Chapter 1- System Analysis and Design

جامعة القادسية/ كلية علوم الحاسوب وتكنولوجيا المعلومات/ قسم نظم المعلومات الحاسوبية/ المرحلة الثانية

❑ Introduction to System Analysis and Design:

- **System Definition:** interrelated components functioning together to achieve an outcome, or It is a group of elements and components that are related and interact with each other and that work together Within specific environmental conditions to achieve a specific goal.
- **System components are:**

– The group of objects (elements) to form the system.

– A set of logical and physical relationships between the components of the system.

– Its system working environment.

❑ Introduction to System Analysis and Design:

- Then System components represents

– Input: all the variables that affect the system and include:

  * Basic Inputs: Which the system depends on in producing the outputs.

  * Resources necessary for the continued operation of the system and for it to perform its functions.

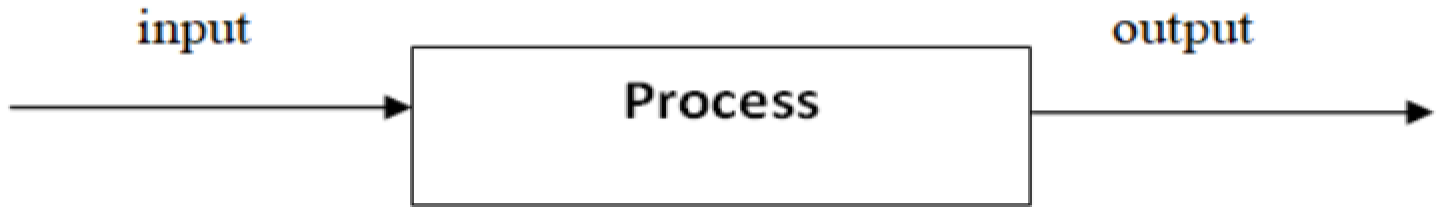
  * Environmental system inputs: the environmental influences that affect the system and are not included in the treatment processes and have an impact External to the behavior of the system and the performance of its operations.

❑ Introduction to System Analysis and Design:

- **Processes:** are the set of interactions between the components of the system, representing the behavior of the internal system.
- **Outputs:** It is what results from the operations of the system and the outputs are related to the goal of the system and include:

  * Key Outputs: What the system primarily produces.

  * Backup output: An output that is entered again to another system or system for reprocessing.



Fig(1): Basic System components

❑ Introduction to System Analysis and Design:

- **Systems Analysis:** activities that enable a person to understand and specify what the new system should accomplish.

- **Importance of System Analysis:**
  Enable the system developer to understand the user's requirements. Without a good analysis of the user's needs and requirements the new system will be inadequate in its solution or erroneous in its implementation.

- **Systems Design:** describes "how" the system will work. It specifies in detail all the components of the system and how they work together to provide the desired solution.

Chapter 2- System Analysis and Design

جامعة القادسية/ كلية علوم الحاسوب وتكنولوجيا المعلومات/ قسم نظم المعلومات الحاسوبية/ المرحلة الثانية

❑Need for System Analysis and Design:

- Installing a system without proper planning leads to great user dissatisfaction and frequently causes the system to fall into disuse.
- Lends structure to the analysis and design of information systems.
- A series of processes systematically undertaken to improve a business through the use of computerized information systems

❑Roles of the System Analyst:

- The analyst plays a key role in information systems development projects.
- Must understand how to apply technology to solve business problems.
- Analyst may serve as change agents who identify the organizational improvement.

❑How can the system analyst starting the analysis?

- Computer systems analysts start their work by asking people what they need their computers to do.
- After analysts understand what the system needs to do, they break down the task into small steps.
- They draw diagrams and charts to show how information will get into the computers, how that information will be processed, and how it will get to the people who need it. For example, analysts might decide how sales information will get into a store's computers and how the computer will add up the information in a way that makes it useful for store managers.
- Analysts experiment with different computer system plans. They try various tools and steps until they find the system that is fastest, easiest, and costs the least.
- Next, analysts decide which computers, software, and tools to buy. They also tell computer programmers how to make any new software that is needed. They give the programmers step-by-step instructions.

- The main job for some systems analysts is getting computers to work together. They connect them into a network. Analysts decide how to get information from one computer to another. Many help people get data from the Internet.

- After planning a system, analysts test it to make sure it works. They check to make sure that information is processed quickly and without mistakes. They also watch to see if the system is easy to use. Often, they have to change their plans to make the systems better.

❑The analyst's approach to problem solving:

- Research and understand the problem.
- Verify that the benefits of solving the problem outweigh the costs.
- Define the requirement for solving the problem.
- Develop a set of solutions (alternatives).
- Decide which solution is best, and make a recommendation.
- Define the details of the chosen solution.
- Implement the solution.
- Monitor to make sure that you obtain the desired results.

❑An analyst should have fundamental technology knowledge of:

- Computers / peripheral devices (hardware).
- Communication networks and connectivity.
- Database and database management systems (DBMS).
- Programming languages (for example: VB.NET or Java).
- Operating systems and utilities.

❑There are four main skills of a system analysts:

- **Analytical Skills** ability to see things as systems, identify, analyze, and solve problems in an optimal way for a specific organization.
- **Technical Skills** ability to understand how computers, data networks, databases, operating systems, etc. work together, as well as their potentials and limitations.
- **Management Skills** include organization’s recourse management, project management (people and money), risk management, and change management.
- **Communication Skills** include effective interpersonal communication (written, verbal, face-to-face conversations, presentations in front of groups), listening.

❑Kinds of technical skills are needed for systems analysts:

- Computers (PCs, mini, mainframes, etc.)
- Computer networks (LAN, WAN, administration, security, etc.)
- Operating systems (UNIX, Windows)
- Data Exchange Protocols (FTP, HTTP, etc.)
- Programming languages (C++, Java, XML, etc.)
- Software applications (Office, project managements, etc.)
- Information systems (databases, MISs, decision support systems)
- System development tools and environments (such as report generators).

❑Kinds of managerial skills are needed for systems analysts:

- Resource management effectively managing the project’s resources, including time, equipment, hardware, software, people, money, etc.
- Project management determining the tasks and resources needed for a project and how they are related to each other,
- Risk management identifying and minimizing risks,
- Change management managing the system’s (organization's) transition from one state to another

❑Kinds of communication skills are needed for systems analysts:

- Clear and effective interpersonal communication, whether written, verbal, from writing reports to face–to–face conversations, to presentations in front of groups.
- Listening (accepting opinions and ideas from other project team members).
- Group facilitation or formal technical reviews (FTR) skills:

a. Setting an agenda.

b. Leading discussions.

c. Involving all parties in the discussion.

d. Summarizing ideas.

e. Keeping discussions on the agenda.

❑Interpersonal and communication skills are crucial to:

- Obtaining information
- Motivating people
- Getting cooperation
- Understanding the complexity and workings of an organization in order to provide necessary support

❑Stakeholders:

- A stakeholder is any person, group, or organization affected by the proposed system or system changes. Or any person that is relate to system by direct or indirect
- Stakeholders include:

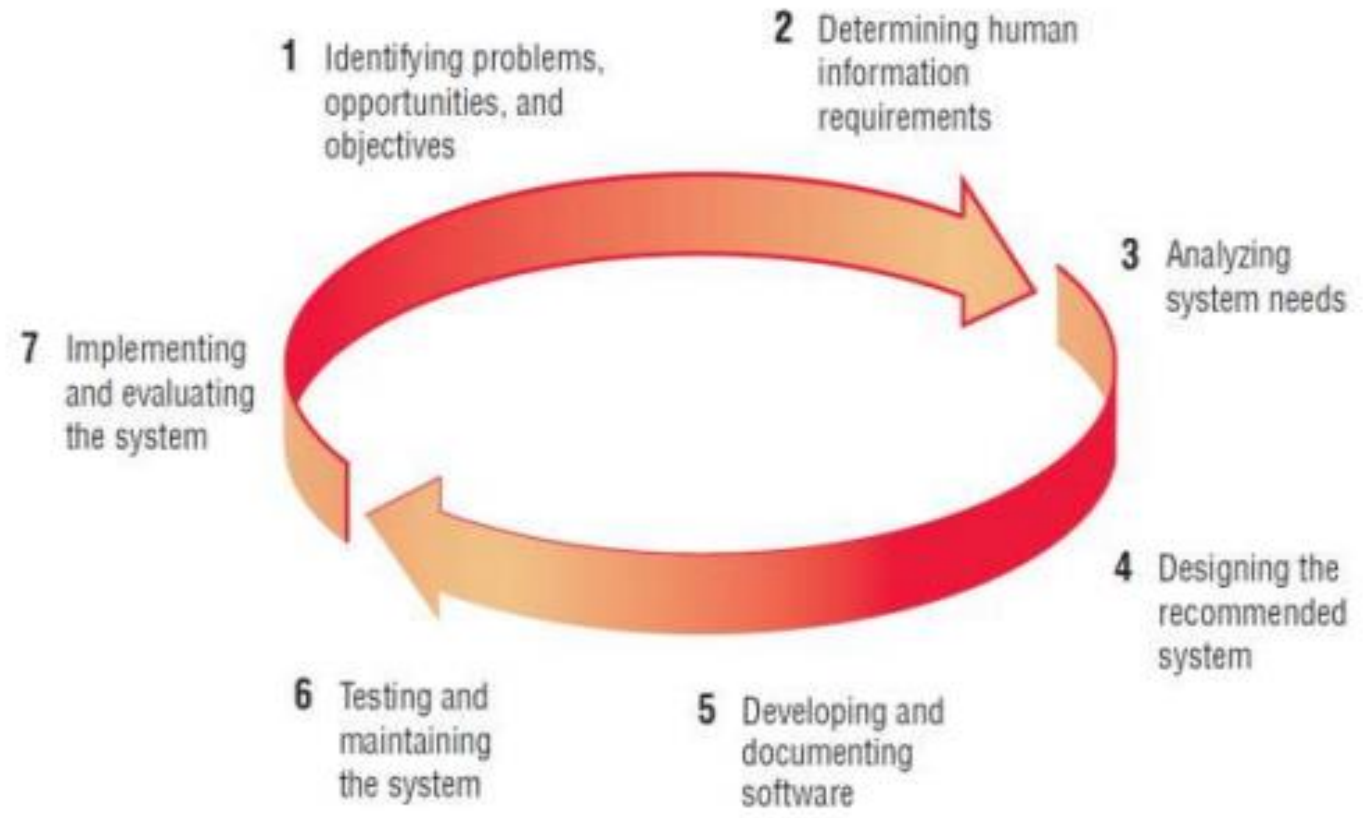
  Project manager, Sales manager, Accounting manager, Customers, Users, Suppliers, Software engineer, consultants…

Chapter 3- System Analysis and Design

جامعة القادسية/ كلية علوم الحاسوب وتكنولوجيا المعلومات/ قسم نظم المعلومات الحاسوبية/ المرحلة الثانية

❑SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC):

- Typically the **SDLC** has 7 steps in the development and improvement of a computer system, may extend or reduce according to the system.


1 Identifying problems, opportunities, and objectives
2 Determining human information requirements
3 Analyzing system needs
4 Designing the recommended system
5 Developing and documenting software
6 Testing and maintaining the system
7 Implementing and evaluating the system

❑IDENTIFYING PROBLEMS, OPPORTUNITIES, AND OBJECTIVES:

- Activity:

- Interviewing user management
- Summarizing the knowledge obtained
- Estimating the scope of the project
- Documenting the results

- Output:

- Feasibility report containing problem definition and objective summaries from which management can make a decision on whether to proceed with the proposed project.

❑DETERMINING REQUIREMENTS:

- Activity:

- Interviewing
- Sampling and investing hard data
- Questionnaires
- Observe the decision maker’s behavior and environment
- Prototyping
- Learn the who, what, where, when, how, and why of the current system

- Output:

- Analyst understands how users accomplish their work when interacting with a computer; and begin to know how to make the new system more useful and usable. The analyst should also know the business functions and have complete information on the people, goals, data and procedure involved.

❑ ANALYZING SYSTEM:

- Activity:

- Create data flow diagrams.
- Complete the data dictionary.
- Analyze the structured decisions made.
- Prepare and present the system proposal.

- Output:

- Recommendation on what, if anything, should be done.

❑DESIGNING THE SYSTEM:

- Activity:

- Design procedures for data entry
- Design the human-computer interface
- Design system controls
- Design files and/or database
- Design backup procedures

- Output:

- Model of the actual system

❑ DEVELOPING AND DOCUMENTING SOFTWARE (Coding):

- Activity:

- System analyst works with programmers to develop any original software
- Works with users to develop effective documentation
- Programmers design, code, and remove syntactical errors from computer programs
- Document software with help files, procedure manuals, and Web sites with Frequently Asked Questions

- Output:

- Computer programs
- System documentation

❑TESTING AND MAINTAINING THE SYSTEM:

- Activity:

- Test the information system
- System maintenance
- Maintenance documentation

- Output:

- Problems, if any.
- Updated programs.
- Documentation.

❑IMPLEMENTING AND EVALUATING THE SYSTEM:

- Activity:

- Train users
- Analyst plans smooth conversion from old system to new system
- Review and evaluate system

- Output:

- Trained personnel
- Installed system

❑ THE IMPACT OF MAINTENANCE (Change):

- Maintenance is performed for two reasons:

- Removing software errors.
- Enhancing existing software.

❑ FEASIBILITY ANALYSES:

- Feasibility Study to decide the cost of the system and payback. There are three types of System Feasibility,

- Technical Feasibility: can we build it?
- Economic Feasibility: should we build it?
- Organizational Feasibility: if we build it, will they come?

❑ TECHNICAL FEASIBILITY:

- Familiarity with application: less familiarity more risk.
- Familiarity with technology: less familiarity generates more risk.
- Project size: large projects have more risk.
- Compatibility: the hard it is so integrate the systems with the company’s existing technology, the higher the risk will be.

❑ ECONOMIC FEASIBILITY:

- Development Costs.
- Annual operating costs.
- Annual benefits (cost saving and revenues).
- Intangible costs and benefits.

❑ ORGANIZATIONAL FEASIBILITY:

- Project manager.
- Senior management.
- Users.
- Other stakeholders.
- Is the project strategically aligned with the business.

Chapter 4- System Analysis and Design

جامعة القادسية/ كلية علوم الحاسوب وتكنولوجيا المعلومات/ قسم نظم المعلومات الحاسوبية/ المرحلة الثانية

❑ Data Flow Diagram (DFD):

- A data flow diagram is a graphical depiction of flow of data through intended software system and is used as 1st step to create an overview of system. It's really useful as it provides overview of data as well as functionality to software designers.

❑ Components of DFD: Entity

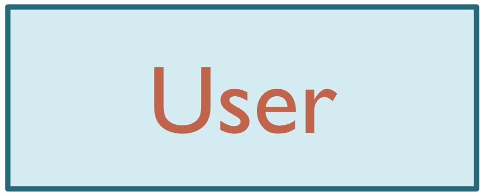
- **External Entities:**

- They could be a person (facebook users), another software(like facebook) or a hardware (sensors) which provide to or consume information from the intended software.

- Represented by rectangle:

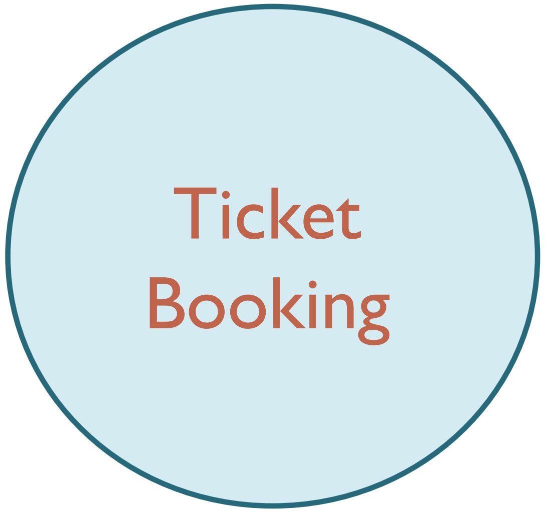
➢ Must be named

➢ No direct data flow between two entities ever.



❑Components of DFD: Process

- A circle (sometimes called a bubble) represents a process or transform that is applied to data and changes it in some way.
- The basic rules:
  - ➢ It must be properly labeled
  - ➢ It must not be repeated in a diagram



❑Components of DFD: Data Flow

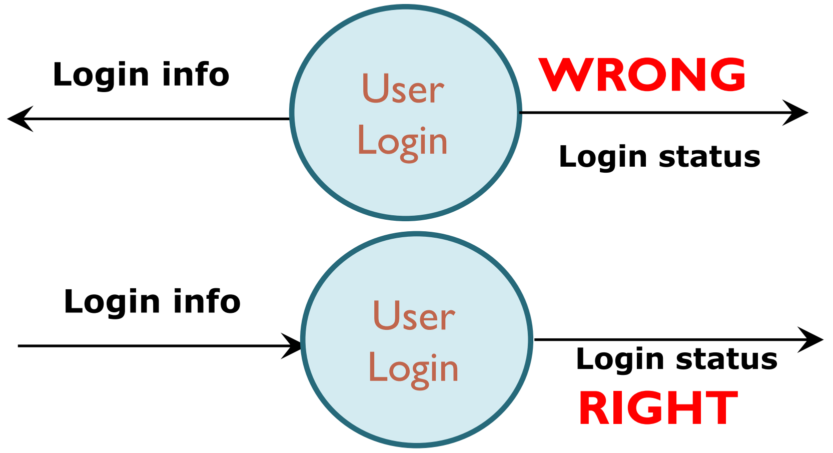
- **The basic rules:**

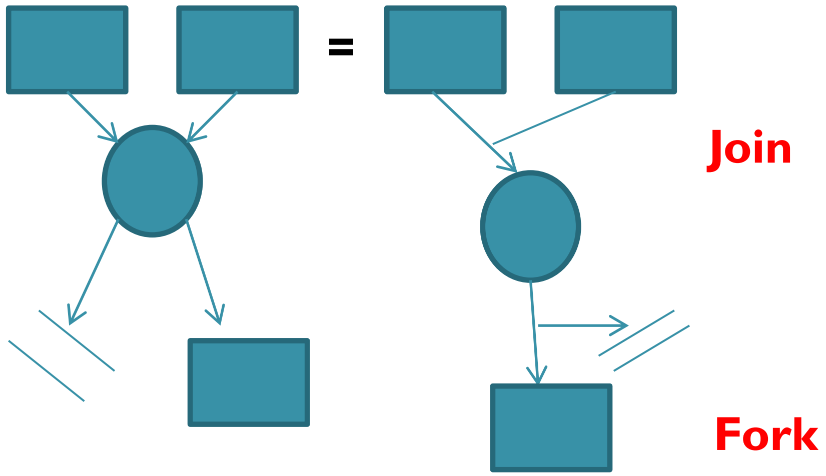
  - Data flows can't be bidirectional, i.e the input data flow and the output data flow for a process, data store or for an entity should always be different.

  - The data flows should always be labeled

  - The labels should be precise and informative

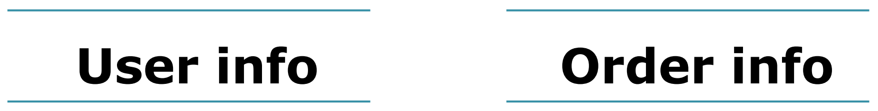
  - You can join two similar input data flows(join) or two similar output data flows (fork)





❑ Components of DFD: Data Store

- Data stores are places where data may be stored. This information may be stored either temporarily or permanently by the user.
- They are internal to the system.
- The basic rules:
  - ➢ Never shown in context level diagram
  - ➢ No direct data flows between two data sources
  - ➢ Symbol:



❑ DFD General rules:

- **Basic rules that apply to all DFDs:**

- No internal logic should be shown like loops, if-else, this is not a flow chart

-In order to keep the diagram uncluttered, you can repeat data stores and external entities.

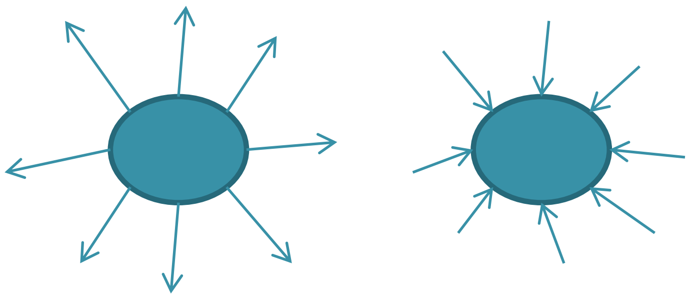
-No process can have only output data flows (a miracle).

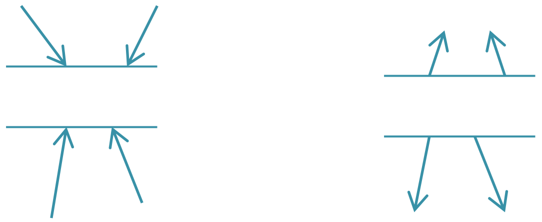
-No process can have only input data flows (black hole).

-Data cannot be moved directly from one store to another without a process.

-Data cannot move directly from an external entity to a data store without a process.

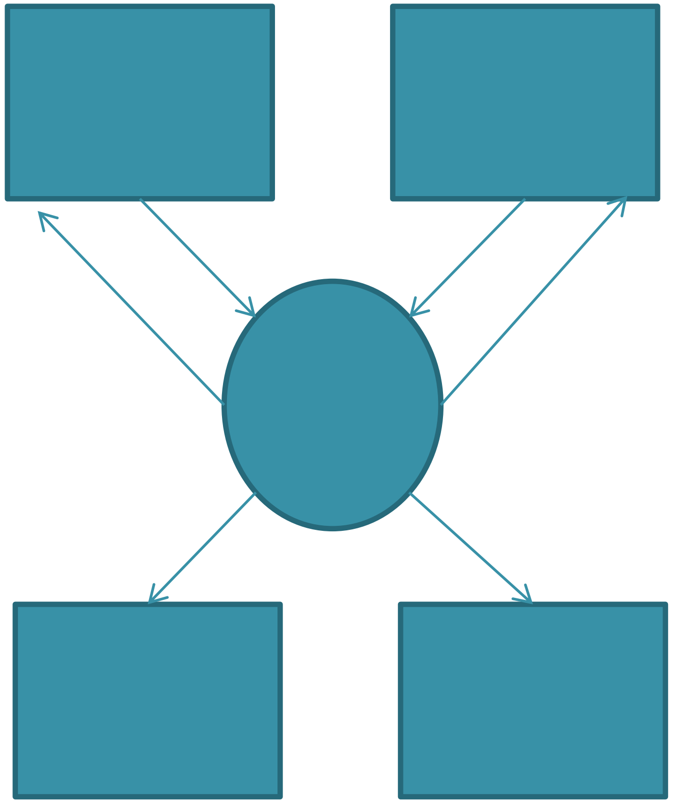
-Data stores can't be sink(only input data flows) or source (only output data flows) in level 1 DFD





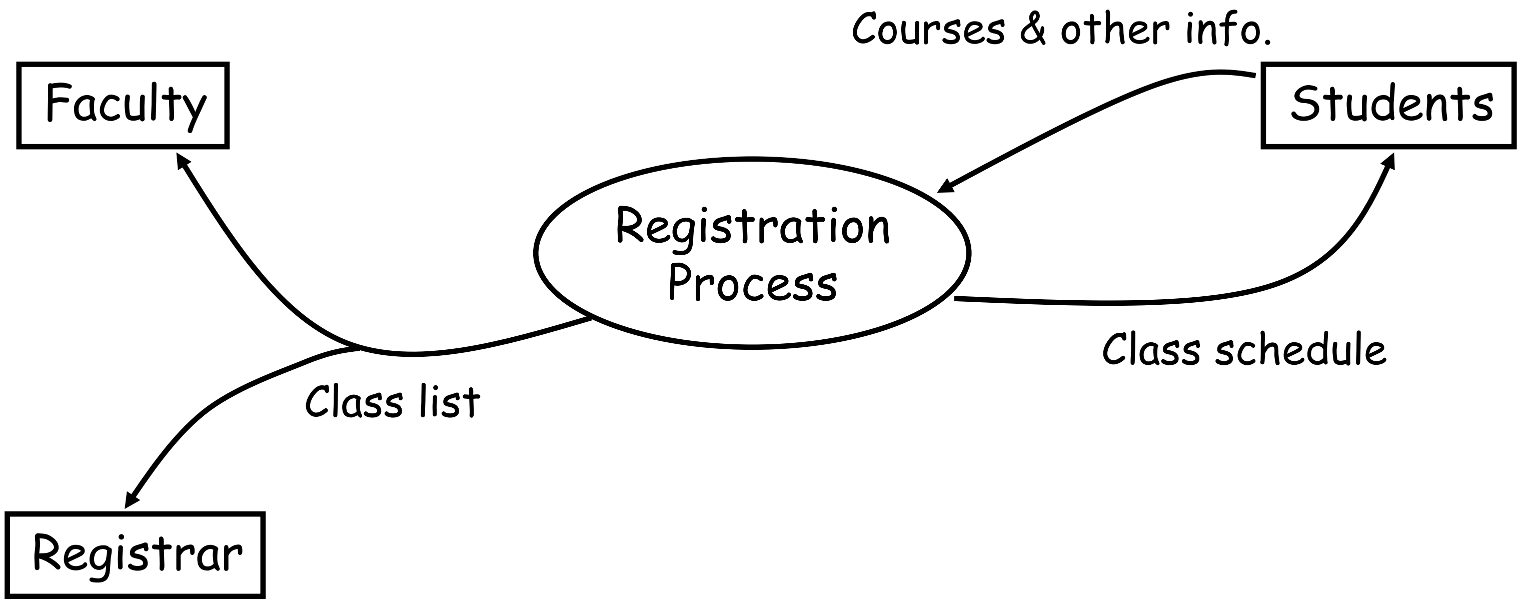
❑ Context Level Diagram – Level 0

- **A level 0 DFD, also called a fundamental system model or a context model.**
- **It represents the entire software element as a single process with input and output data.**
- **All the external entities should be identified and shown.**
- **Rule:**
  - Only one process
  - Data flows should be labeled.
  - No data store can be shown in context diagram as they are internal to the system only.



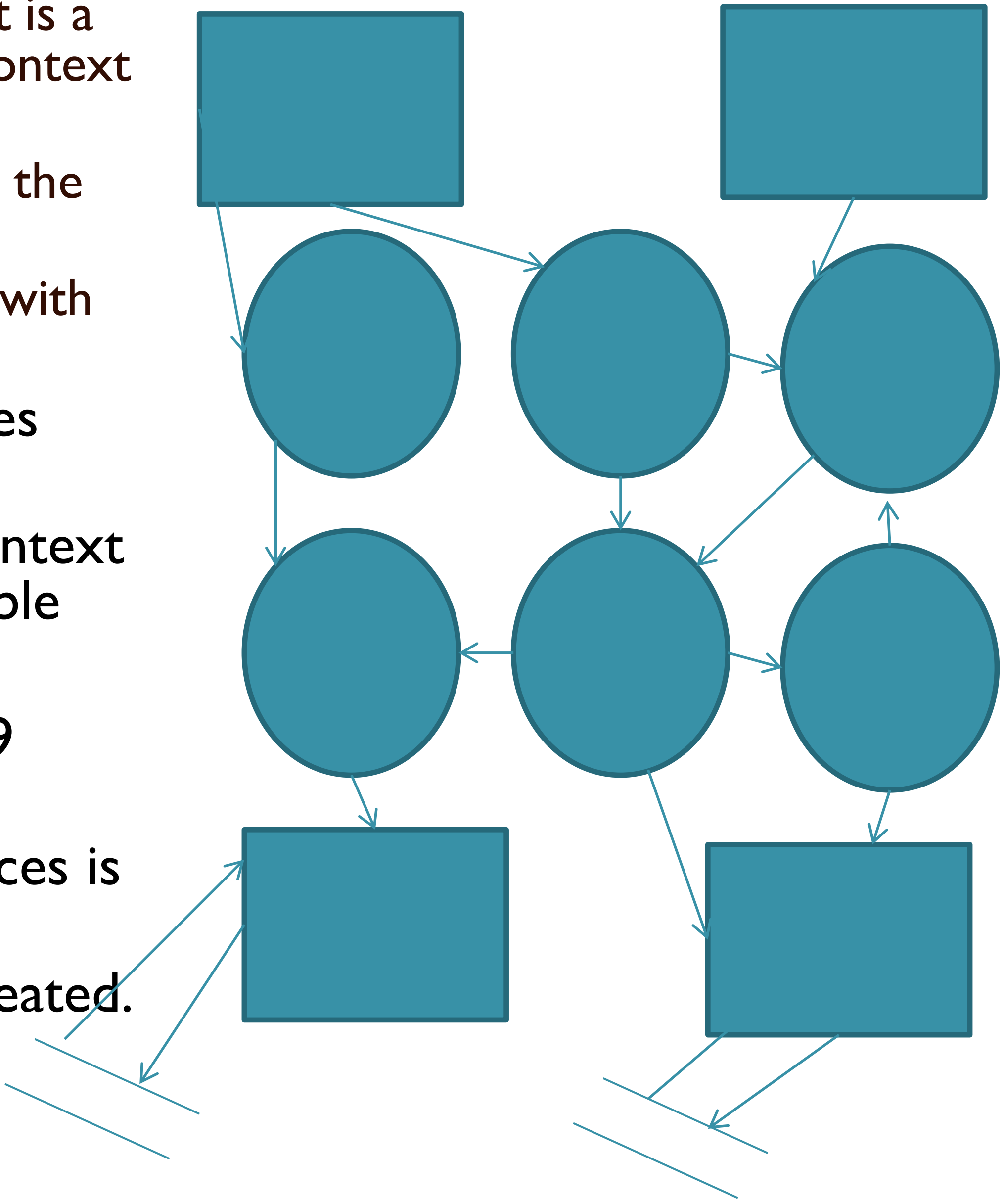
❑ Course Registration System:

- **Context Diagram for Course Registration System**



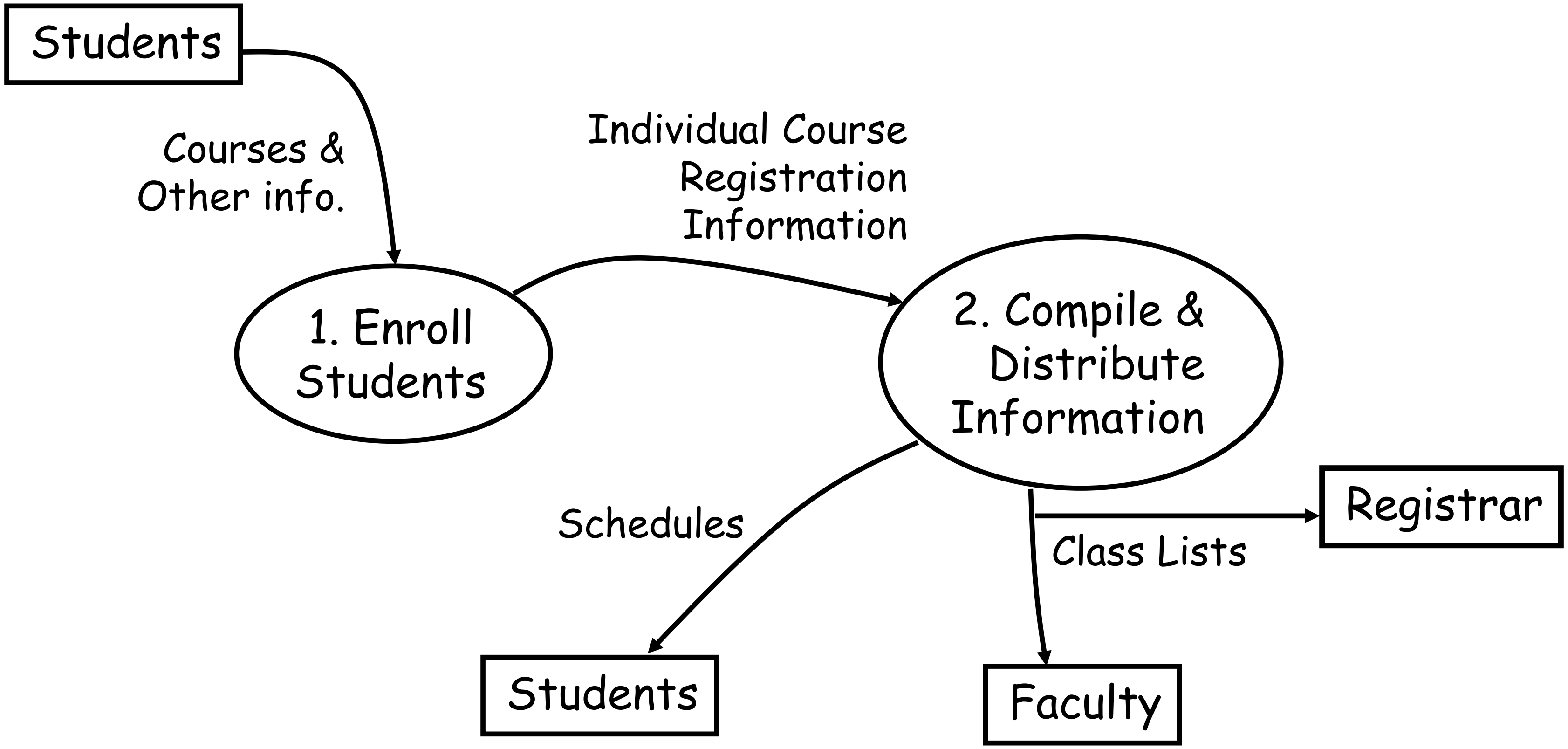
❑ Level 1 DFD :

- The level I DFD we construct is a more refined version of the context diagram.
- It covers the entire system, all the main processes are shown
- The DFD should be balanced with respect to context diagram
  - ➢ No new external entities should be there
  - ➢ The data flows from context diagrams should be visible
- Rules:
  - ➢ It should consists of 5-9 processes(bubbles).
  - ➢ Repetition of data sources is allowed.
  - ➢ Process can not be repeated.



❑ DFD: Course Registration System

- **Level 1 DFD:**



- Note: External entity Students is replicated to avoid crossing lines..

❑ DFD: Course Registration System

- **Level 2 DFD:**

- Only those processes that merit being expanded need to have level 2 DFDs .
- Level 2 DFD completely describes any one process from the level 1 DFD.
- Rules:

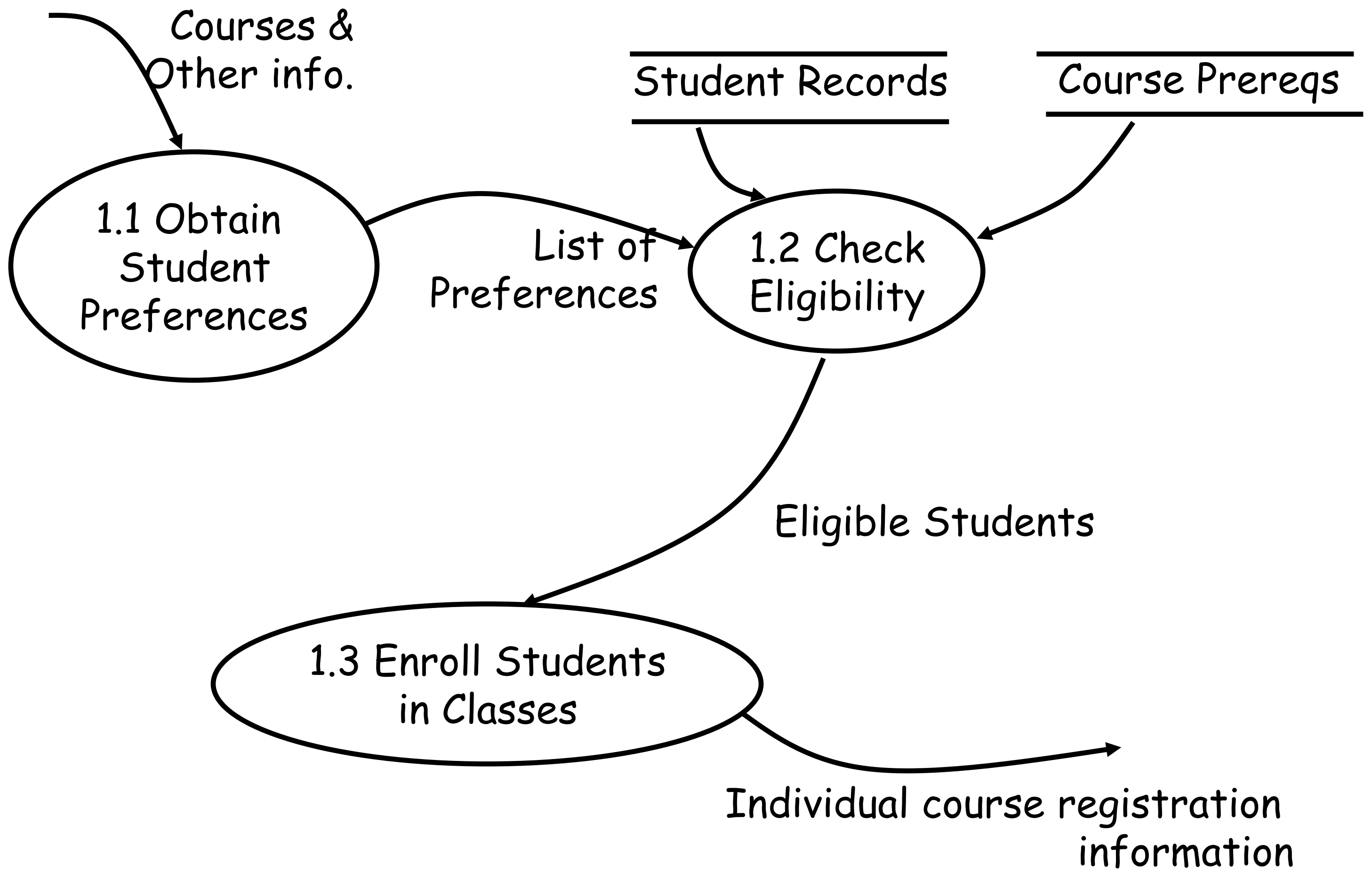
  -All the data flows into and out of selected process on the level 1 DFD also appear on the level 2 DFD

  - Repetition of data sources is allowed.

  - A Data store can appear as a sink or source within level 2 DFD

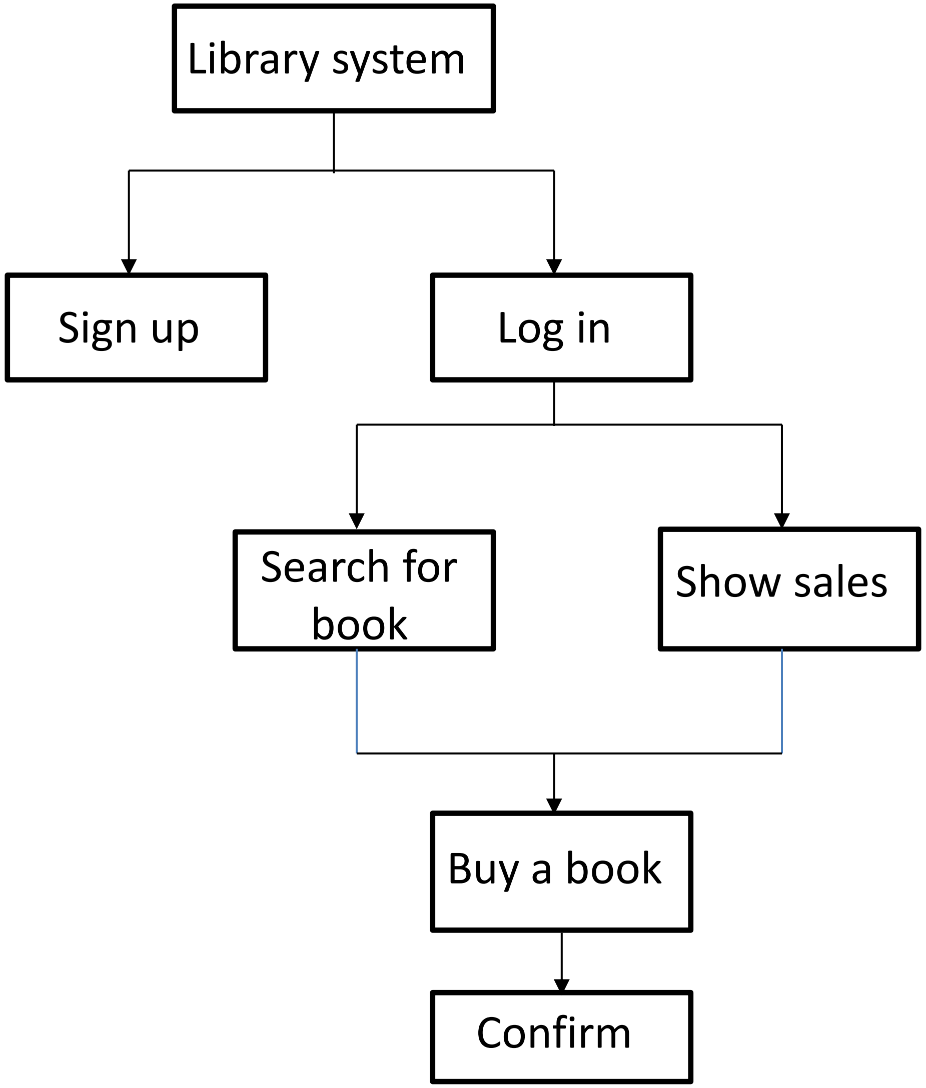
❑ DFD: Course Registration System:

- **Level 2 DFD:**

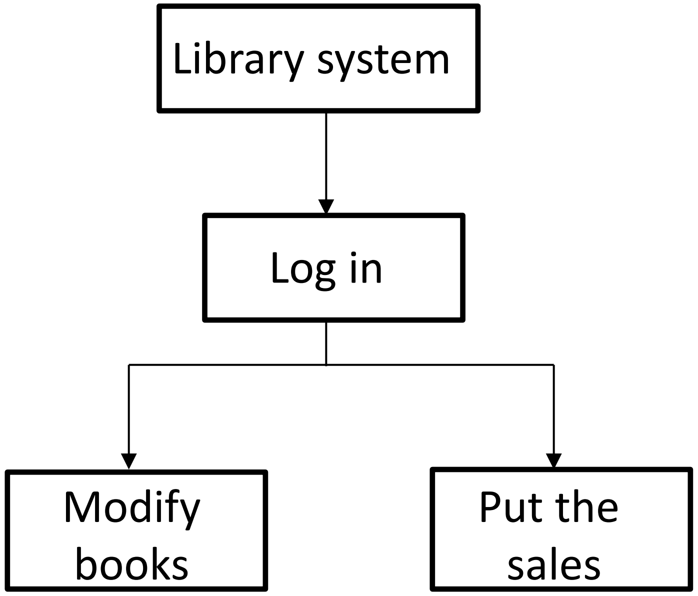


Example on Data Flow Diagram Online Library System

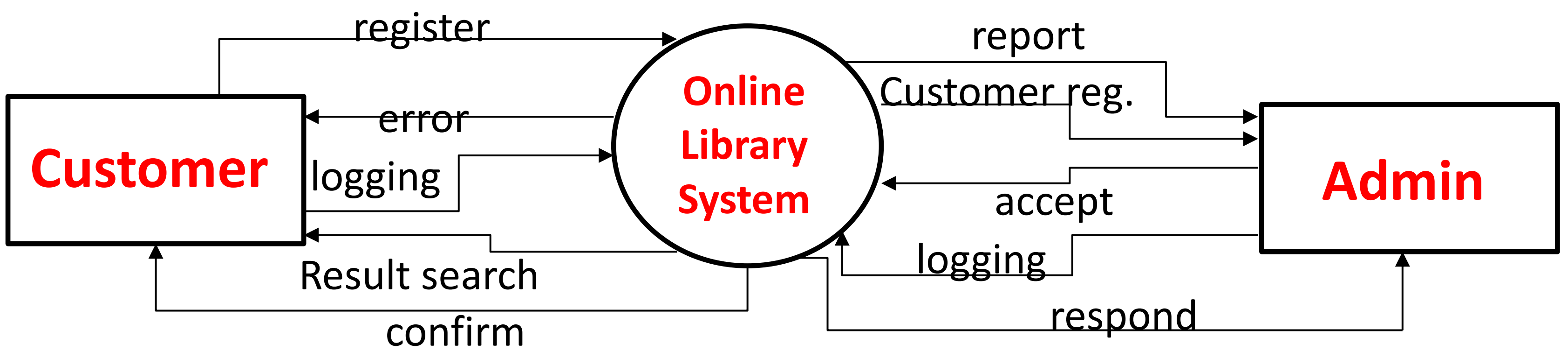
The Structure of Online Library System



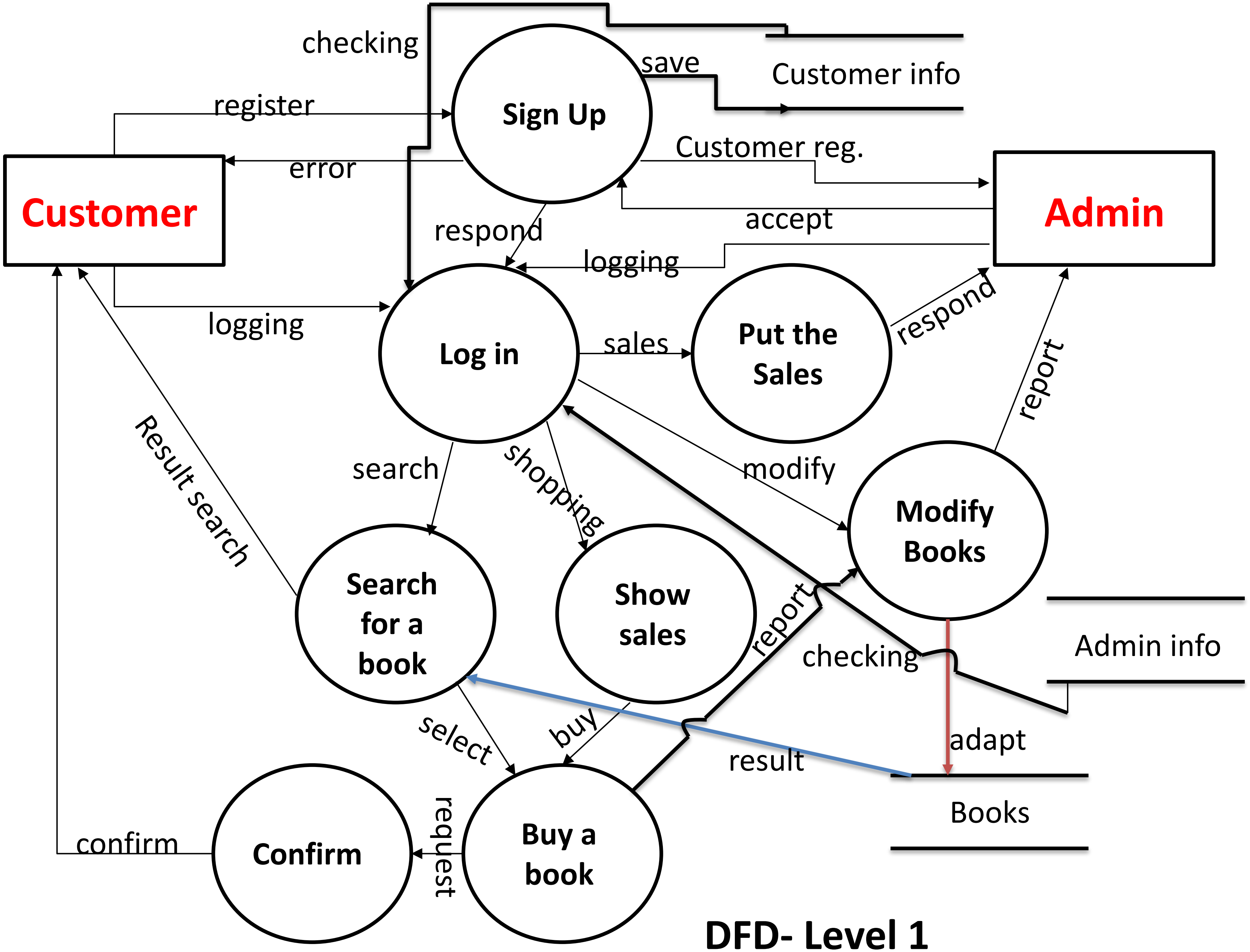
Customer



Admin



Context Diagram- 0

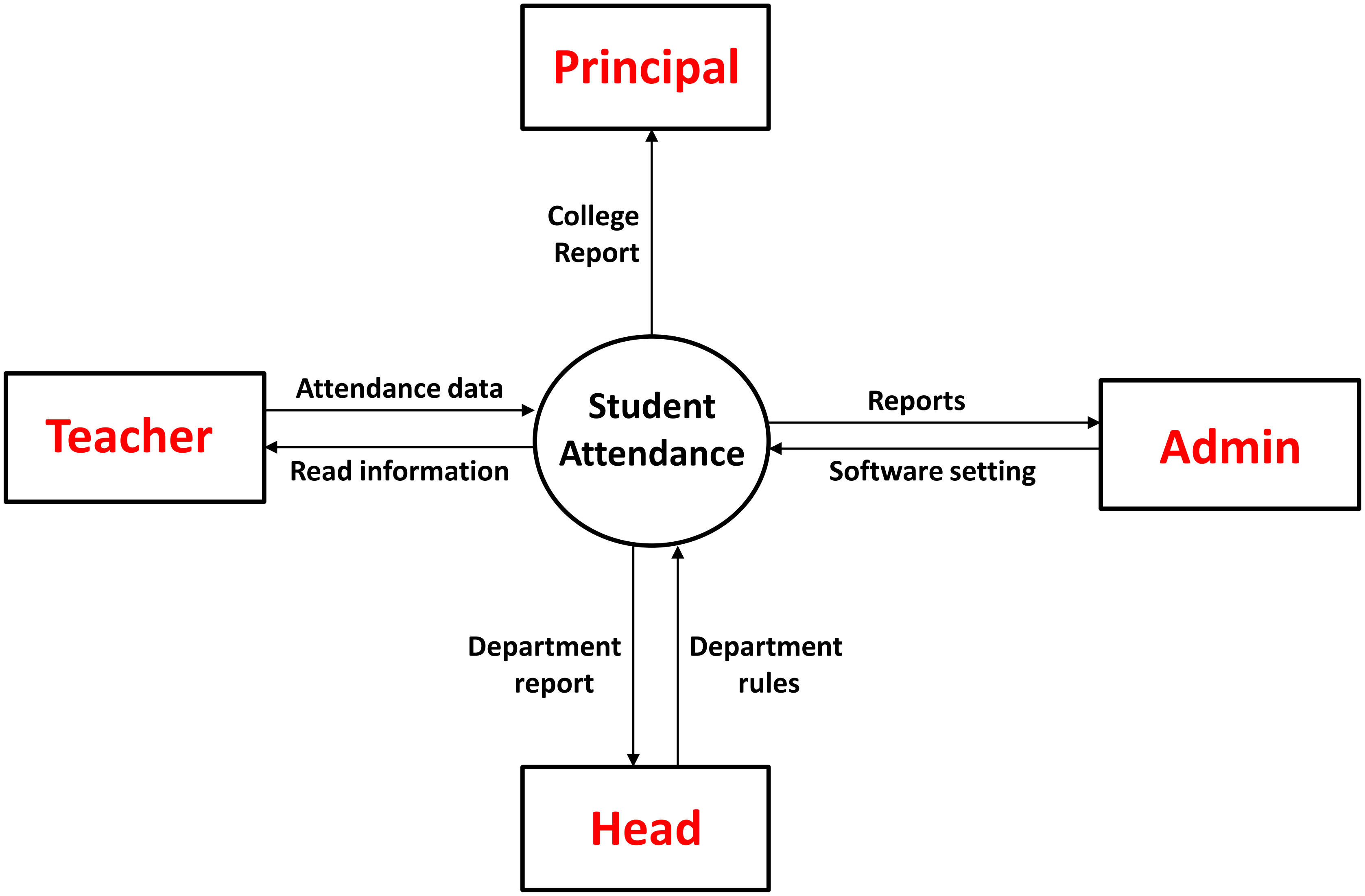


checking
register
Sign Up
save
Customer info
Customer reg.
error
Customer
Admin
accept
respond
logging
logging
Log in
sales
Put the Sales
respond
report
Result search
search
shopping
modify
Modify Books
Search for a book
Show sales
report
checking
Admin info
select
buy
result
adapt
Books
confirm
Confirm
request
Buy a book

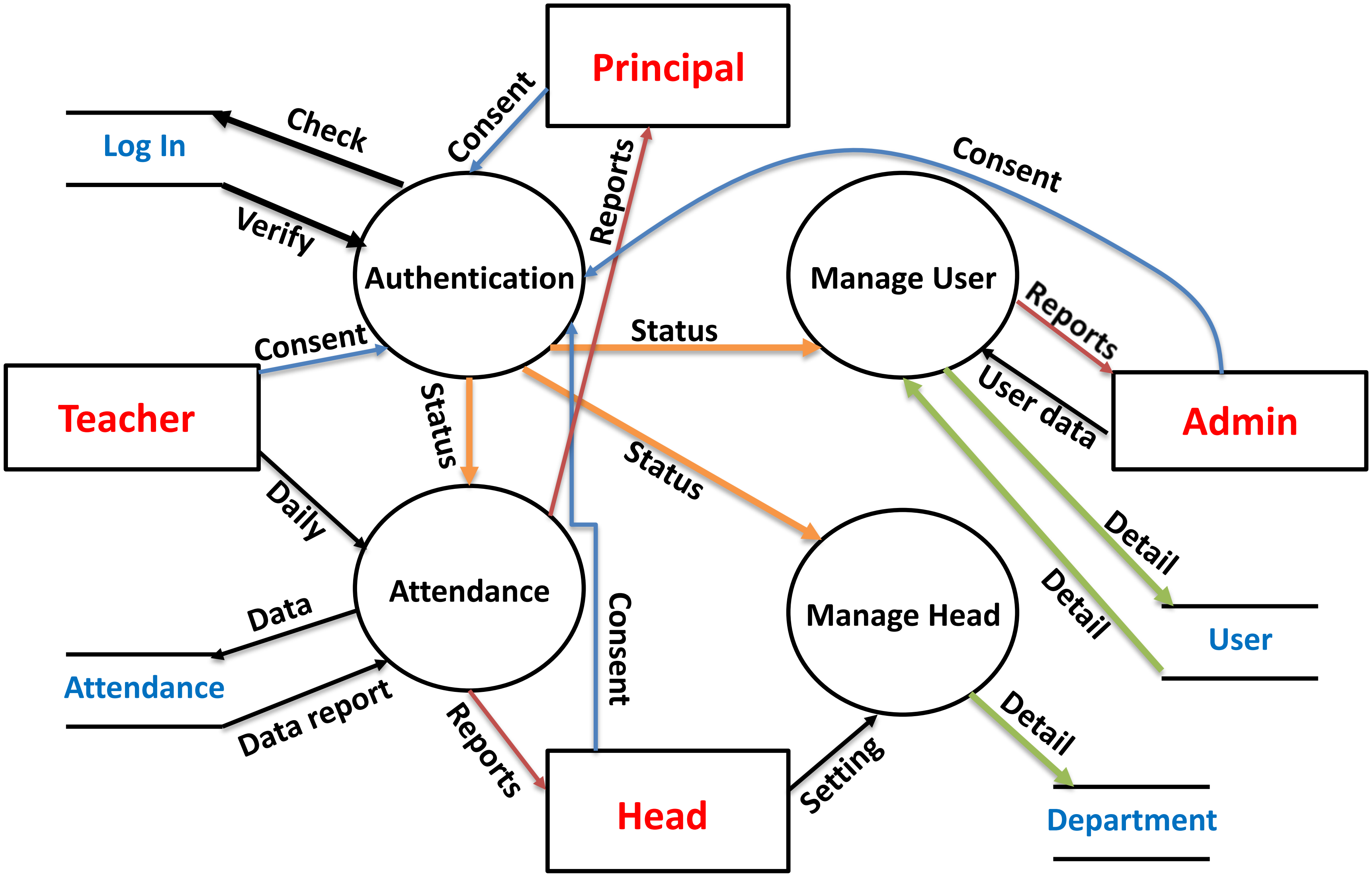
DFD- Level 1

Example on Data Flow Diagram

Student Attendance System



Context Diagram 0


Principal
Log In
Check
Consent
Verify
Authentication
Reports
Consent
Manage User
Status
Consent
Teacher
Status
Reports
User data
Admin
Status
Daily
Attendance
Detail
Detail
Manage Head
User
Data
Consent
Attendance
Data report
Reports
Detail
Setting
Head
Department

Level-1

Chapter 5- Introduction to System Analysis and Design

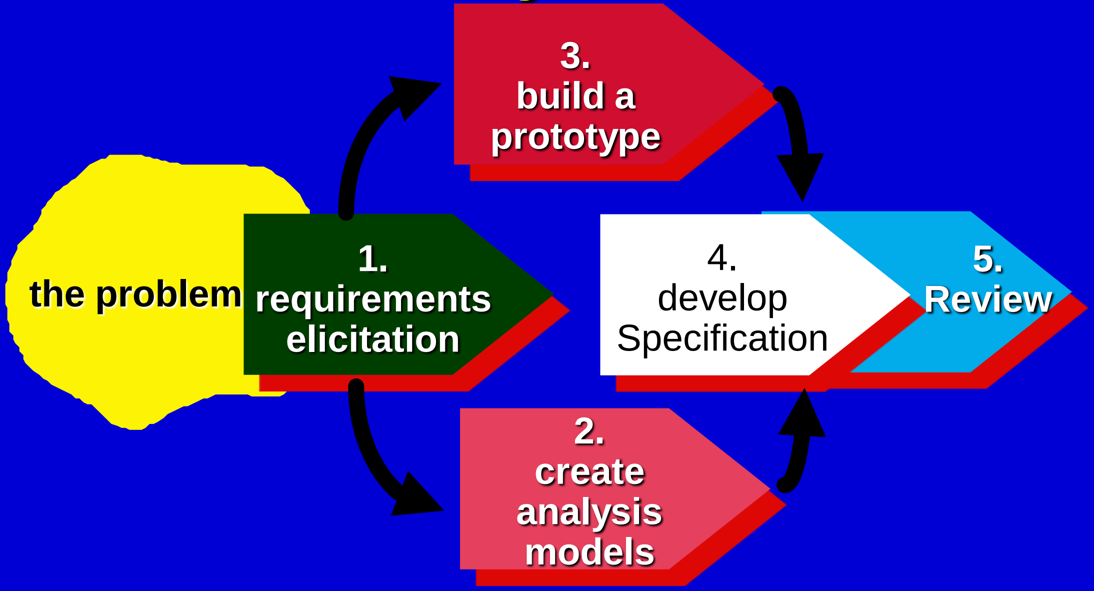
جامعة القادسية/ كلية علوم الحاسوب وتكنولوجيا المعلومات/

قسم نظم المعلومات الحاسوبية/ المرحلة الثانية

Requirements Analysis

- Five areas of effort
  - Problem recognition
  - Evaluation and solution synthesis
  - Modeling
  - Specification
  - Review

The Analysis Process



1. Requirements Elicitation

Context-Free Questions

- Focuses on the customer, the overall goals, and benefits:
  - Who is behind the request for this work?
  - Who will use the solution?
  - What will be the economic benefit of a successful solution?
  - Is there another source for the solution that you need?

- **Enable the analyst to gain a better understanding of the problem and the customer to voice his or her perceptions about a solution:**
  - **How would you characterize “good output that would be generated by a successful solution?**
  - **What problem(s) will this solution address?**
  - **Can you show me the environment in which the solution will be used?**
  - **Will special performance issues or constraints affect the way the solution is approached?**

- Focuses on the effectiveness of the meeting:
  - Are you the right person to answer these questions? Are your answers “official”?
  - Are my questions relevant to the problem that you have?
  - Am I asking too many questions?
  - Can anyone else provide additional information?
  - Should I be asking you anything else?

Facilitated Application Specification Techniques (FAST)

- A meeting is conducted at a neutral site and attended by both software engineers and customers.
- Rules for preparation and participation are established.
- An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas.
- A “facilitator” controls the meeting.
- A “definition mechanism” is used
- ….

Quantity Function Deployment (QFD)

- QFD is a quality management technique that translates the <u>needs of the customer</u> into <u>technical requirements</u> for software.

- Three types of requirements
  - Normal requirement
  - Expected requirement
  - Exciting requirement

Use-Cases

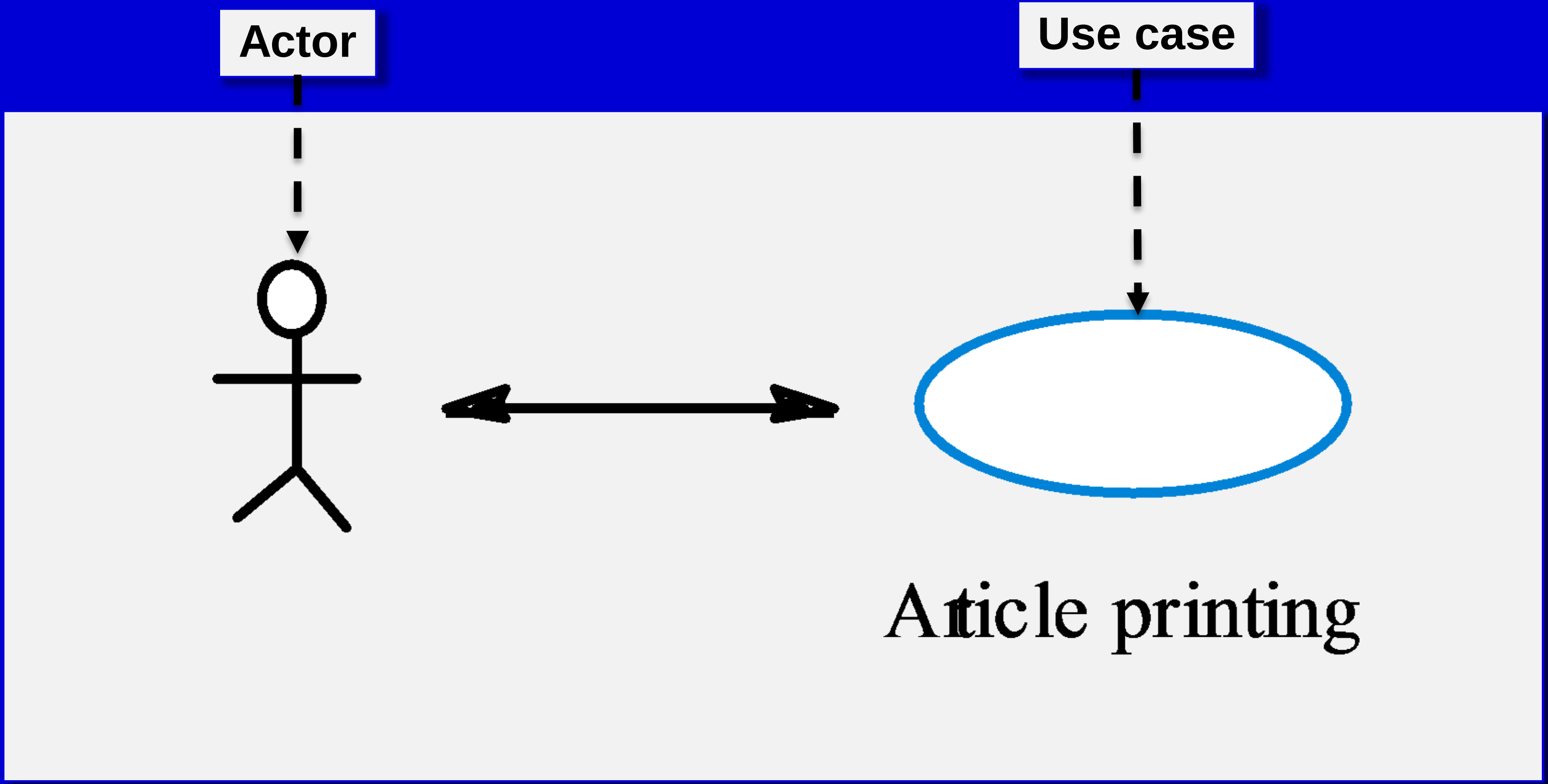
- A collection of scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an "actor"—a person or device that interacts with the software in some way
- Each scenario answers the following questions:
  - What are the main tasks of functions performed by the actor?
  - What system information will the actor acquire, produce or change?
  - Will the actor inform the system about environmental changes?
  - What information does the actor require of the system?
  - Does the actor wish to be informed about unexpected changes

Use Cases

- The details of each use case should also be documented by a use case description: E.g.,
  - Print receipt – A customer has paid for an item via a valid payment method. The till should print a receipt indicating the current date and time, the price, the payment type and the member of staff who dealt with the sale.
    - [Alternate Case] – No print paper available – Print out “Please enter new till paper” to the cashier’s terminal. Try to print again after 10 seconds.

An alternate case here is something that could potentially go wrong and denotes a different course of action.

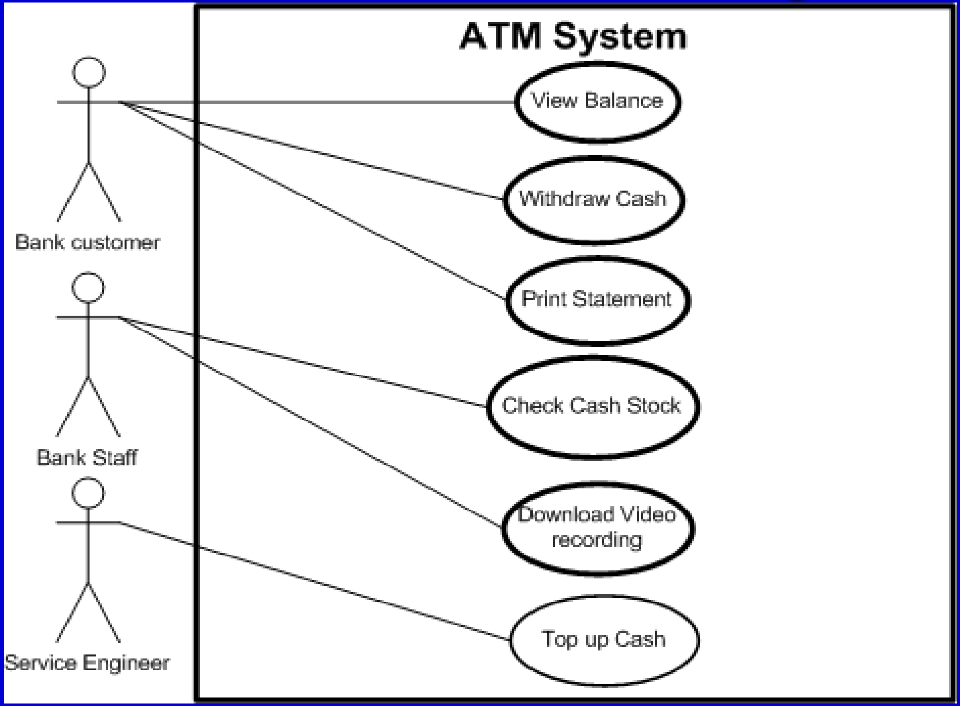
Example - Article Printing Use-Case



ATM machine

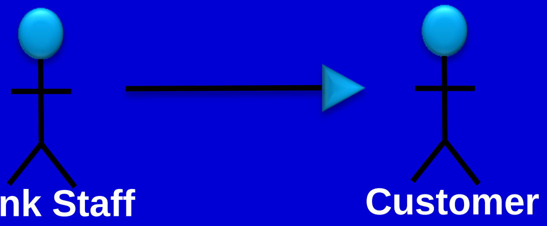
- Actors
  - Customers
  - Bank staff
  - ATM service engineer
- Use cases
  - Withdraw cash
  - Check balance
  - Add cash to machine

Example - ATM Use Case Diagram



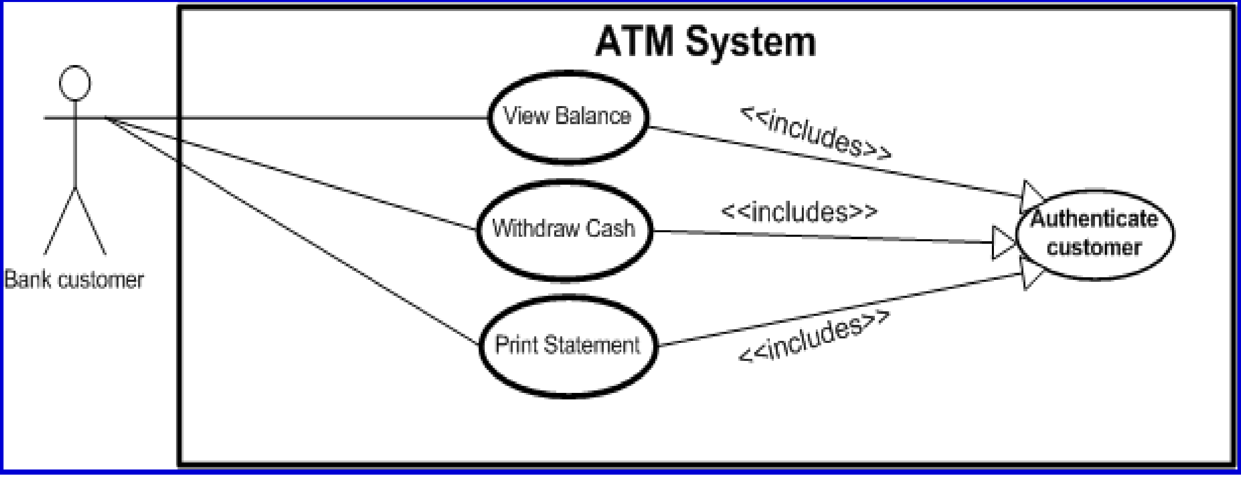
Advanced Use Case Diagrams

- We can draw a box (with a label) around a set of use cases to denote the system boundary, as on the previous slide (“library system”).
- Inheritance can be used between actors to show that all use cases of one actor are available to the other:



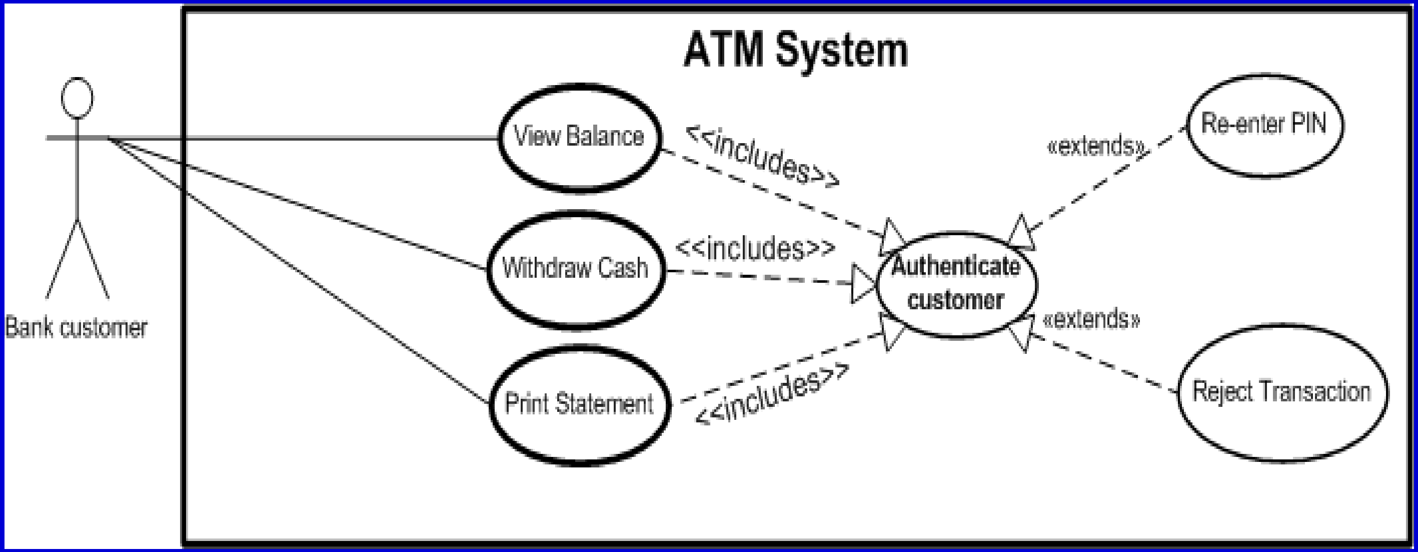
Include Relations

- **If several use cases include, as part of their functionality, another use case, we have a special way to show this in a use-case diagram with an <<include>> relation.**



Extend Relations

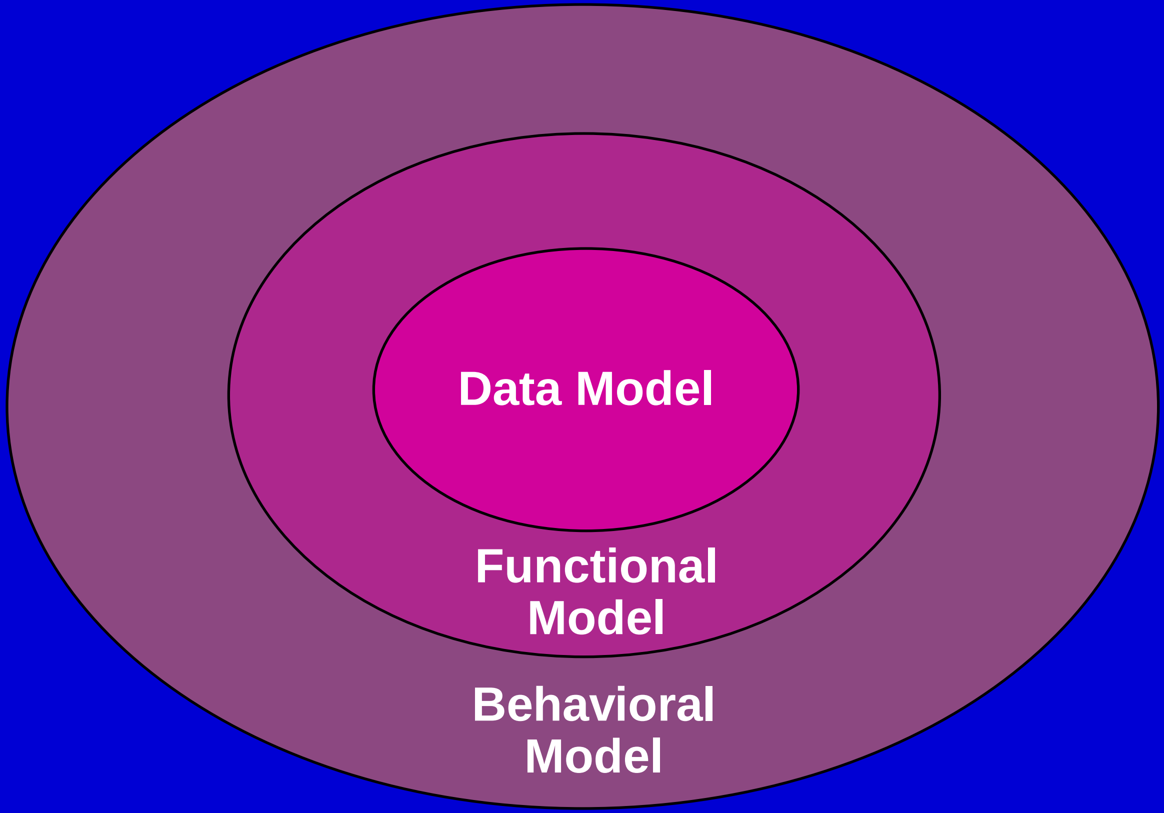
- If a use-case has two or more significantly different outcomes, we can show this by extending the use case to a main use case and one or more subsidiary cases.



In summary

- Include
  - When the other use case is always part of the main use case
- Extend
  - When the other use case, sometime is needed

2. Analysis Model



Analysis Principles

1. The information domain of a problem must be represented and understood. (Encompasses all data objects that contain numbers, text, images, audio, or video. Information content or data model (shows the relationships among the data and control objects that make up the system), Information flow (represents the manner in which data and control objects change as each moves through the system), Information structure (representations of the internal organizations of various data and control items).

2. The functions that the software is to perform must be defined.

3. The behavior of the software must be represented.

4. The models that depict information, function, and behavior must be partitioned in a manner that uncovers detail in a layered fashion.

5. The analysis process should move from essential information toward implementation detail.

Analysis Guiding Principle

- *Understand the problem* **before you begin to create the analysis model***.*
- *Develop prototypes* **that enable a user to understand how human/machine interaction will occur.**
- *Record the origin* **of and** *the reason* **for ever requirement.**
- *Use multiple views* **of requirements***.*
- *Rank* **requirements.**
- *Work to eliminate ambiguity.*

Analysis Principle I
Model the Data Domain

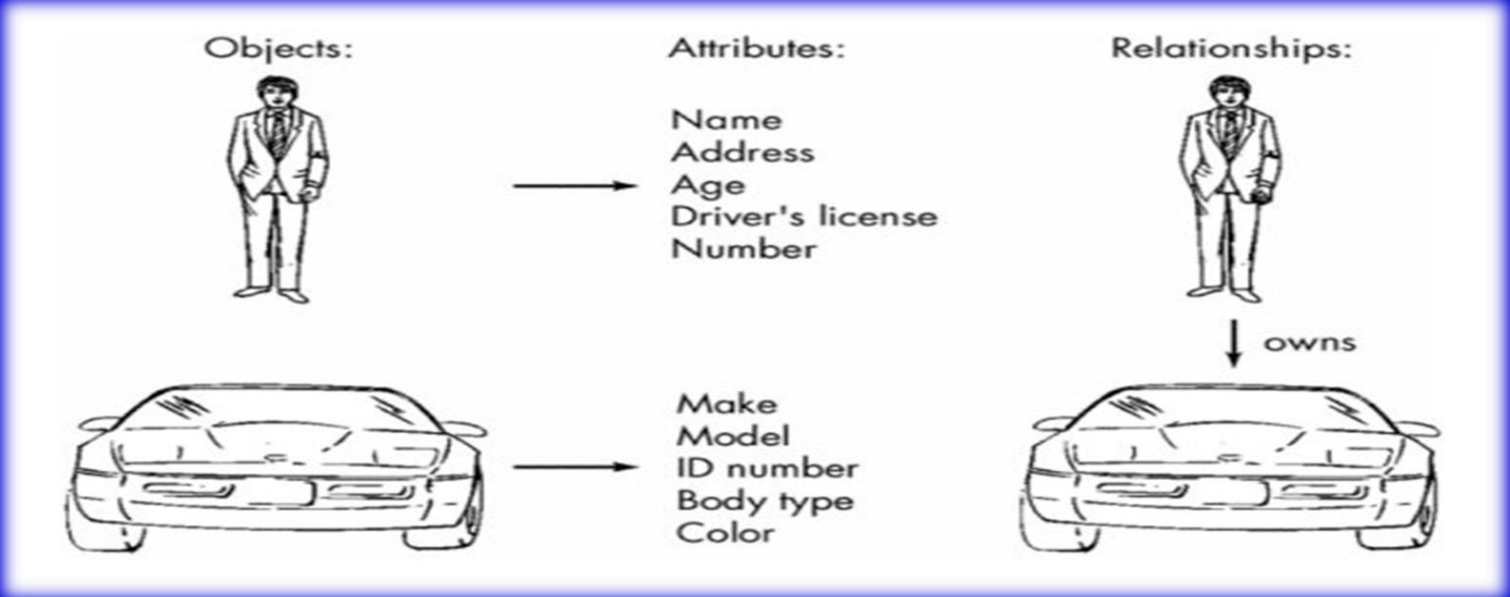
- ❑ Define data objects
- ❑ Describe data attributes
- ❑ Establish data relationships

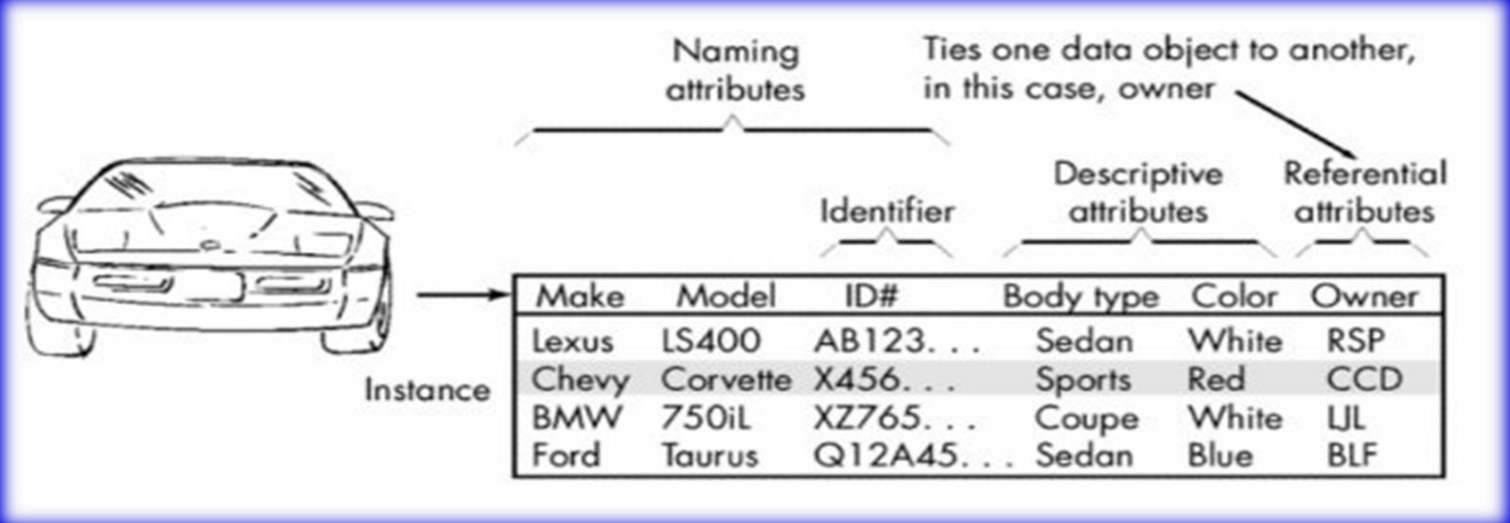
Data Modeling:

➢ **Entity Relationship Diagram - (ERD)**

❑ **Questions Answered**
- What are the primary data objects to be processed by the system?
- What is the composition of each data object and what attributes describe the object?
- Where do the the objects currently reside?
- What are the relationships between each object and other objects?
- What are the relationships between the objects and the processes that transform them?

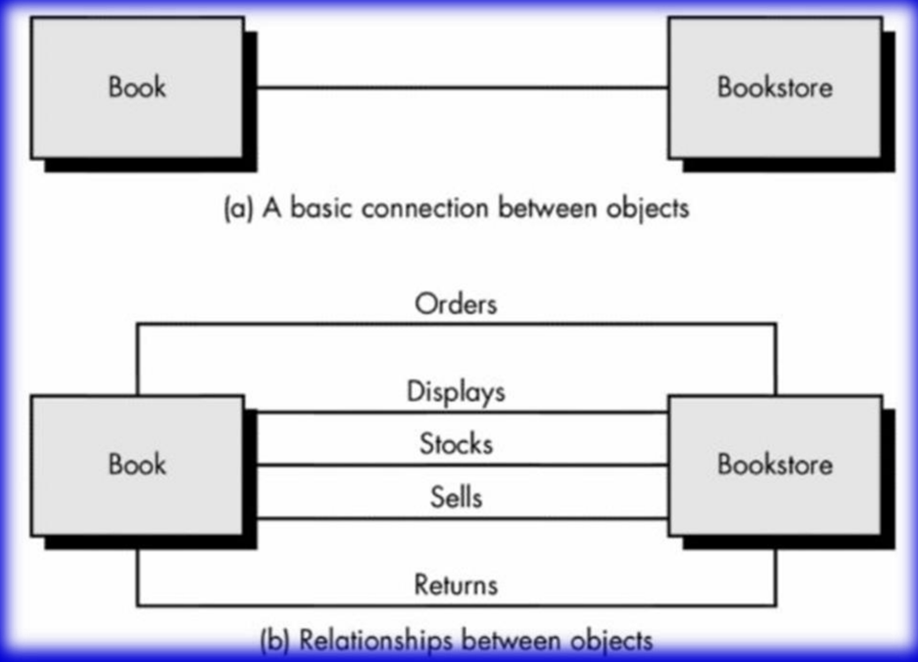
❑ **Components**
- Data object - any person, organization, device, or software product that produces or consumes information, e.g. report, event, place, structure
- Attributes - name a data object instance, describe its characteristics, or make reference to another data object





Make	Model	ID#	Body type	Color	Owner
Lexus	LS400	AB123. . .	Sedan	White	RSP
Chevy	Corvette	X456. . .	Sports	Red	CCD
BMW	750iL	XZ765. . .	Coupe	White	LJL
Ford	Taurus	Q12A45. . .	Sedan	Blue	BLF

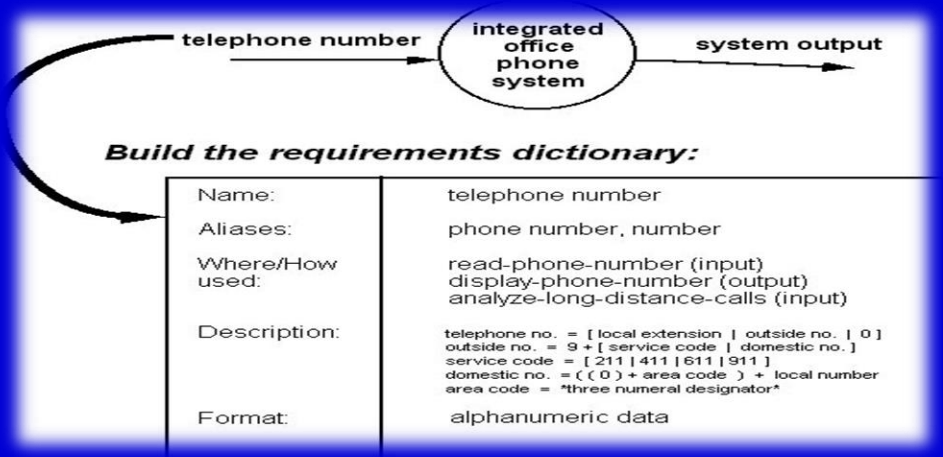
- **Relationships - indicate the manner in which data objects are connected to one another**



(a) A basic connection between objects

(b) Relationships between objects

- **Data Dictionary –(DD): indicate the description of all data objects.**



Build the requirements dictionary:

Name:	telephone number
Aliases:	phone number, number
Where/How used:	read-phone-number (input) display-phone-number (output) analyze-long-distance-calls (input)
Description:	telephone no. = [local extension | outside no. | 0] outside no. = 9 + [service code | domestic no.] service code = [211 | 411 | 611 | 911] domestic no. = ((0) + area code) + local number area code = *three numeral designator*
Format:	alphanumeric data

Analysis Principle II
Model Function

- ❑ Identify functions that transform data objects
- ❑ Indicate how data flow through the system
- ❑ Represent producers and consumers of data

Analysis Principle III
Model Behavior

- ❑ Indicate different states of the system
- ❑ Specify events that cause the system to change state

❑ Behavioral Modeling:

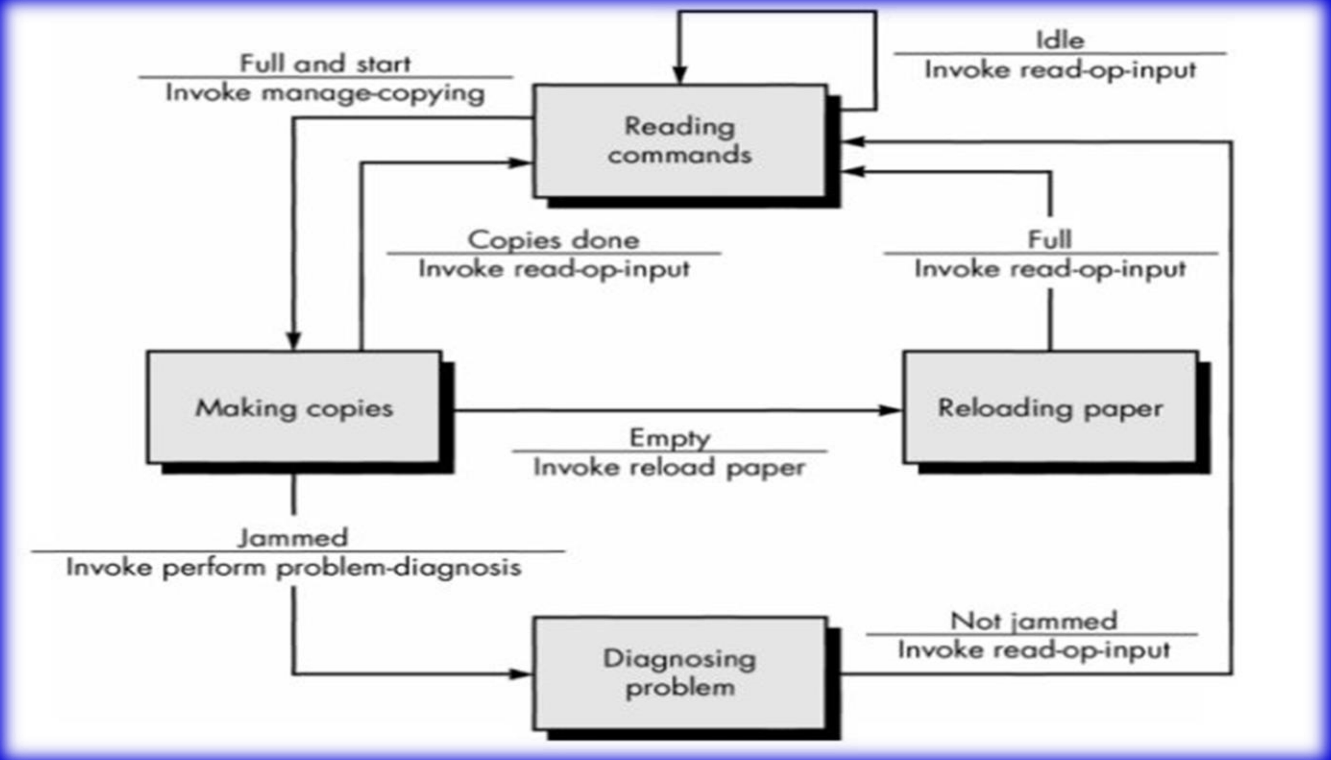
- The behavioral model indicates how software will respond to external events to create the model, the analyst must perform the following steps:
- Evaluate all use-cases to fully understand the sequence of interaction within the system.
- Identify events that drive the interaction sequence and understand how these events relate to specific objects.
- Create a sequence for each use-case.
- Build a state diagram for the system.
- Review the behavioral model to verify accuracy and consistency.

In the context of behavioral modeling, two different characterizations of states must be considered:

the state of each class as the system performs its function and, the state of the system as observed from the outside as the system performs its function

- State Transition Diagrams represent the system states and events that trigger state transitions
  - state— a set of observable circum-stances that characterizes the behavior of a system at a given time
  - state transition— the movement from one state to another
  - event— an occurrence that causes the system to exhibit some predictable form of behavior
  - action— process that occurs as a consequence of making a transition
- STD's indicate actions (e.g. process activation) taken as a consequence of a particular event
- A state is any observable mode of behavior

Example of State Transition Diagram for *Photocopier Software*.



Analysis Principle IV
Partition the Models

- **refine each model to represent lower levels of abstraction**
  - **refine data objects**
  - **create a functional hierarchy**
  - **represent behavior at different levels of detail**

Partitioning

- Horizontally moving – decomposing problem
- Vertically moving – increasing detail

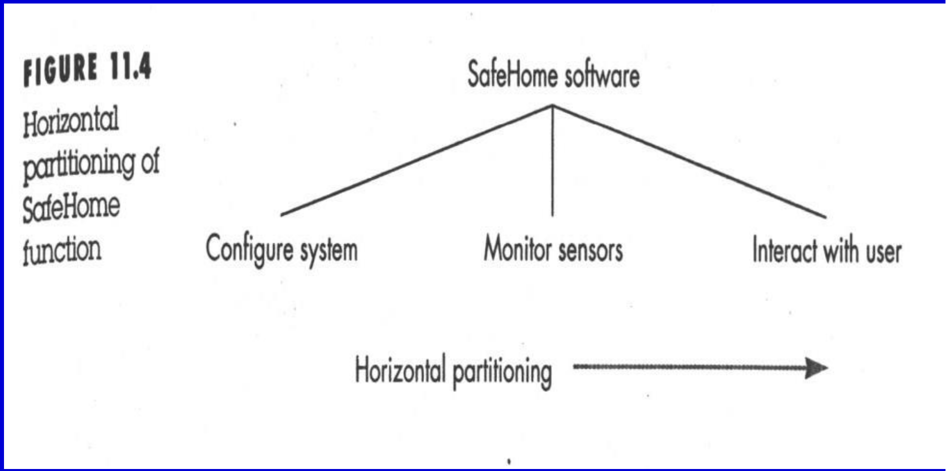


FIGURE 11.4 Horizontal partitioning of SafeHome function

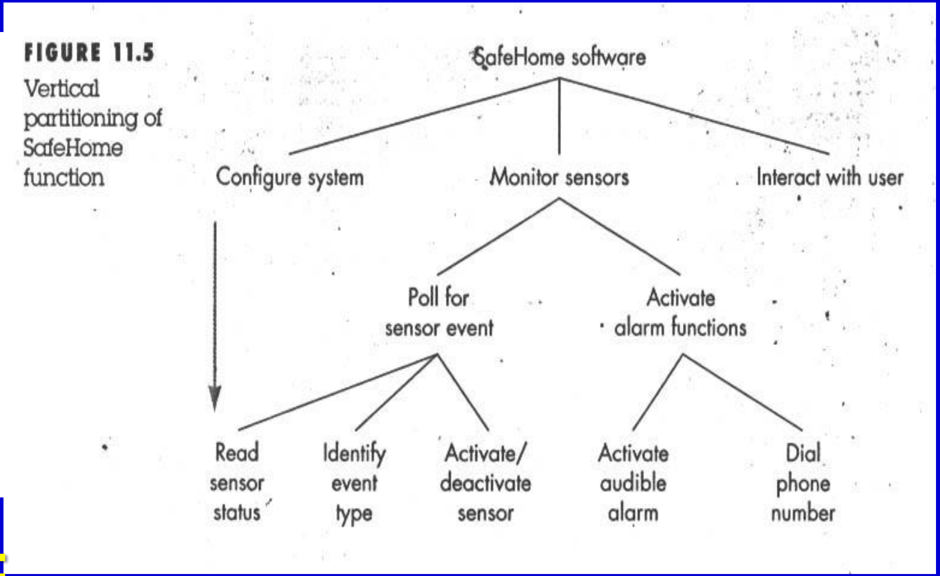


FIGURE 11.5 Vertical partitioning of SafeHome function

- **Prototyping Methods and Tools:**

- Fourth generation techniques - 4GT tools allow software engineer to generate executable code quickly
- Reusable software components - assembling prototype from a set of existing software components
- Formal specification and prototyping environments - can interactively create executable programs from software specification models