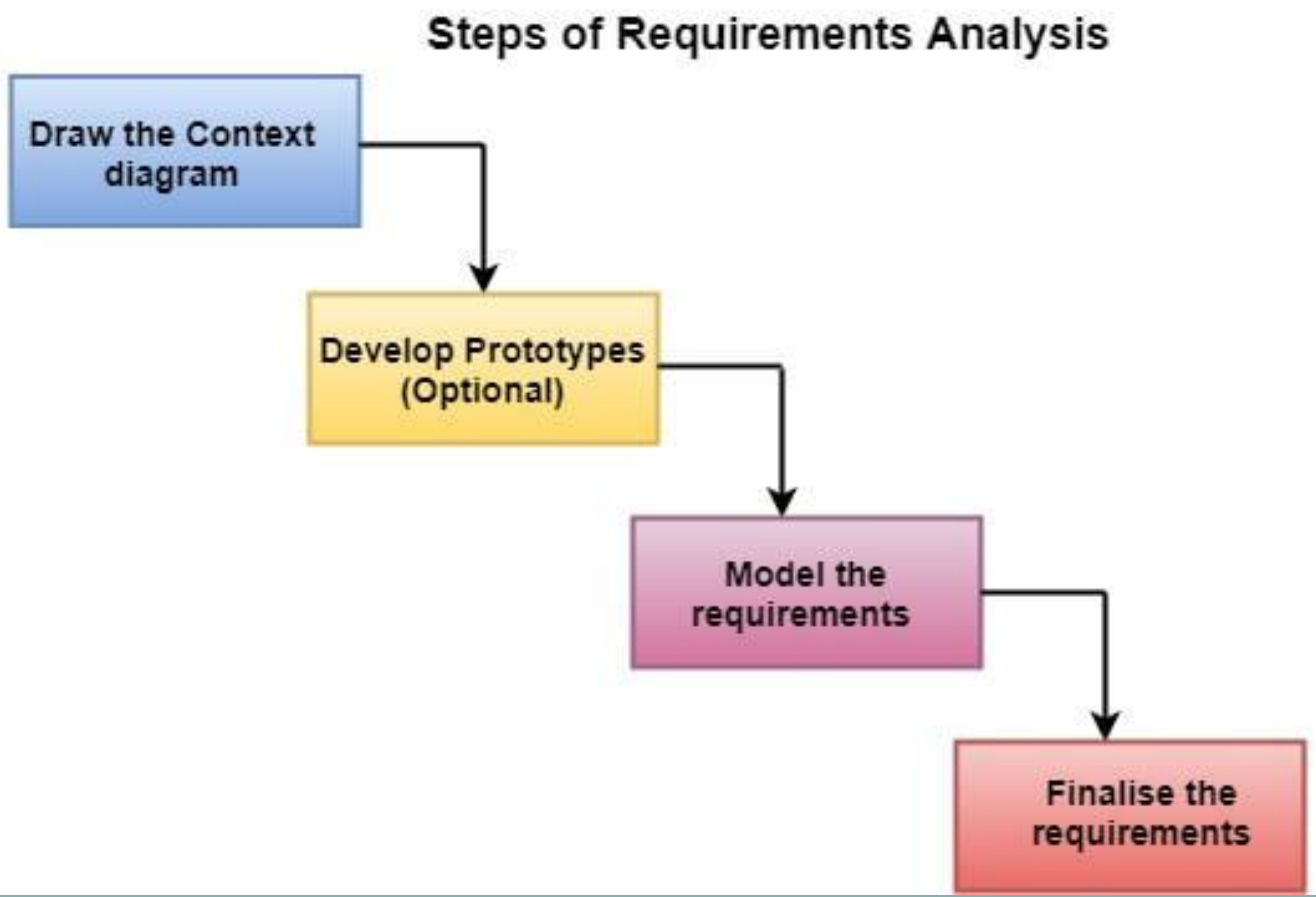
Overview of Analysis Modeling

- At a technical level, SE begins with a building an analysis model of a target system.
- Requirements analysis
- ➢ species software's operational characteristics
- ➢ indicates software's interface with other system elements
- ➢ establishes constraints that software must meet
- Objectives
- 1 to describe what the customer requires
- 2 establish a basis for the creation of a SW design
- 3 to define requirements that validated a set of can be once the software is built

Analysis Modeling

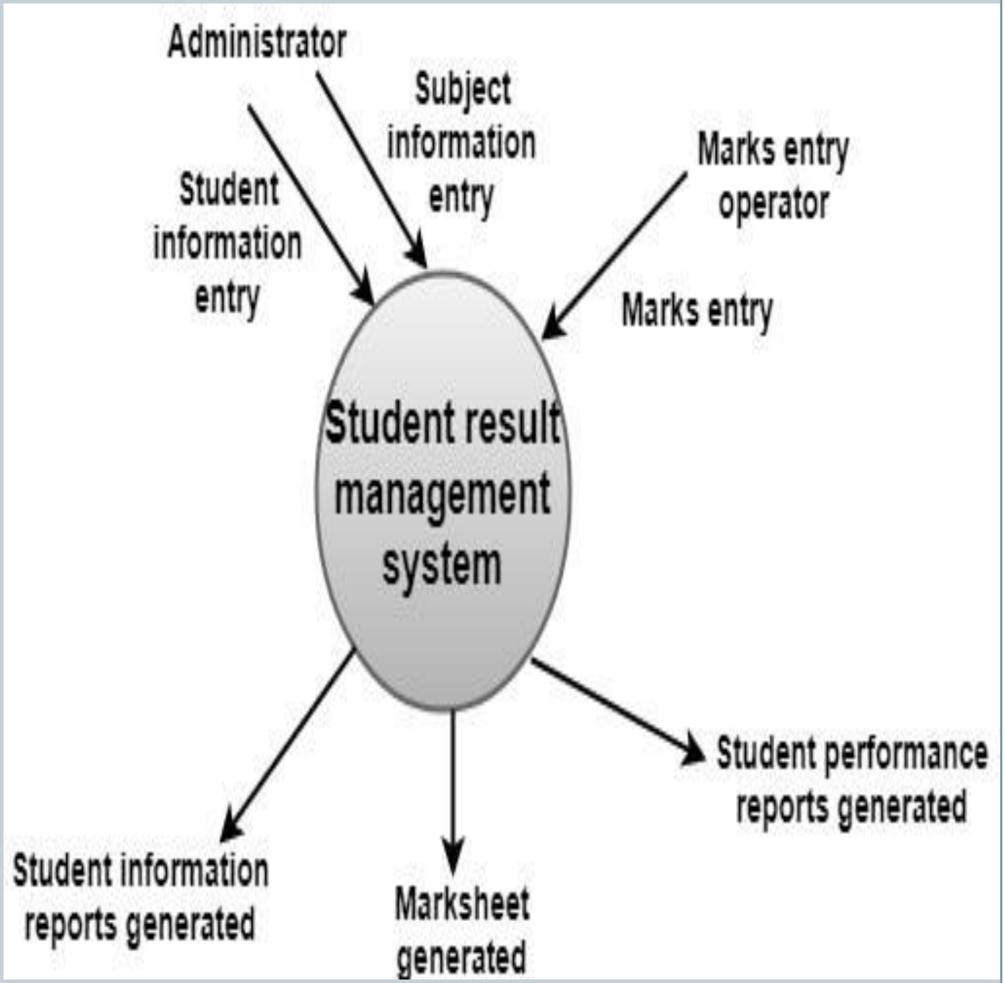
- Requirements analysis allows the software engineer to:
- elaborate on basic requirements established during earlier requirement engineering tasks
- build models that depict
  - user scenarios
  - functional activities
  - problem classes and their relationships
  - system and class behavior
  - flow of the data as it is transformed

Steps of Requirements Analysis


Steps of Requirements Analysis
Draw the Context diagram
Develop Prototypes (Optional)
Model the requirements
Finalise the requirements

Steps of Requirements Analysis

- **(i) Draw the context diagram:** The context diagram is a simple model that defines the boundaries and interfaces of the proposed systems with the external world. It identifies the entities outside the proposed system that interact with the system.


Administrator
Subject information entry
Student information entry
Marks entry operator
Marks entry
Student result management system
Student performance reports generated
Student information reports generated
Marksheet generated

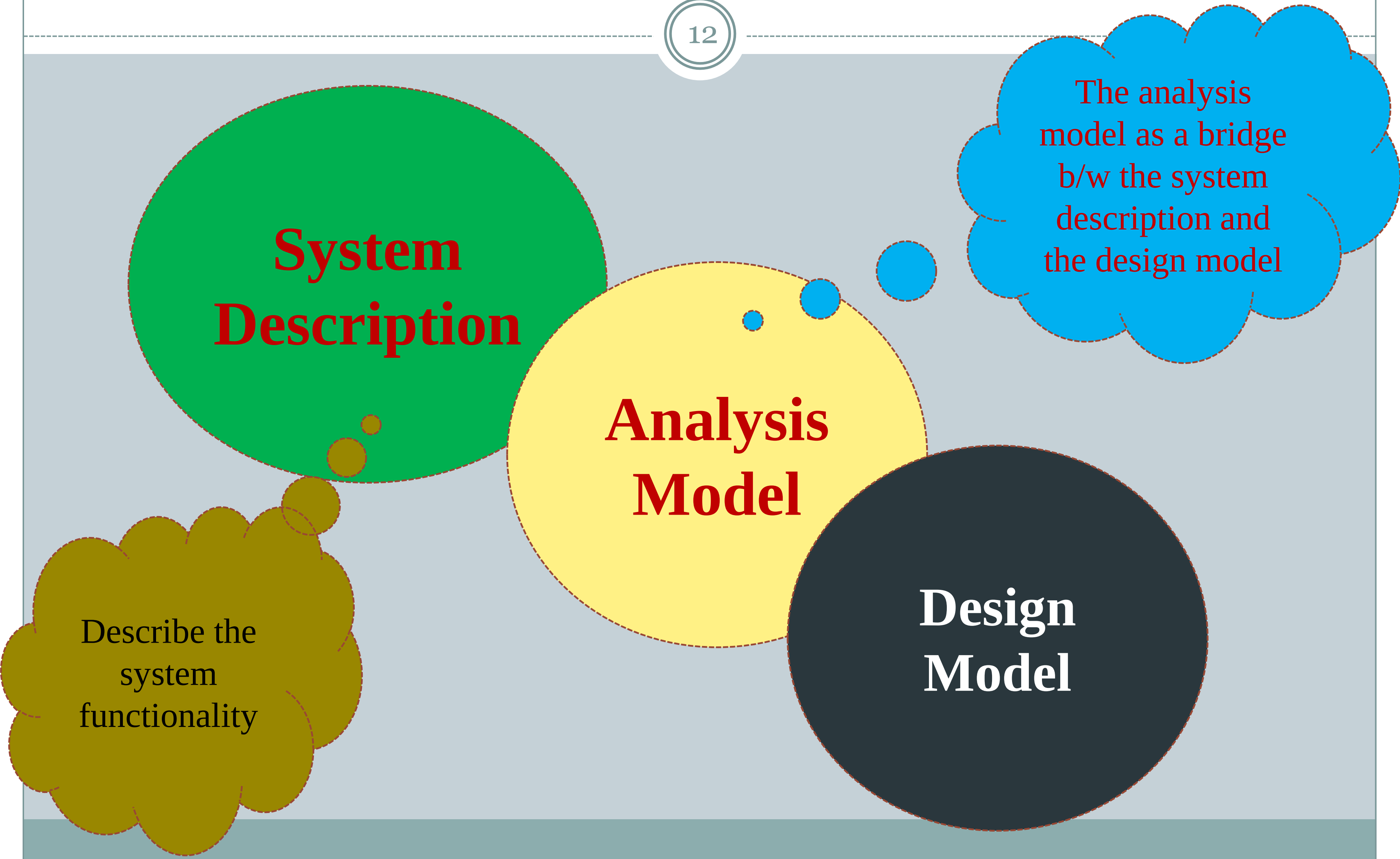
Steps of Requirements Analysis

- **(ii) Development of a Prototype (optional):** One effective way to find out what the customer wants is to construct a prototype, something that looks and preferably acts as part of the system they say they want. We can use their feedback to modify the prototype until the customer is satisfied continuously.
- **(iii) Model the requirements:** This process usually consists of various graphical representations of the functions, data entities, external entities, and the relationships between them. The graphical view may help to find incorrect, inconsistent, missing, and superfluous requirements. Such models include the Data Flow diagram, Entity-Relationship diagram, Data Dictionaries, State-transition diagrams, etc.

Steps of Requirements Analysis

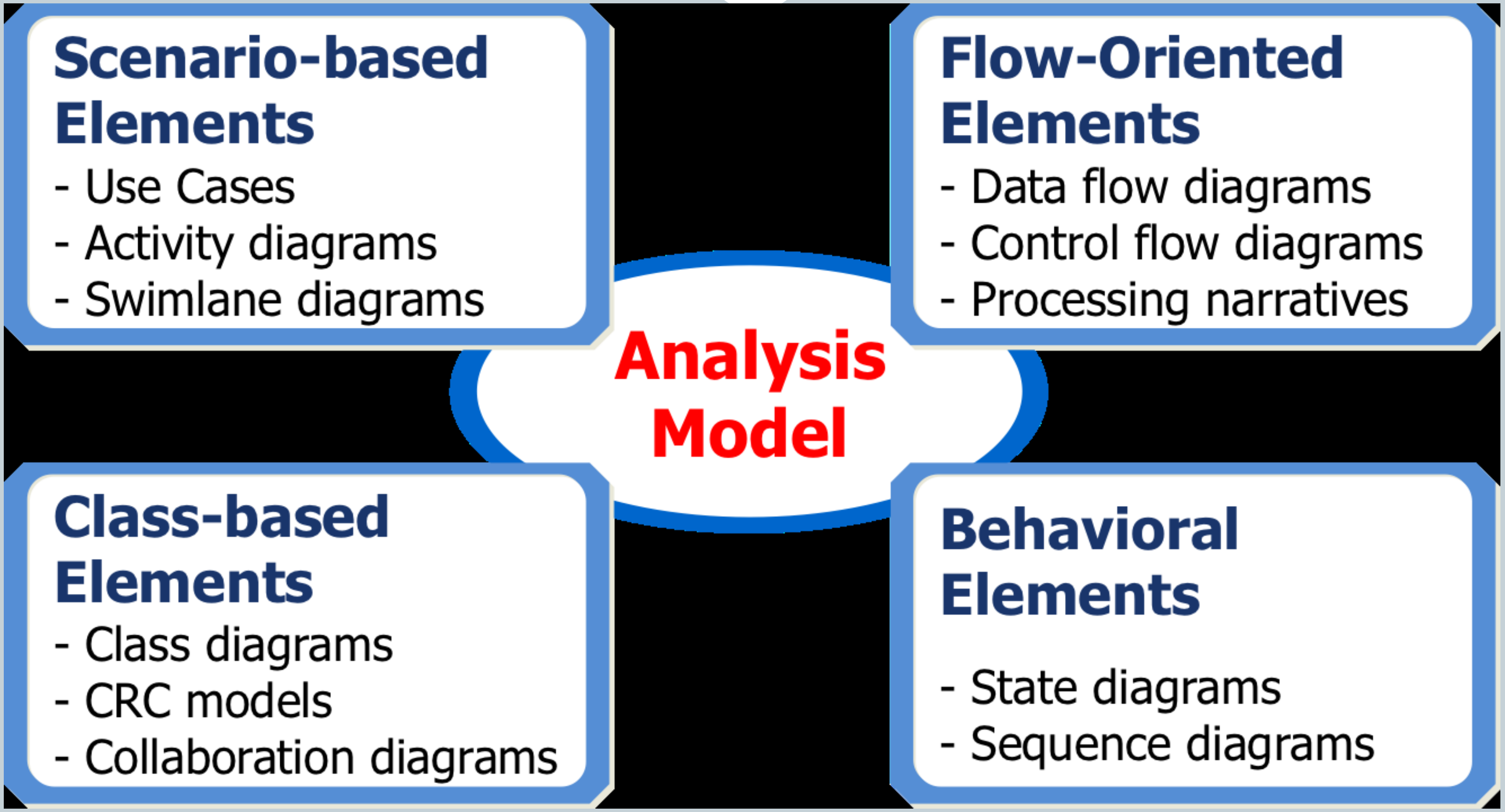
- **(iv) Finalize the requirements:** After modeling the requirements, we will have a better understanding of the system behavior. The inconsistencies and ambiguities have been identified and corrected. The flow of data amongst various modules has been analyzed. Elicitation and analyze activities have provided better insight into the system. Now we finalize the analyzed requirements, and the next step is to document these requirements in a prescribed format.

Requirement Analysis


System Description
Analysis Model
Design Model
The analysis model as a bridge b/w the system description and the design model
Describe the system functionality

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Elements of Analysis Model



Scenario-Based Modeling

- Use-cases are amply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases)
- 1 How should we write about?
- 2 How much should we write about?
- 3 How detailed should we make our description?
- 4 How should we organize the description?

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Scenario-Based Modeling

Activity Diagram

- Supplements the use-case by providing a diagrammatic representation of procedural flow

  - used as business modeling notation along with Business
- Process Modeling Notation

Swimlane Diagrams

- Allows the modeler to represent the flow of activities described by the use-case
- This diagram indicates which actor or analysis class has responsibility for the action described by an activity rectangle

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Class-based Modeling

Class-based modeling represents:

- objects that the system will manipulate
- operations (methods or services) that will be applied to the
- objects to effect the manipulation
- relationships (some hierarchical) between the objects
- collaborations that occur between the classes that are defined

The elements of a class-based model

- classes, objects, attributes, operations
- CRC models
- collaboration diagrams and packages

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Class-based Modeling

- Class-Responsibility-Collaborator (CRC) Modeling
- Analysis classes have \responsibilities"

  Responsibilities are the attributes and operations encapsulated by the class
- Analysis classes collaborate with one another

  a property transfer or the completion of a series of robot movements
- Collaborators are those classes that are required to provide a class with the information needed to complete a responsibility
- In general, a collaboration implies either a request for information or a request for some action

Behavioral Modeling

- Behavioral model indicates how software will respond to external events.

Behavioral model creation steps:

- evaluate all use-cases to fully understand the sequence of interaction within the system
- identify events that drive the interaction sequence and understand how these events relate to specfic objects
- Create a sequence for each use-case
- Build a state diagram for the system
- Review the behavioral model to verify accuracy and consistency

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Flow-Oriented Modeling

- The flow model, at its base, is **a simple graphical representation of how information and artifacts flow through the system as it is used**. … A flow model gives you an overview of how information, artifacts, and work products flow among user work roles and parts of the product or system as the result of user actions.

- It shows how data objects are transformed by processing the function.
- Represents how data objects are transformed as they move through the system

Rules of Thumb

1 The model should focus on requirements that are visible within the problem or business domain

- the level of abstraction should be relatively high

2 Each element of the analysis model should

- add to an overall understanding of software requirements
- provide insight into the

- information domain
- function of the system
- behavior of the system

3 Delay consideration of infrastructure and other non-functional models until design

4 Minimize coupling throughout the system

5 Be certain that the analysis model provides value to all stakeholders

6 Keep the model as simple as it can be

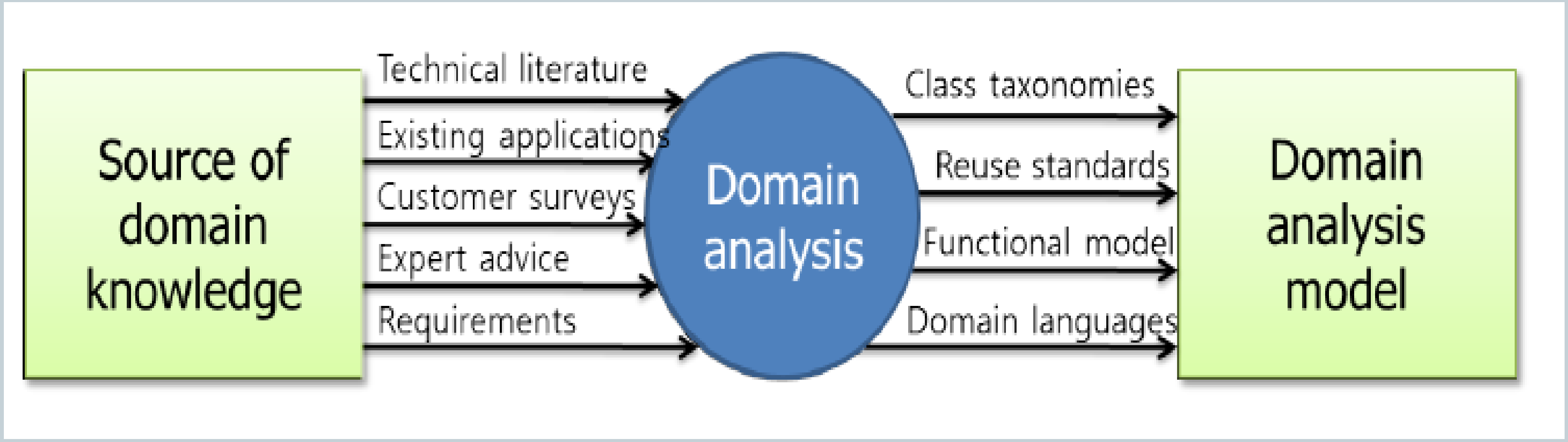
CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Domain Analysis

- Software Domain Analysis (Structured Analysis)
- the identification, analysis, and specification of common requirements from a specific application domain, typically for reuse on multiple projects within that application domain ...
- Object-Oriented Domain Analysis
- the identification, analysis, and specification of common, reusable capabilities within a specific application domain, in terms of common objects, classes, subassemblies, and frameworks . . .

- **by Donald Firesmith**
- Define the domain to be investigated
- Collect a representative sample of applications in the domain
- Analyze each application in the sample
- Develop an analysis model for the objects.

Domain Analysis



Data Modeling Concepts

- Analysis modeling often begins with data modeling. Analyst defines all data, objects that are processed within the systems, the relationships between the data objects, and other information that is pertinent to the relationships.

Data Modeling

Analysis modeling often begins with data modeling

- examines data objects independently of processing
- focuses attention on the data domain
- Indicates how data objects relate to one another

Relationship among data objects can be expressed in UML very well.

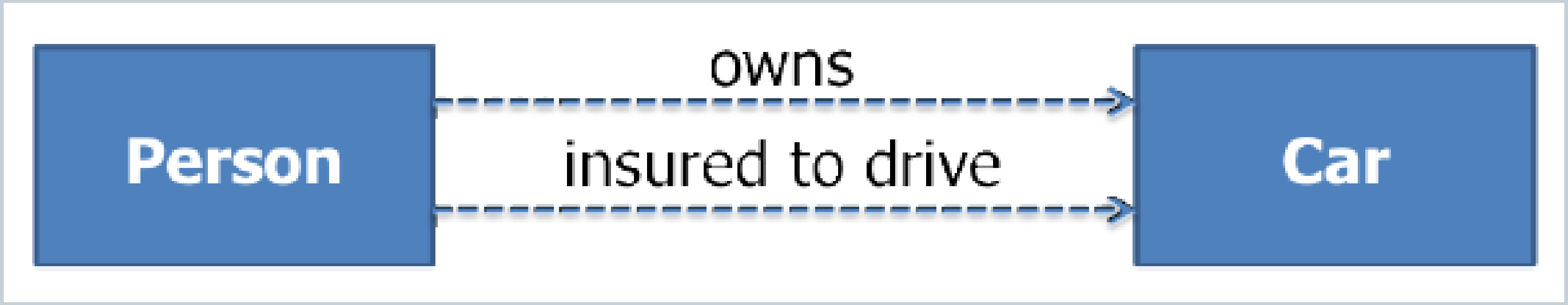
Typical data objects

- External entities: printer, user, sensor
- Things: reports, displays, signals
- Occurrences or events: interrupt, alarm
- Roles: manager, engineer, salesperson
- Organizational units: division, team
- Places: manufacturing floor
- Structure: employee record

Data Modeling

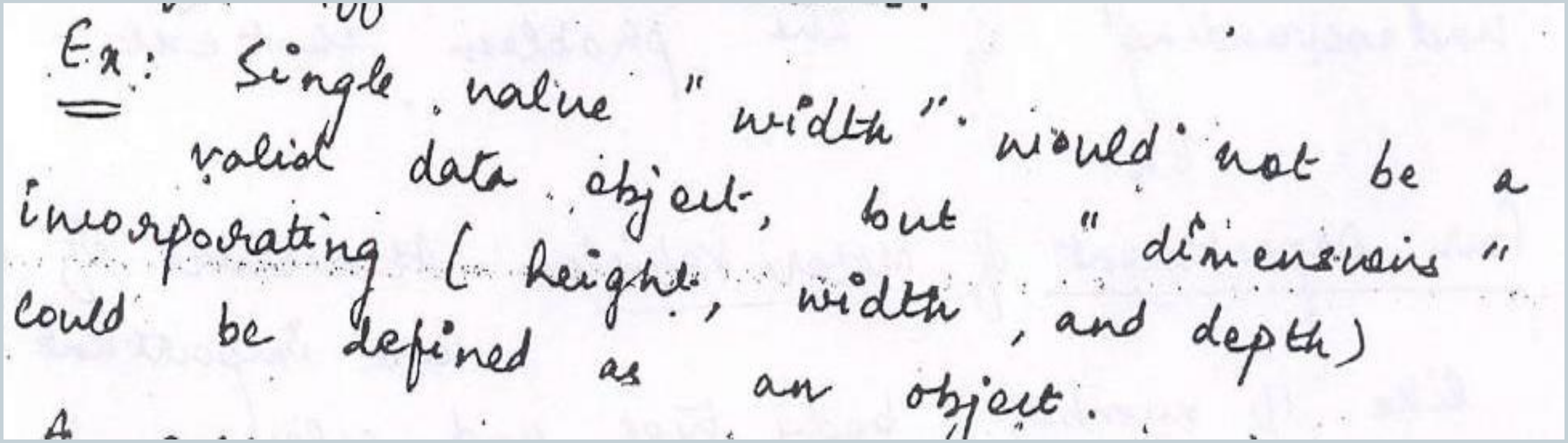
- Data objects, data attributes, relationships
  - Data objects are the representation of composite information that are process by software
  - A data object encapsulates data only

- Entity Relationship Diagram (ER Diagram)



Data Modeling Concepts

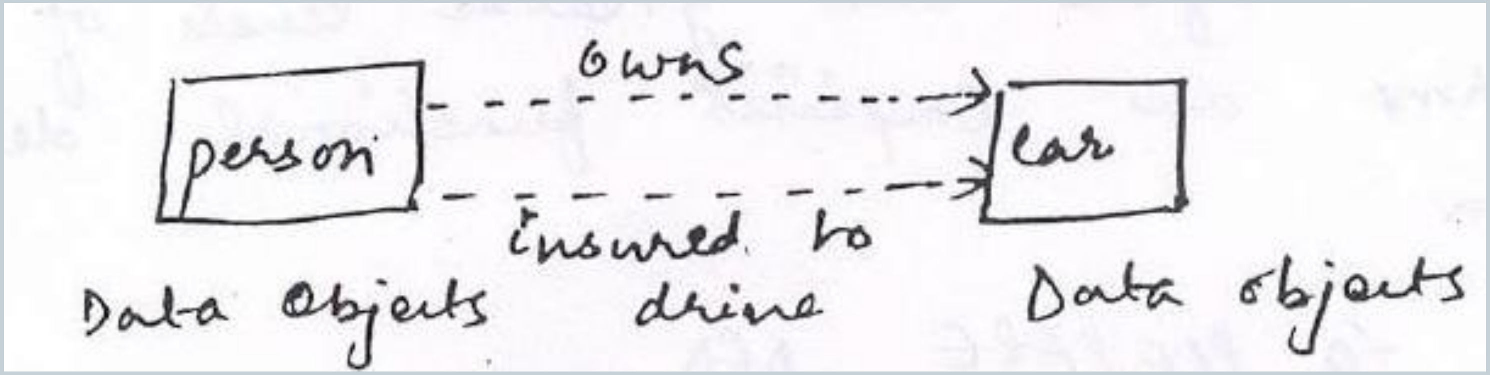
- A. Data Objects:
- A data objects is a representation of almost any composite information i.e. something that has a number of different attributes.



Ex: Single value "width" would not be a valid data object, but "dimensions" incorporating (height, width, and depth) could be defined as an object.

Data Modeling Concepts

- Relationships:
- Data objects are connected to one another in different ways.
- Consider the two data objects person and car.
- A person owns a car
- A person is insured to drive a car.



Data Modeling Concepts

- Cardinality and Modality:
- Suppose object X relates to object Y does not provide enough information for software engineering purposes.
- We must understand how many occurrences of object X are related to how many occurrences object Y. This leads to data modeling concept called “cardinality”.
- Example:
- 1:1, n:n etc.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Data Modeling Concepts

- The modality of a relationship is 0 (Zero) if there is no explicit need for the relationship to occur or the relationship is optional. The modality is 1 (one) if an occurrence of the relationship is mandatory.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Flow Oriented Modeling

- The flow oriented modeling represents how data objects are transformed at they move through the system. Derived from structured analysis, flow models use the data flow diagram, a modeling notation that depicts how input is transformed into output as data objects move through the system. Each software function that transforms data is described by a process specification or narrative. In addition to data flow, this modeling element also depicts control flow.

Flow Oriented Modeling

- Although flow oriented modeling is perceived as an outdated technique by some software engineers, it continues to be one of the most widely used requirements analysis notations in use today. Flow oriented modeling focuses on structured analysis and design, follows a top to down methodology and uses a graphical technique depicting information flows and the transformations that are applied as data moves from input to output.

Flow Oriented Modeling

- The modeling tools that are used to build a data flow oriented model include context diagrams, data flow diagrams, entity relationship diagram, control flow diagram, state transition diagram, data dictionary, process specification and control specification. Although the data flow diagram (DFD) and related diagrams and information are not a formal part of UML (Unified Modeling Language), they can be used to complement UML diagrams and provide additional insight into system requirements and flow. The flow oriented modeling 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.

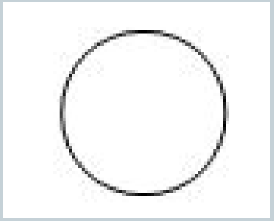
Data Flow Diagram :

- The data flow diagram represents the flows of data between different process in a business. It is a graphical technique that depicts information flow and transforms that are applied as data from input to output. It provides a simple, intuitive method for describing business processes without focusing on the details of computer systems. DFDs are attractive techniques because they provide what users do rather than what computers do. In DFD, there are four symbols are used :

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

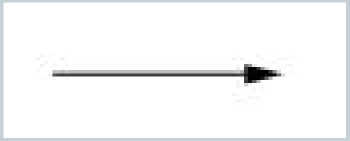
Data Flow Diagram :

- **1. Process :**



The circle represents the process. An activity that changes or transforms data flows. Since they transform incoming data to outgoing data, all processes must have inputs and outputs on a DFD.

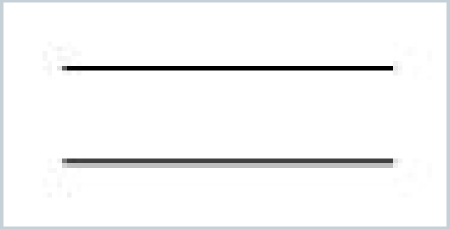
2. Data Flow :
The labeled arrows indicate incoming and outgoing data flow. Movement of data between external entities, processes and data stores is represented with an arrow symbol, which indicates the direction of flow.



CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Data Flow Diagram :

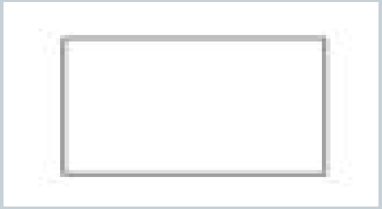
- **3. Data Store :**



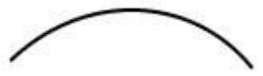
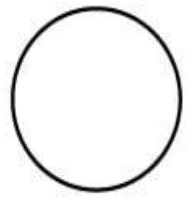
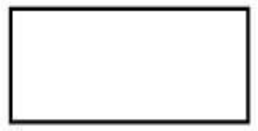
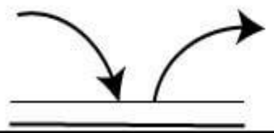
The rectangle represents an external entity. A data store does not generate any operations but simply holds data for later access.

- **4. External Entity :**

In Data Flow Diagrams external entities produce and consume data that flows between the entity and the system being diagrammed.



Data Flow Diagram

Symbol	Name	Function
	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
	Process	Perfroms Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

Symbols for Data Flow Diagrams

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

Data Flow Diagram :

- These data flows are the inputs and outputs of the DFD. Data objects are represented by labeled arrows, and transformations are represented by circles. 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.
- In DFD there are various levels of DFD, which provide details about the input, processes, and output of a system. Note that the level of detail of process increases with increase in level(s). However, these levels do not describe the system's internal structure or behavior. These levels are listed next :

Data Flow Diagram :

1. **Level 0 DFD** : This shows an overall view of the system. Level a DFD is also known as context diagram.

2. **Level 1 DFD** : This elaborates level a DFD and splits the process into a detailed form.

3. **Level 2 DFD** : This elaborates level 1 DFD and displays the process(s) in a detailed form.

4. **Level 3 DFD** : This elaborates level 2 DFD and displays the process(s) in a detailed form.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Creating a Data Flow model :

- The data flow diagram enables us to develop models of the information domain and functional domain. As the DFD is refined into greater levels of detail, you perform an implicit functional decomposition of the system. The Simple guidelines the derivation of a data flow diagram :-

1. All icons must be labeled with meaningful names.

2. The DFD evolves through a number of levels of details.

3. Always begin with a context level diagram (also called level 0).

- **4**. Always show external entities at level 0 and 1.

5. The level 0 data flow diagram should depict the software/system as a single bubble.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Creating a Data Flow model :

6. Primary input and output should be carefully noted.

7. Refinement should begin by isolating candidate processes, data objects, and data stores to be represented at the next level.

8. All arrows and bubbles should be labeled with meaningful names.

9. Information flow continuity must be maintained from level to level.

10. One bubble at a time should be refined.

Example of Data Flow Diagram :

- The below figure shows the level 0 DFD diagram of food ordering system in restaurant.

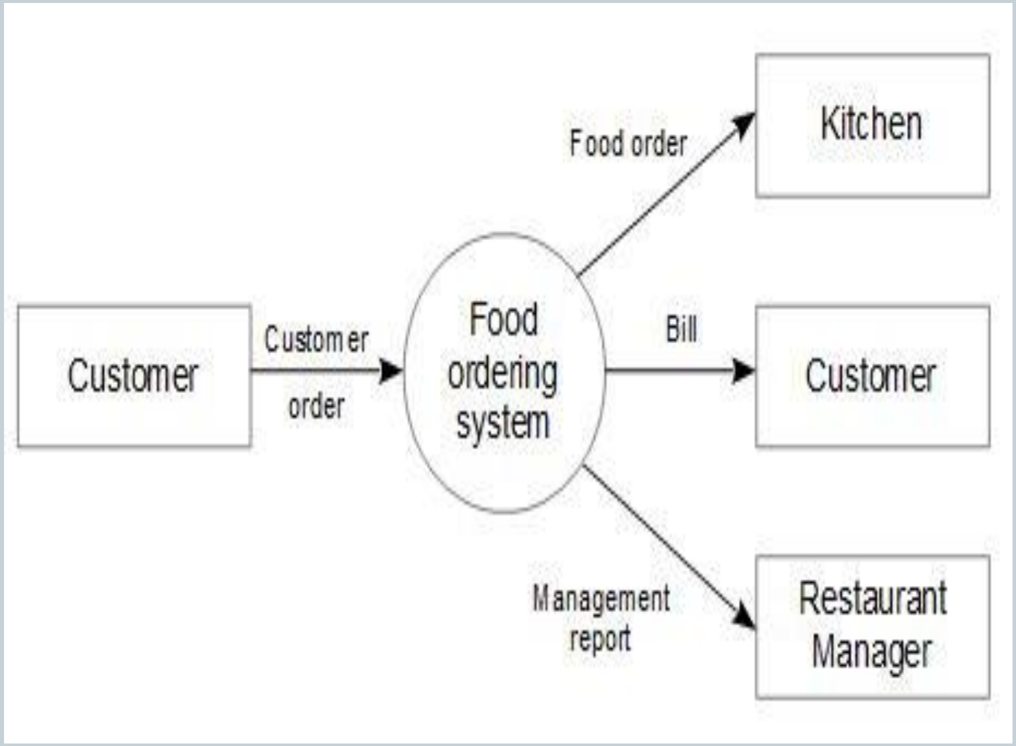
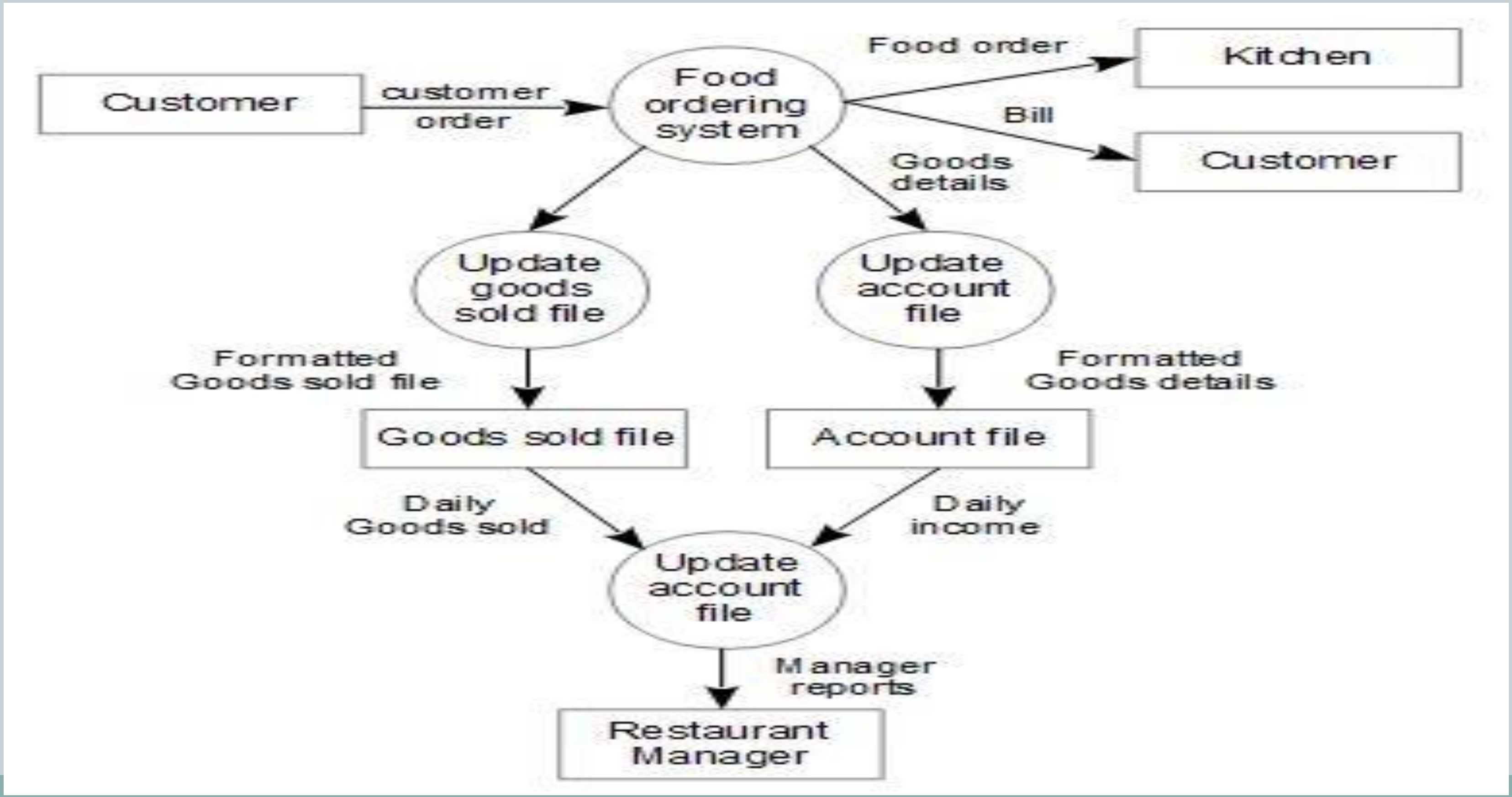


Figure : Level 0 DFD for "food ordering system" in restaurant

Here customer, kitchen and restaurant managers are external entities.
The "food ordering system" accepts the food order from the customer and forwards the order to the kitchen.
When the service is provided to the customer, the system generates the bill.
A copy of customer bill can be submitted to the manager as a part of restaurant management report.

Example of Data Flow Diagram :

- This level 0 DFD can be extended to level 1 DFD to show more details showing exact data flow and processes.



Data Flow Diagram

- **The following observations about DFDs are essential:**
- All names should be unique. This makes it easier to refer to elements in the DFD.
- Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
- Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
- Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Data Flow Diagram

- Control Specifications

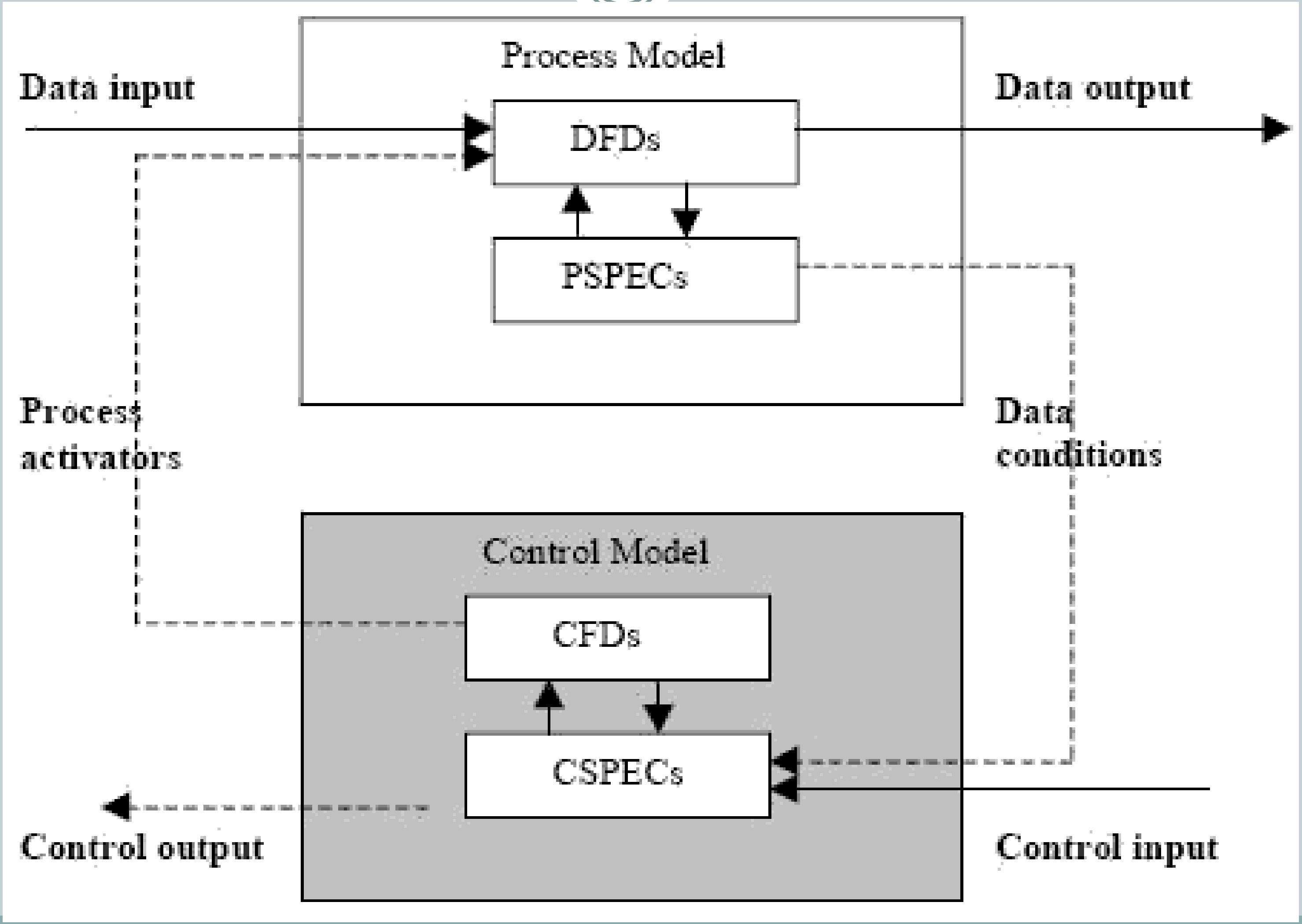
- The Control Specifications (CSPEC) is used to indicate (1) how the software behaves when an event or control signal is sensed and (2) which processes are invoked as a consequence of the occurrence of the event. The control specification (CSPEC) contains a number of important modeling tools. The control specification represents the behavior of the system in two ways. The CSPEC contains a state transition diagram that is sequential specification of behavior. It also contains a process activation table (PAT) -a combinatorial specification of behavior.

Data Flow Diagram

- Process Specifications

- A process specification is a method used to document, analyze and explain the decision-making logic and formulas used to create output data from process input data. Its objective is to flow down and specify regulatory/engineering requirements and procedures. High-quality, consistent data requires clear and complete process specifications.

Data Flow Diagram



DFD Practice examples

Draw a context diagram of DFD for Supermarket App

THANK YOU