Unit 2 Requirement Analysis and Specification

Prepared by :

Dr. Pooja M Bhatt

Computer Engineering Department,

MBIT

Index

- Understanding the Requirement
- Requirement Modeling
- Software Requirement Specification (SRS)
- Requirement Analysis and Requirement Elicitation
- Requirement Engineering

Requirements Analysis is Hard

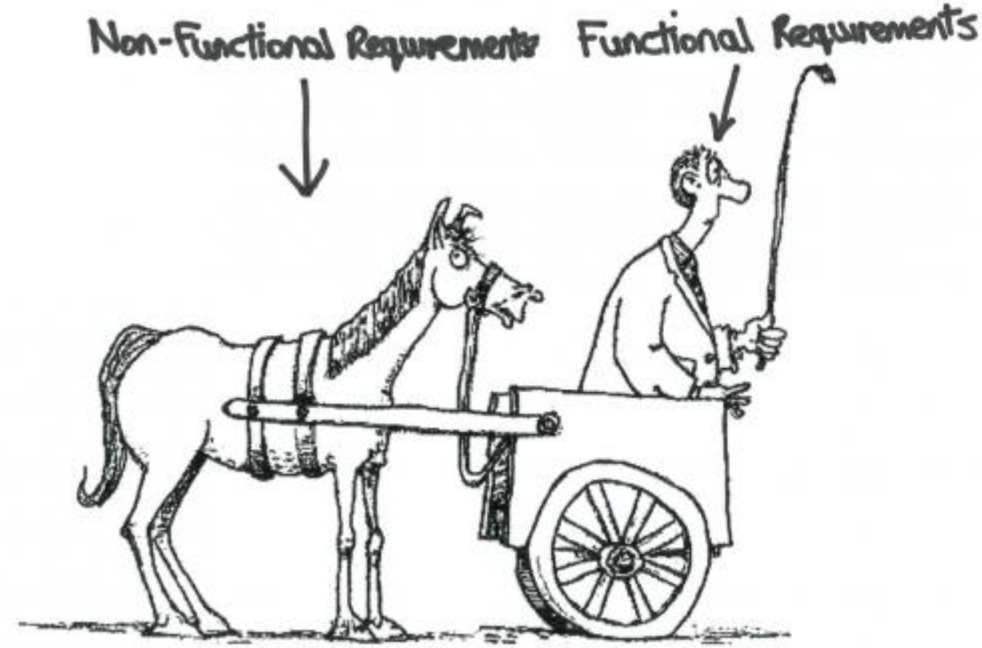
- The **seeds** of major software **disasters** are usually **sown** in the **first three months** of commencing the software project by **Capers Jones.**

Requirement Engineering

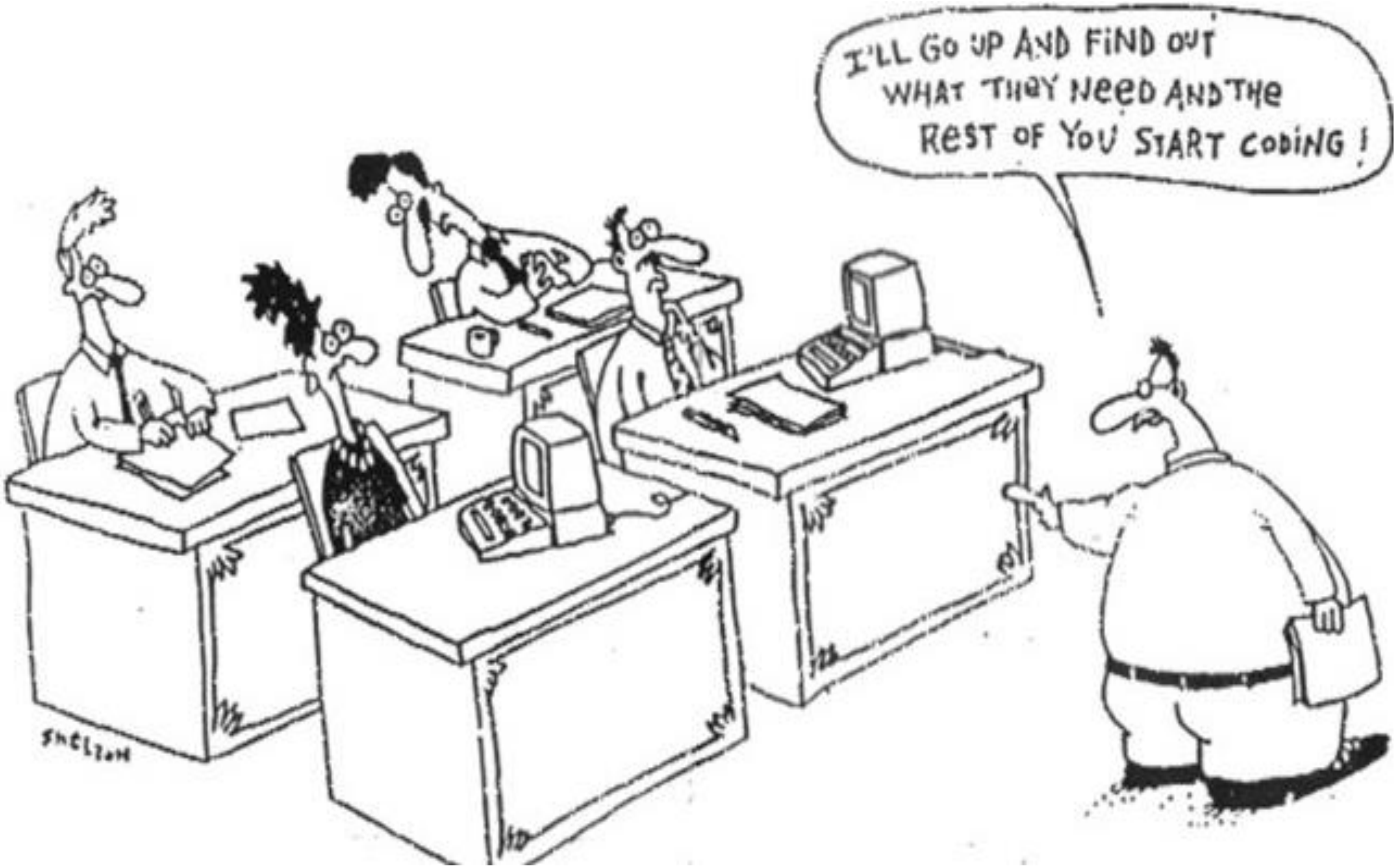
- **Tasks** and **techniques** that lead to an **understanding** of **requirements** is called **requirement engineering**.
- Requirement engineering **provides** the appropriate **mechanism** for **understanding**
- What customer wants
- Analyzing needs
- Assessing feasibility
- Negotiating a reasonable solution
- Specifying solution unambiguously
- Validating the specification
- Managing requirements

Functional & Non-Functional requirements

- Requirements generally fall into two types:
- 1. Functional requirements
- 2. Non-Functional requirements



Don't put what you want to do before how you need to do it


I'LL GO UP AND FIND OUT WHAT THEY NEED AND THE REST OF YOU START CODING!
SHELTON

- **A functional requirement will describe a particular behavior of function of the system when certain conditions are met, for example: “Send email when a new customer signs up” or “Open a new account”.**

functional requirements include:

- Business Rules
- Transaction corrections, adjustments and cancellations
- Administrative functions
- Authentication
- Authorization levels
- Audit Tracking
- External Interfaces
- Reporting Requirements
- Historical Data
- Legal or Regulatory Requirements

A non-functional requirement will describe how a system should behave and what limits there are on its functionality.

Typical Non-Functional requirements include:

- Response time
- Throughput
- Utilization
- Static volumetric
- Scalability
- Capacity
- Availability
- Reliability
- Recoverability
- Maintainability
- Serviceability
- Security
- Regulatory
- Manageability
- Environmental
- Data Integrity
- Usability
- Interoperability

Library Management System

- **Function Requirements**
- **Add Article:** New entries must be entered in database
- **Update Article:** Any changes in articles should be updated in case of update
- **Delete Article:** Wrong entry must be removed from system
- **Inquiry Members:** Inquiry all current enrolled members to view their details
- **Inquiry Issuance:** Inquiry all database articles
- **Check out Article:** To issue any article must be checked out
- **Check In article:** After receiving any article system will reenter article by Checking
- **Inquiry waiting for approvals:** Librarian will generates all newly application which is in waiting list
- **Reserve Article:** This use case is used to reserve any book with the name of librarian, it can be pledged
- **Set user Permission:** From this user case Librarian can give permission categorically, also enabling/disabling of user permission can be set through this use case

Library Management System

- **Non-Function Requirements**
- **Safety Requirements:** The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.
- **Security Requirements:** We are going to develop a secured database for the university. There are different categories of users namely teaching staff, administrator, library staff ,students etc., Depending upon the category of user the access rights are decided. It means if the user is an administrator then he can be able to modify the data, delete, append etc., all other users other than library staff only have the rights to retrieve the information about database.
- **Software Constraints:** The development of the system will be constrained by the availability of required software such as database and development tools.

Requirements Engineering Tasks

- **Inception:**
- Roughly define scope
- A basic understanding of a problem, people who want a solution, the nature of solution desired
- **Elicitation (Requirement Gathering)**
- Define requirements
- The practice of collecting the requirements of a system from users, customers and other stakeholders.
- **Elaboration**
- Further define requirements
- Expand and refine requirements obtained from inception & elicitation
- Creation of User scenarios, extract analysis class and business domain entities

Requirements Engineering Tasks

- **Negotiation**
- Reconcile conflicts
- Agree on a deliverable system that is realistic for developers and customers
- **Specification**
- Create analysis model
- It may be written document, set of graphical models, formal mathematical model, collection of user scenarios, prototype or collection of these.
- **SRS (Software Requirement Specification)** is a document that is created when a detailed description of all aspects of software to build must be specified before starting of project.

Requirements Engineering Tasks

- **Validation**
- Ensure quality of requirements
- Review the requirements specification for errors, ambiguities omissions (absence) and conflicts, missing information
- **Requirements Management**
- It is a set of activities to identify, control & trace requirements & changes to requirements (Umbrella Activities) at any time as the project proceeds.

Elicitation

- **Problems of scope**
- System boundaries are ill-defined.
- Customers will provide irrelevant information.
- **Problems of understanding**
- Customers never know exactly what they want.
- Customers don’t understand capabilities and limitations.
- Customers have trouble fully communicating needs.
- **Problems of volatility**
- Requirements always change.

Project Inception

- During the **initial project meetings**, the following tasks should be accomplished
- **Identify the project stakeholders**
- These are the folks we should be talking to
- **Recognize multiple viewpoints**
- Stakeholders may have different (and conflicting) requirements
- **Work toward collaboration**
- It's all about reconciling conflict and commonality
- Project manager may make final decision about which requirements make the cut.
- **Ask the first questions**
- Who is behind the ? What are the benefits? Another source?
- What is the problem? What defines success? Other constraints?
- Am I doing my job right?

Elicitation of requirement gathering

Collaborative Elicitation: One-on-one Q&A sessions rarely succeed in practice; collaborative strategies are more practical

Elicitation work products

- Collaborative elicitation should result in several **work products**
- A bounded statement of scope
- A list of stakeholders
- A description of the technical environment
- A list of requirements and constraints
- Any prototypes developed
- A set of use cases
- Characterize how users will interact with the system
- Use cases tie functional requirements together

Quality Function Deployment (QFD)

- **This is a technique that translates the needs of the customer into technical requirements for software**.
- It emphasizes an understanding of **what is valuable** to the customer and then deploys these values throughout the engineering process through functions, information, and tasks.
- It identifies **three types of requirements**.
- **Normal requirements:** These requirements are the objectives and goals stated for a product or system during meetings with the customer.
- **Expected requirements:** These requirements are implicit to the product or system and may be so fundamental that the customer does not explicitly state them.
- **Exciting requirements:** These requirements are for features that go beyond the customer's expectations and prove to be very satisfying when present.

The requirement analysis model

1) System Description

2) Analysis Model (bridge between 1 and2)

3) Design Model

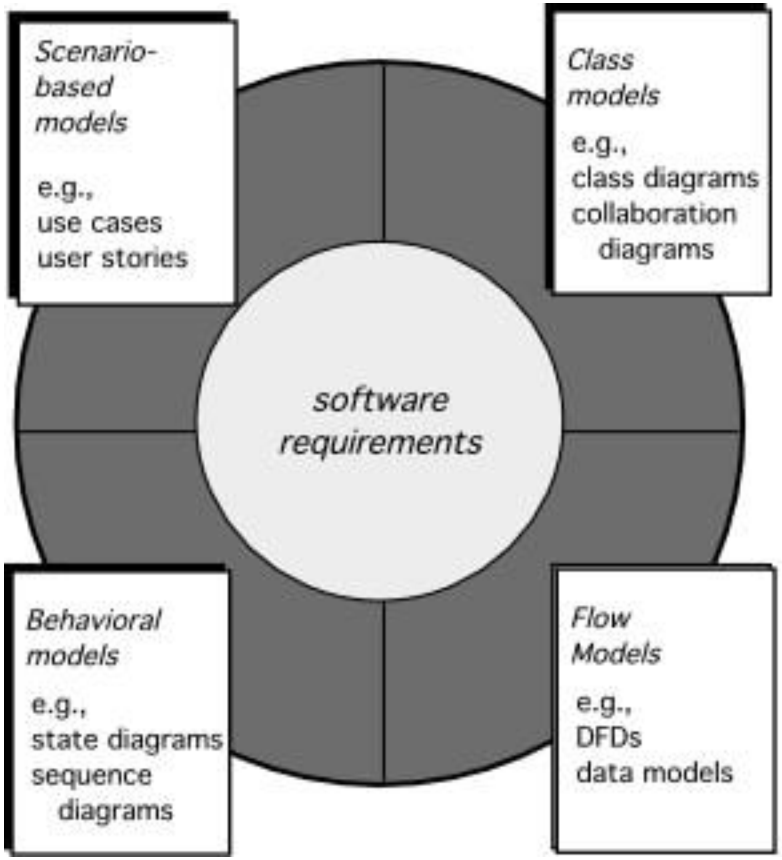
- **Requirement analysis follow analysis rules of thumb.**
- Make sure all points of view are covered.
- Abstraction at high level.
- Coupling between module should be low.
- Simple model should be developed.
- Every element should add value.
- Model should provide value to all stakeholders.

Analysis modeling approach

- **Structured**
- Models data elements: attributes, relationships
- Model processes that transform data
- **Object oriented**
- Model analysis classes: data, process
- Models class collaborations

- Both approaches are used in practice.

Elements of the requirement Model



Scenario-based models

e.g.,
use cases
user stories

Class models

e.g.,
class diagrams
collaboration diagrams

software requirements

Behavioral models

e.g.,
state diagrams
sequence diagrams

Flow Models

e.g.,
DFDs
data models

Elements of the requirement Model

- **Scenario-based:** user interaction with system
  - Use-case: descriptions of the interaction
- **Data-based:** data objects in system
  - ER (entity-relation) Diagrams
  - Class-based: data + methods
    - OO, Class diagrams, implied by scenarios
- **Behavioral-based:** how external events change system
  - State, sequence diagram
- **Flow-based:** information transforms
  - Data flow, control flow diagrams

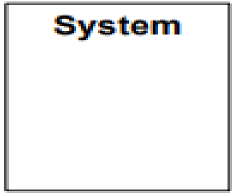
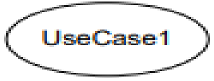
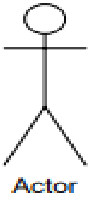
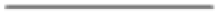
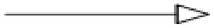
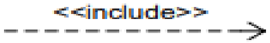
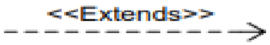
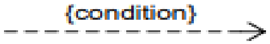
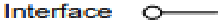
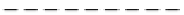
USE CASE

- The purpose of use case diagram is to capture the dynamic aspect of a system.
- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an "actor"
- Actor: a person or device that interacts with the software
- Each scenario answers the following questions:
- Who is the primary actor, the secondary actor (s)?
- What are the actor's goals?
- What preconditions should exist before the story begins?
- What main tasks or functions are performed by the actor?
- What extensions might be considered as the story is described?

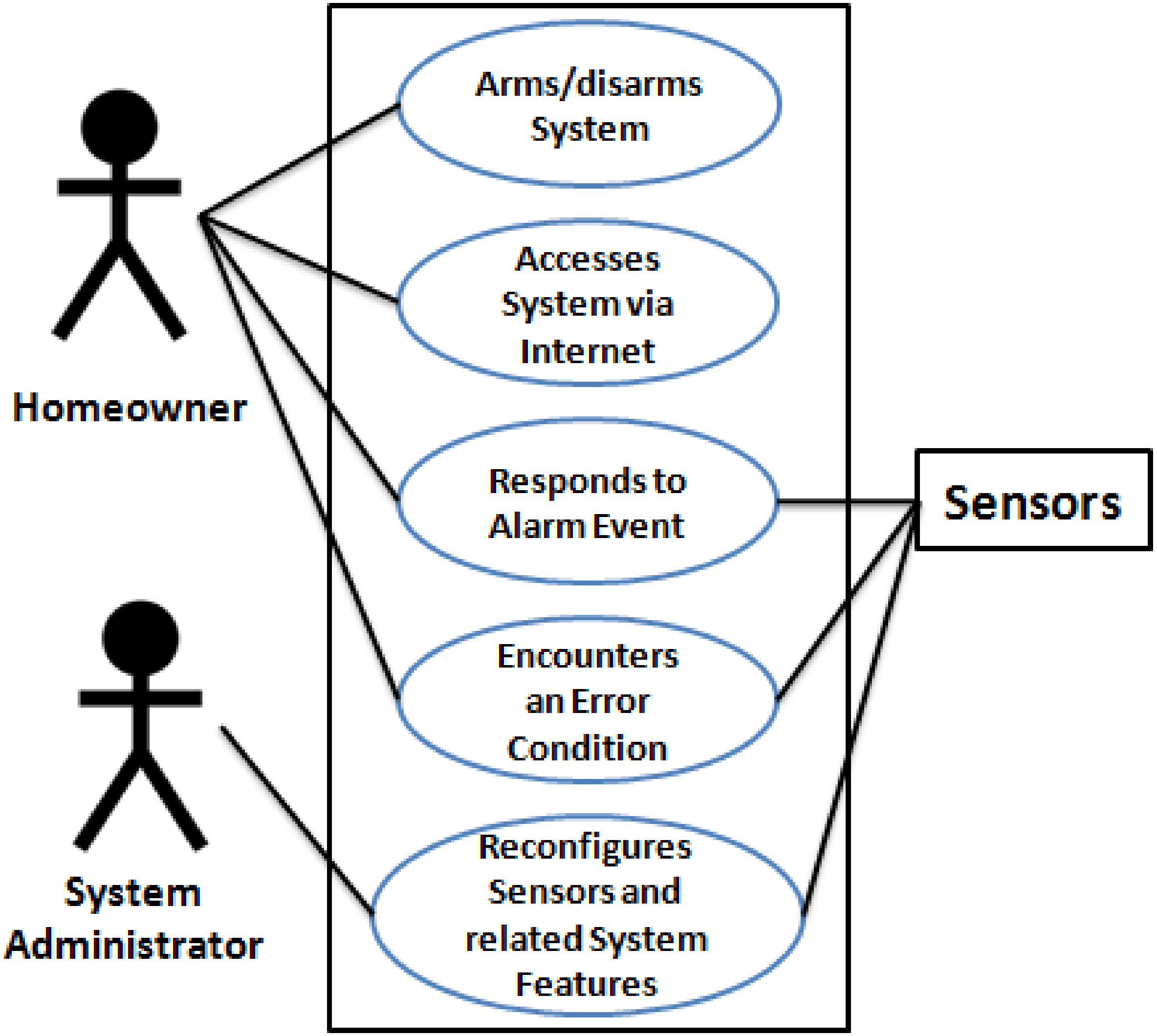
USE CASE

- What variations in the actor's interaction are possible?
- What system information will the actor acquire, produce, or change?
- Will the actor have to inform the system about changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?

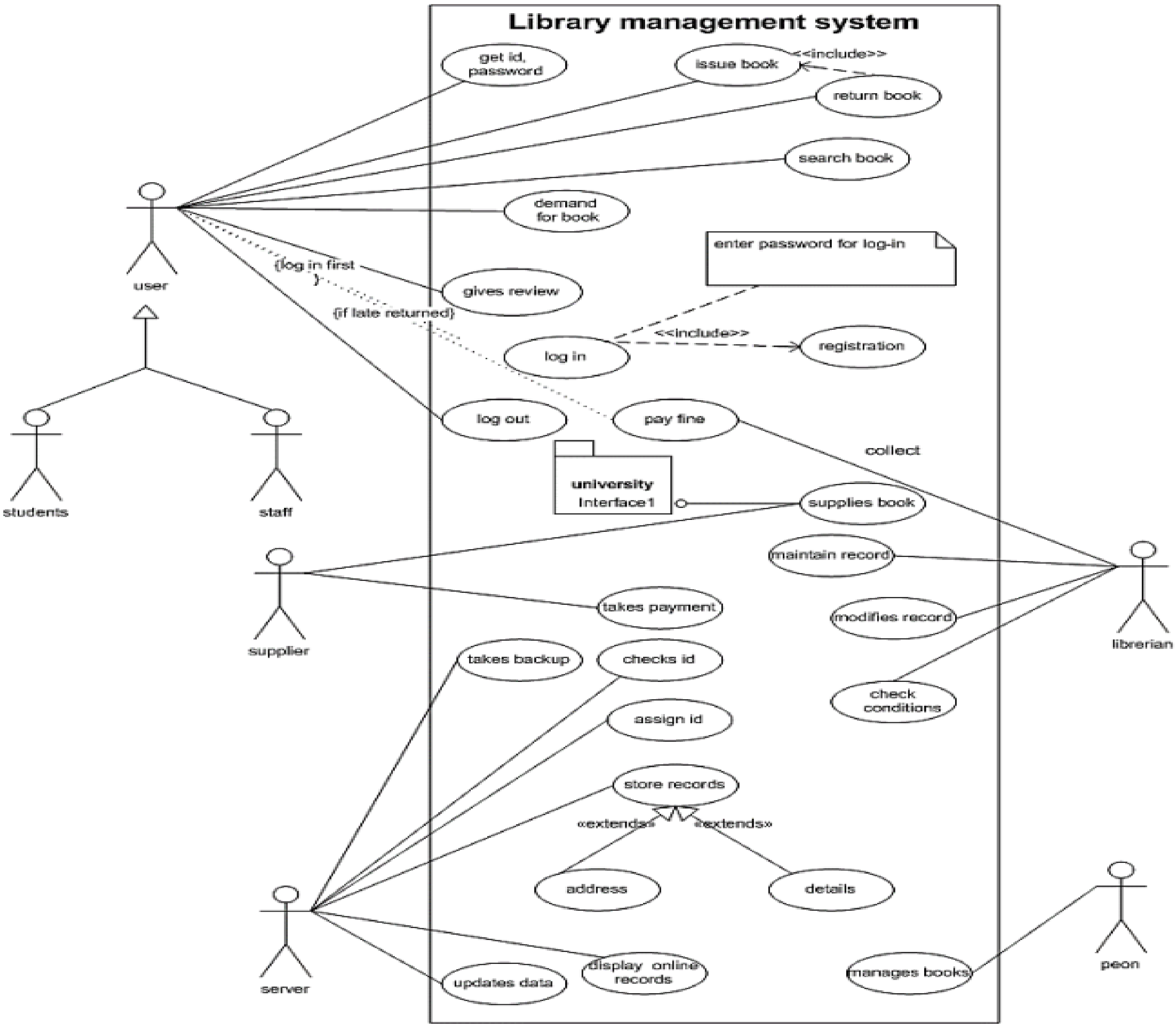
USE CASE DIAGRAM SYMBOL

Name	Notation	Description
System boundary		The scope of a system can be represented by a system boundary. The use cases of the system are placed inside the system boundary, while the actors who interact with the system are put outside the system. The use cases in the system make up the total requirements of the system.
Use case		A use case represents a user goal that can be achieved by accessing the system or software application.
Actor		Actors are the entities that interact with a system. Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system. So an actor may be people, computer hardware, other systems, etc. Note that actor represent a role that a user can play, but not a specific user.
Association		Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association corresponds to a sequence of actions between the actor and use case in achieving the use case.
Generalizatio n		A generalization relationship is used to represent inheritance relationship between model elements of same type.
Include		An include relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.
Extends		An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.
Constraint		Show condition exists between actors an activity.
Interface		Interface is used to connect package and use-case. Head is linked with package and tail linked with use-case.
Anchor		Anchor is used to connect a note the use case in use case diagram

Use case for home automation system


Arms/disarms System
Accesses System via Internet
Responds to Alarm Event
Encounters an Error Condition
Reconfigures Sensors and related System Features
Homeowner
System Administrator
Sensors

Use case for library management system

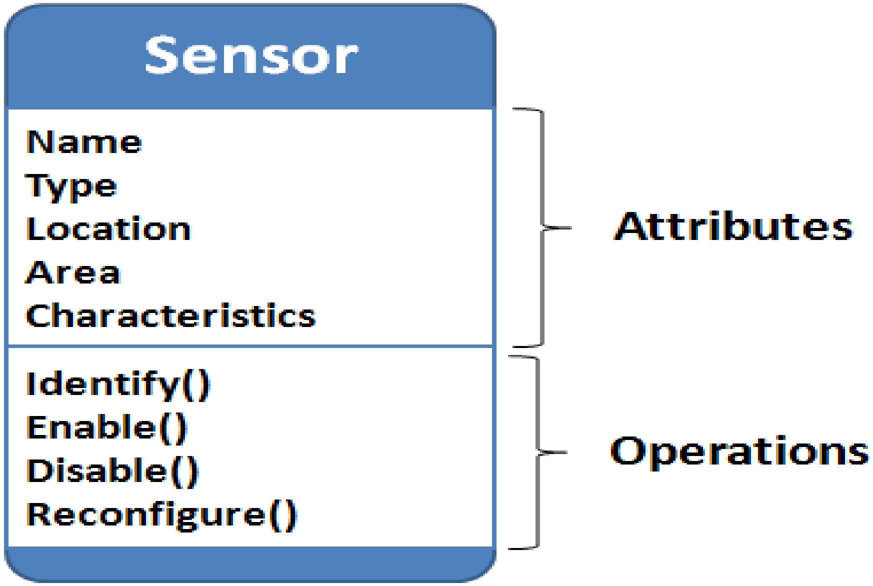

Library management system
get id, password
issue book
<<include>>
return book
search book
demand for book
enter password for log-in
gives review
{log in first }
{if late returned}
log in
<<include>>
registration
log out
pay fine
collect
university
Interface1
supplies book
maintain record
modifies record
check conditions
takes payment
takes backup
checks id
assign id
store records
«extends»
«extends»
address
details
display online records
updates data
manages books
user
students
staff
supplier
server
librerian
peon

- **1. Use Case Title :Login**
- **2 Abbreviated Title** :Login
- **3. Use Case Id:1**
- **4. Actors:**Librarian , USERS,suppliers, server,peon
- **5. Description:** To interact with the system, LMS will validate its registration with this system. It also defines the actions a user can perform in LMS.

- **5.1 Pre Conditions:** User must have proper client installed on user terminal
- **5.2 Task Sequence**
- 1. System show Login Screen
- 2. User Fill in required information. Enter user name and password 3. System acknowledge entry
- **5.3 Post Conditions:** System transfer control to user main screen to proceed further actions
- **5.4 Exception:** If no user found then system display Invalid user name password error message and transfer control to Task Sequence no.1
- **6 Modification history:** Date 31-01-2020
- **7 Author:** Jone James Project ID LMS

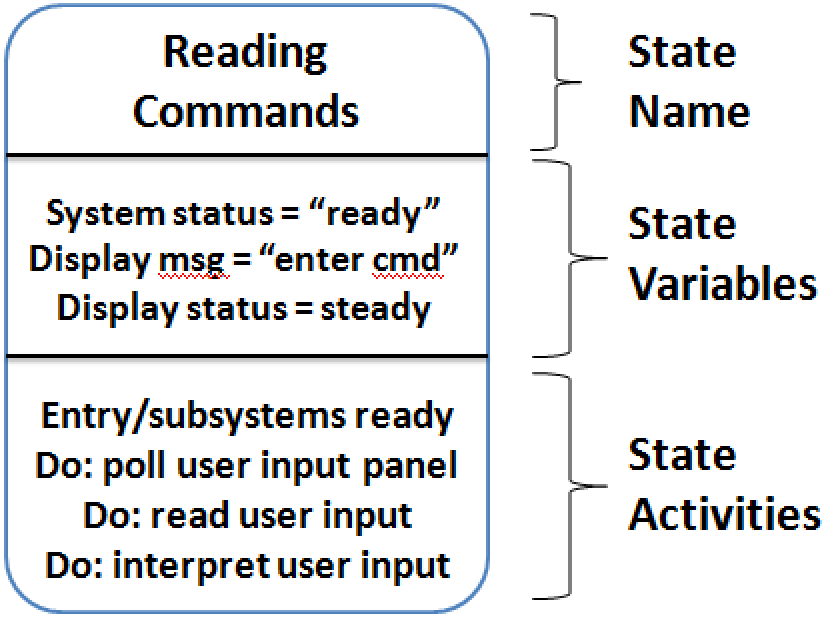
Class diagram

- It **describes** the structure of a system by showing the **system's classes**, their **attributes**, and the **operations relationships** among **objects**.
- Example of class of sensor



State Diagram

- It is used to **describe** the **behavior** of **systems**.
- It requires that the system described is composed of a finite number of states.



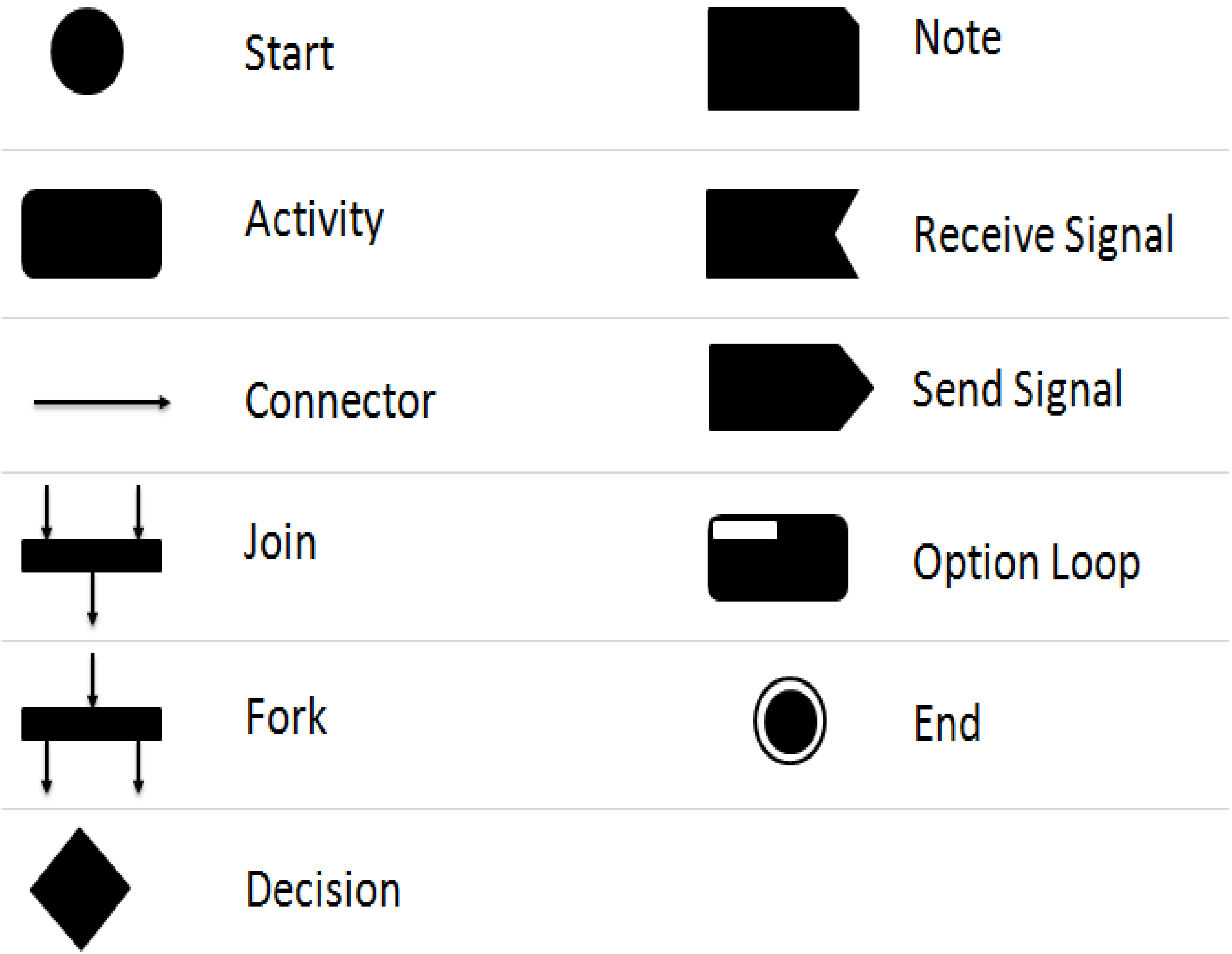
Reading Commands

System status = "ready"
Display msg = "enter cmd"
Display status = steady

Entry/subsystems ready
Do: poll user input panel
Do: read user input
Do: interpret user input

Activity & Swim lane Diagram

- **Activity diagram** is basically a **flowchart** to **represent** the **flow** from one **activity** to another **activity**
- The activity can be described as an operation of the system.
- A **swim lane diagram** is a type of activity diagram. Like activity diagram, it diagrams a process from start to finish, but it also **divides** these **steps** into **categories** to help **distinguish** which departments or employees are **responsible** for each set **of actions**
- A swim lane diagram is also useful in helping **clarify responsibilities** and help departments work together in a world where departments often don't understand what the other departments do.



Start

Activity

Connector

Join

Fork

Decision

Note

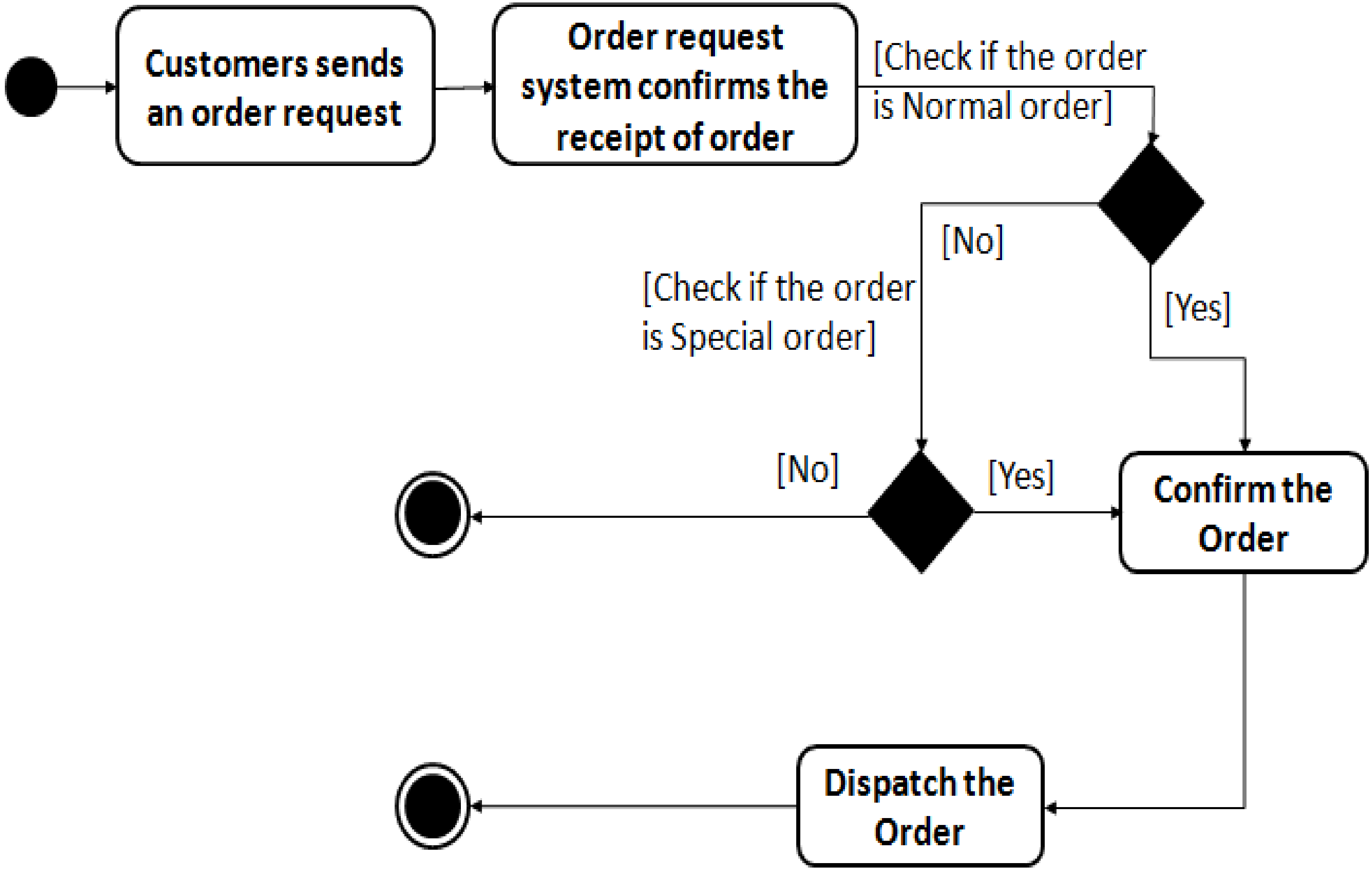
Receive Signal

Send Signal

Option Loop

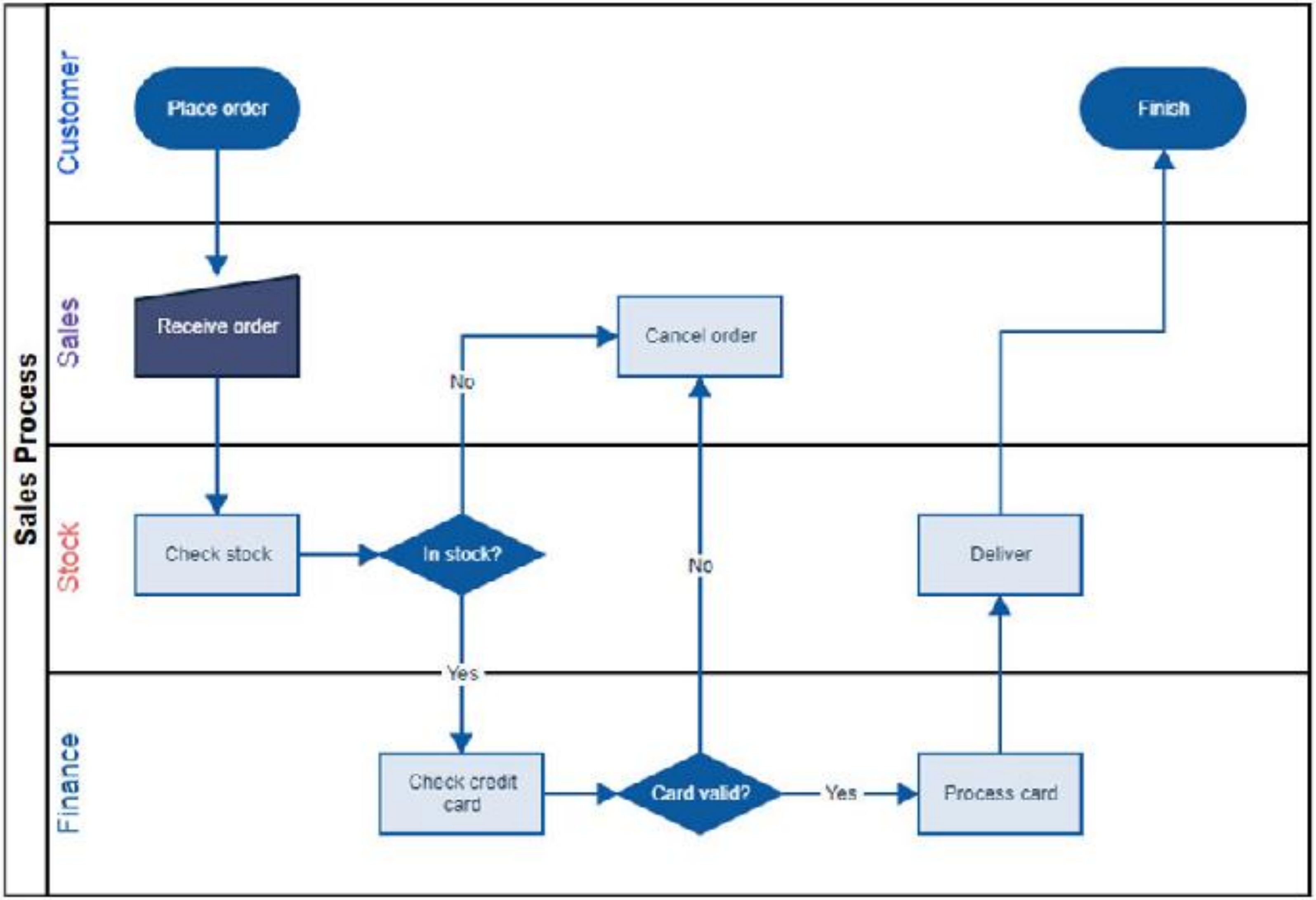
End

Example 1



Customers sends an order request
Order request system confirms the receipt of order
[Check if the order is Normal order]
[No]
[Yes]
[Check if the order is Special order]
[No]
[Yes]
Confirm the Order
Dispatch the Order

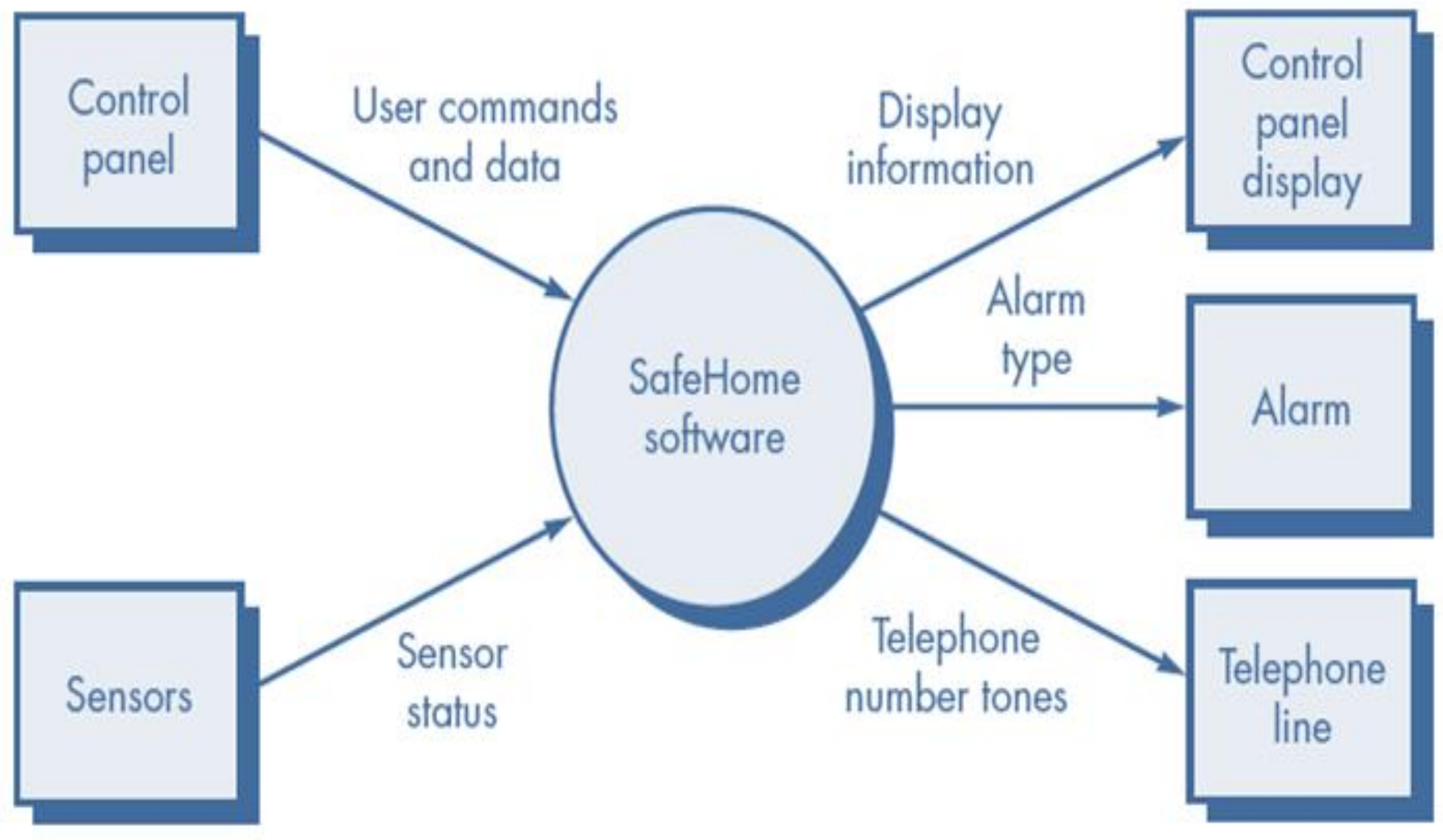
Swim lane diagram



DFD

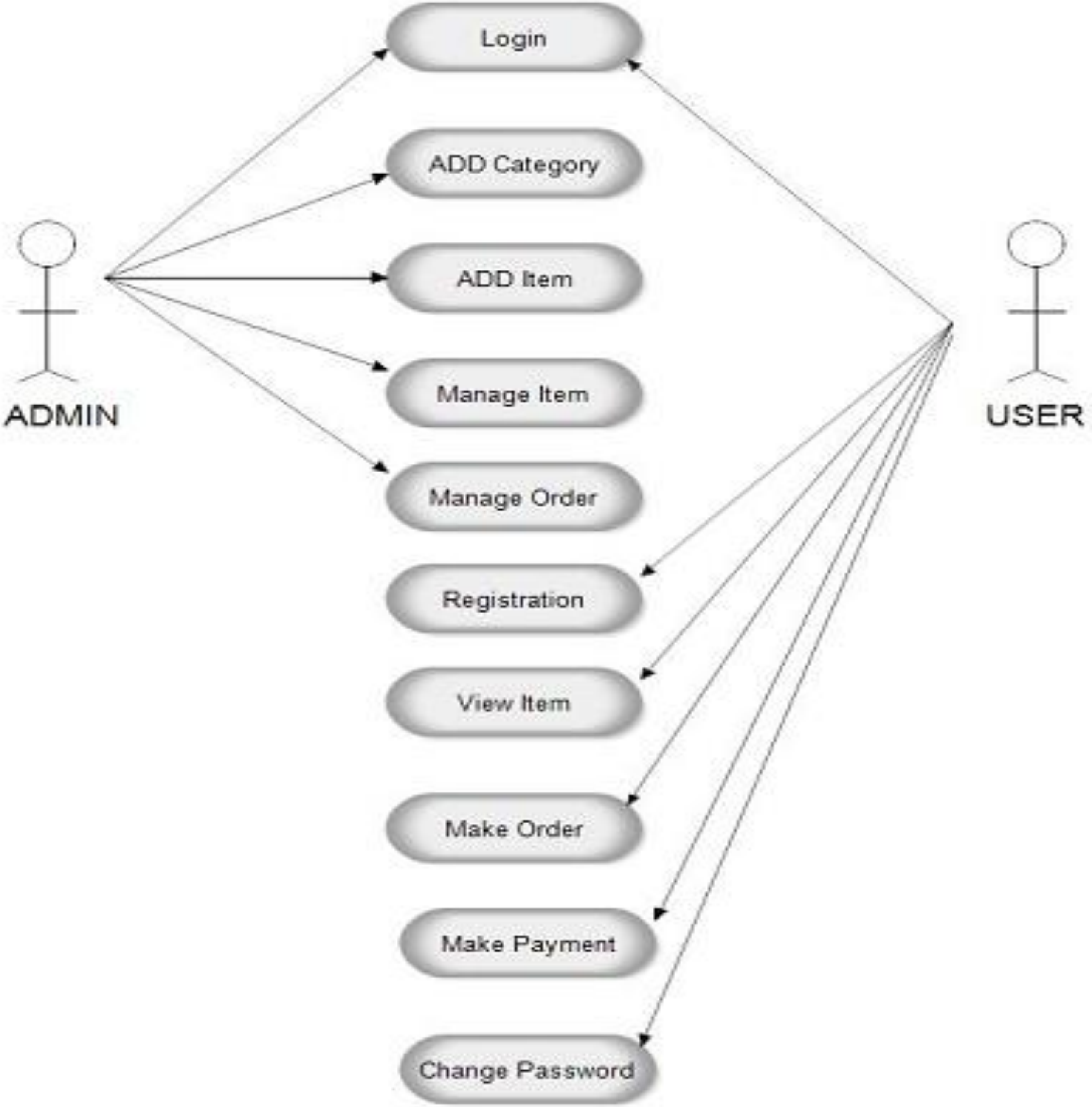
- It is a graphical representation of the "flow" of data through an information system, modeling its process aspects
- It is often **used** as a preliminary step to create an overview of the system, which can later be elaborated.

Context-level DFD for safe home security function



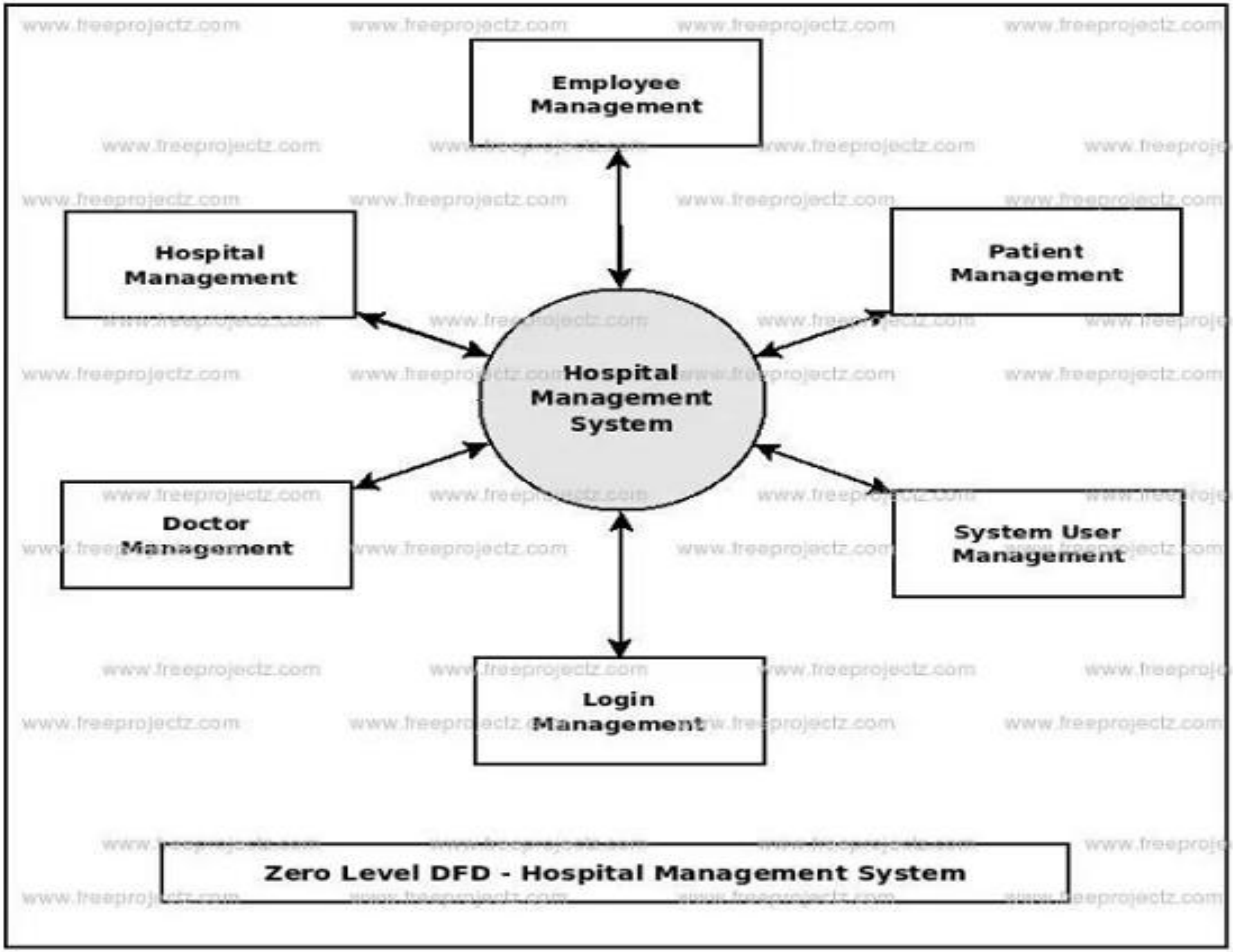
Example 2

Use-case for online shopping

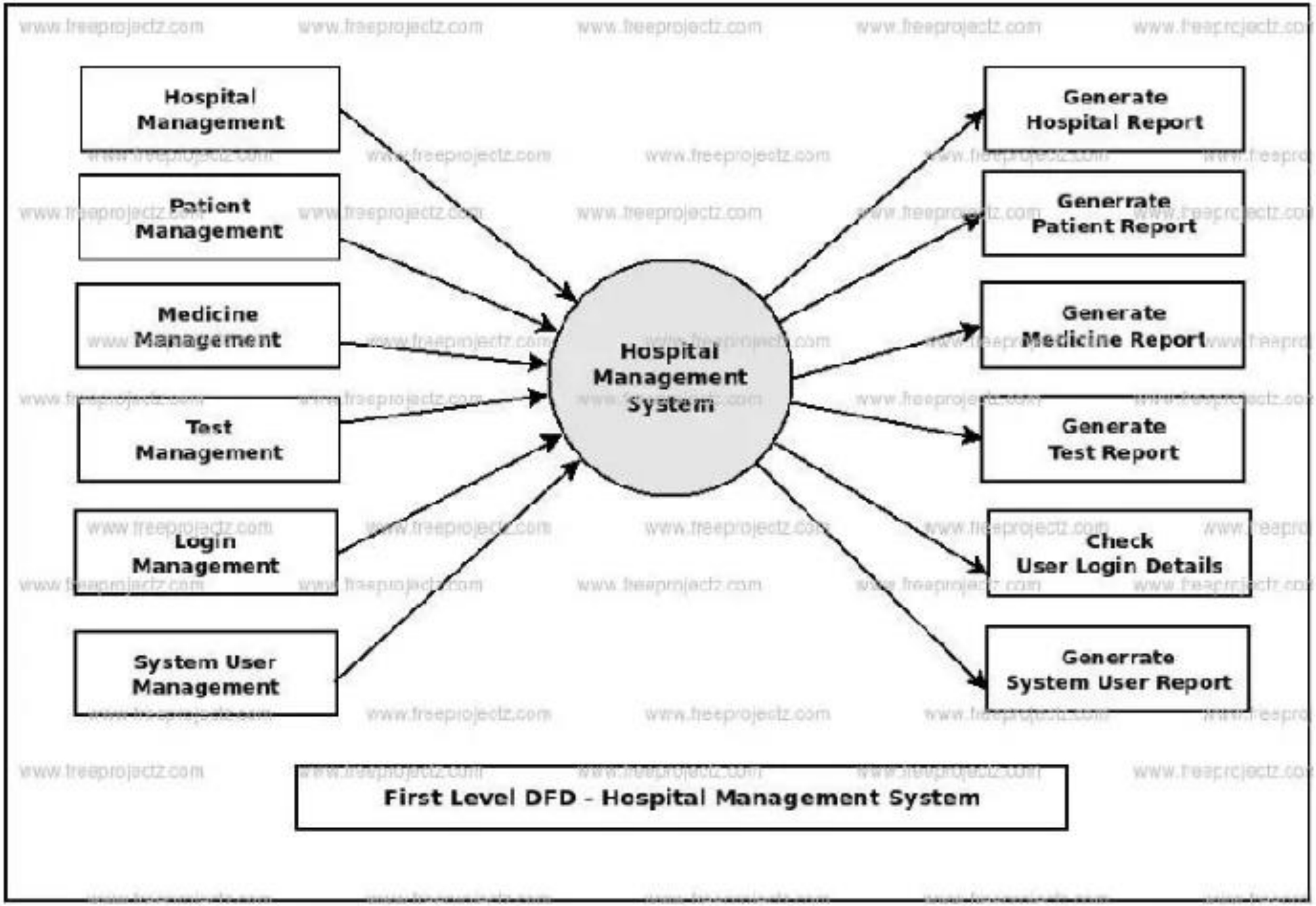

Login
ADD Category
ADD Item
Manage Item
Manage Order
Registration
View Item
Make Order
Make Payment
Change Password
ADMIN
USER

Example 3

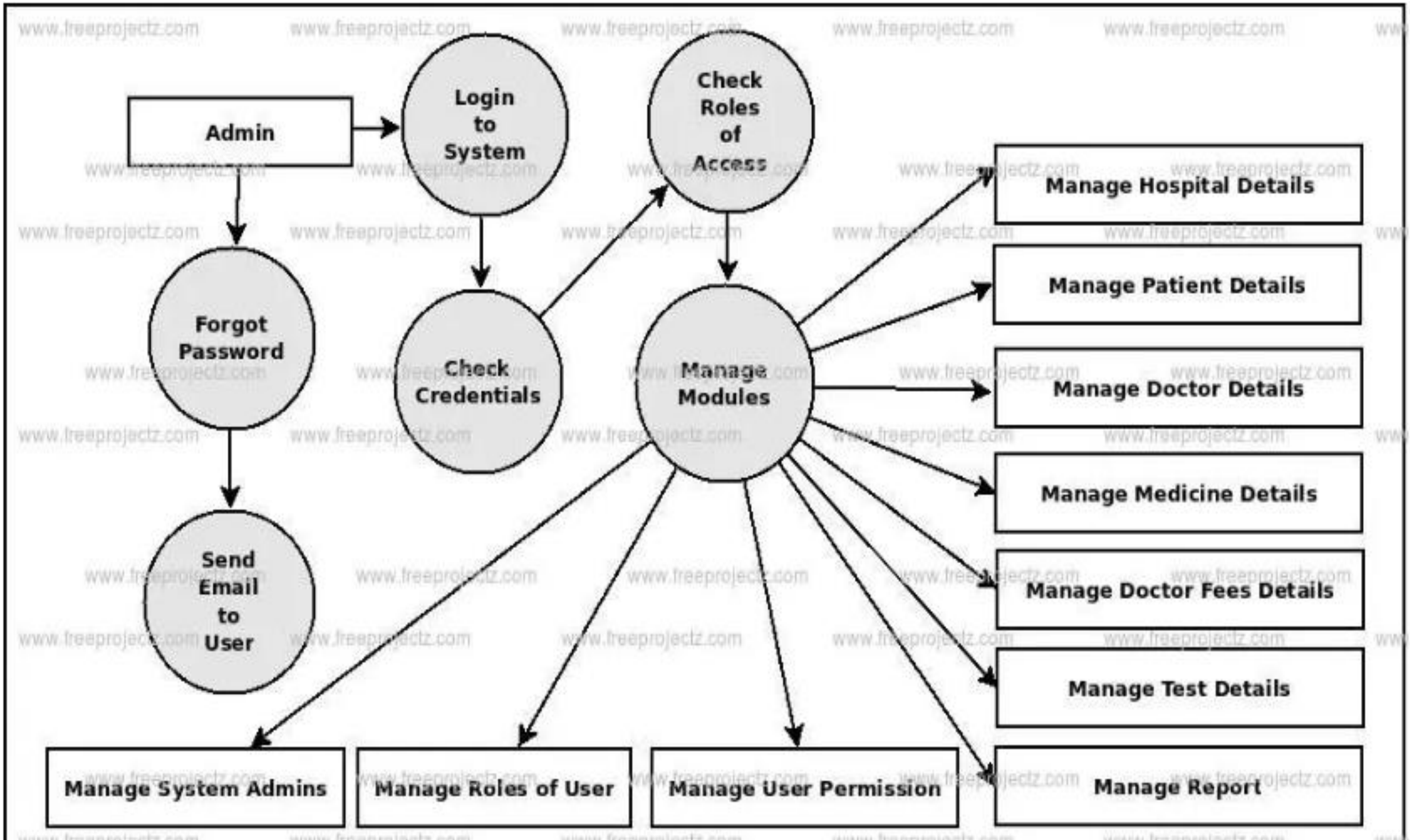
Hospital management DFD



Zero Level DFD - Hospital Management System



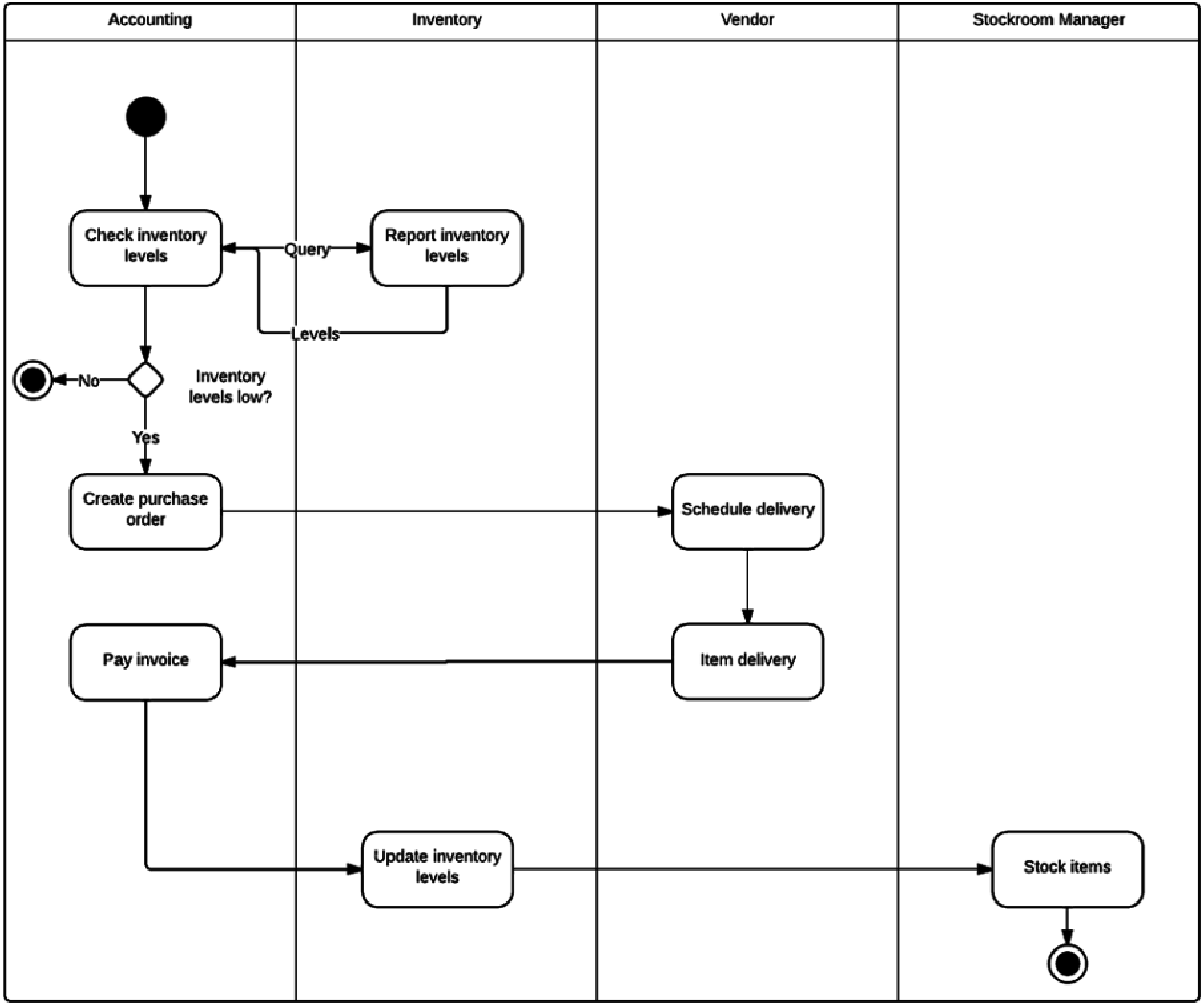
First Level DFD - Hospital Management System



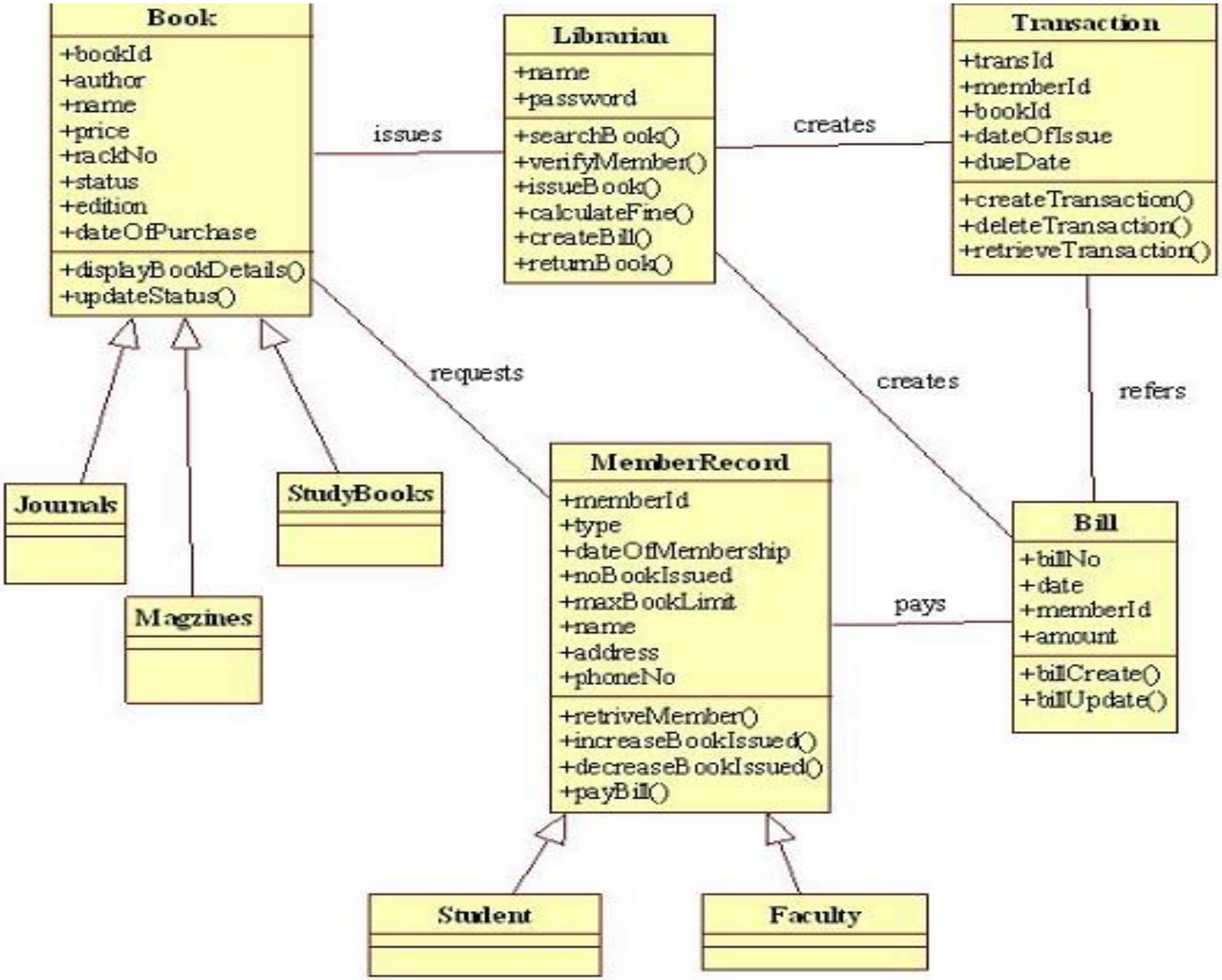
Second Level DFD - Hospital Management System

Example 4

Swim lane diagram


Accounting
Inventory
Vendor
Stockroom Manager
Check inventory levels
Query
Report inventory levels
Levels
No
Inventory levels low?
Yes
Create purchase order
Schedule delivery
Pay invoice
Item delivery
Update inventory levels
Stock items

Example 5 (class diagram of Library system)



Book
+bookId
+author
+name
+price
+rackNo
+status
+edition
+dateOfPurchase
+displayBookDetails()
+updateStatus()

Librarian
+name
+password
+searchBook()
+verifyMember()
+issueBook()
+calculateFine()
+createBill()
+returnBook()

Transaction
+transId
+memberId
+bookId
+dateOfIssue
+dueDate
+createTransaction()
+deleteTransaction()
+retrieveTransaction()

issues

creates

requests

creates

refers

Journals

Magzines

StudyBooks

MemberRecord
+memberId
+type
+dateOfMembership
+noBookIssued
+maxBookLimit
+name
+address
+phoneNo
+retriveMember()
+increaseBookIssued()
+decreaseBookIssued()
+payBill()

pays

Bill
+billNo
+date
+memberId
+amount
+billCreate()
+billUpdate()

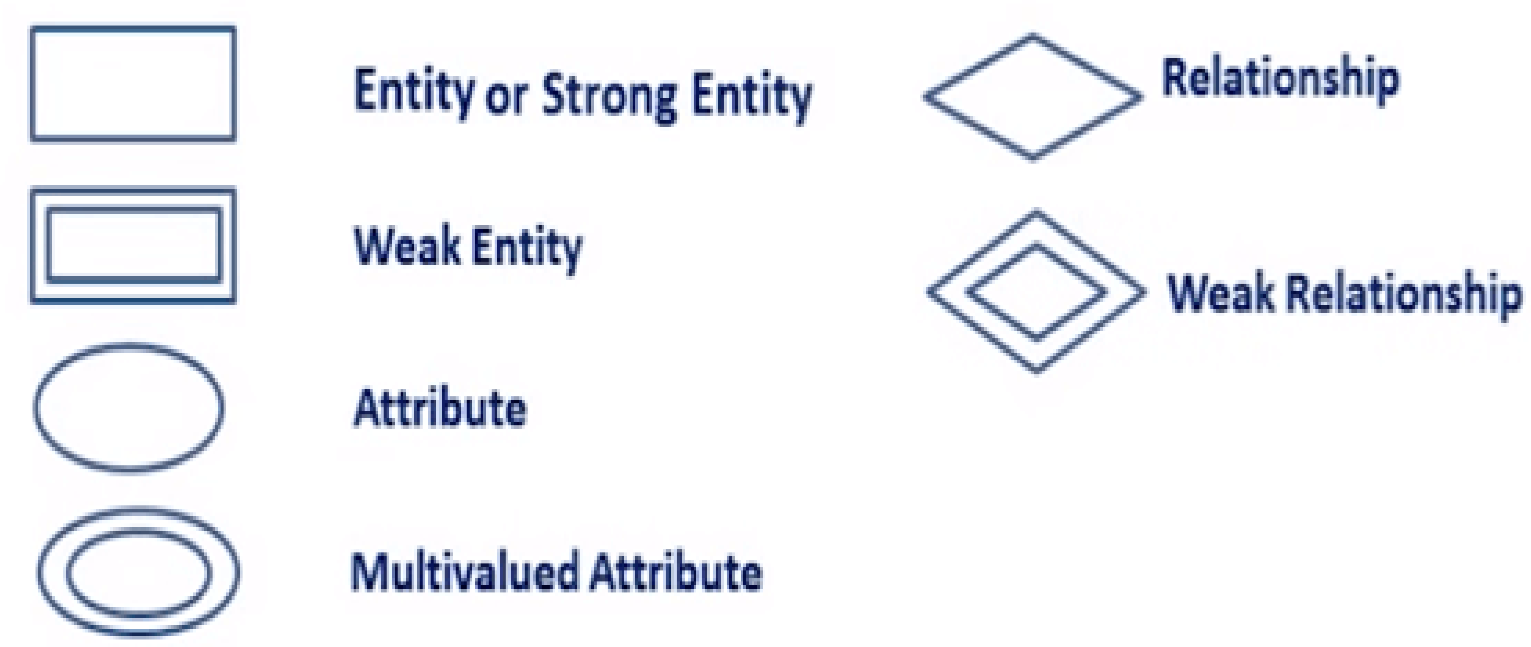
Student

Faculty

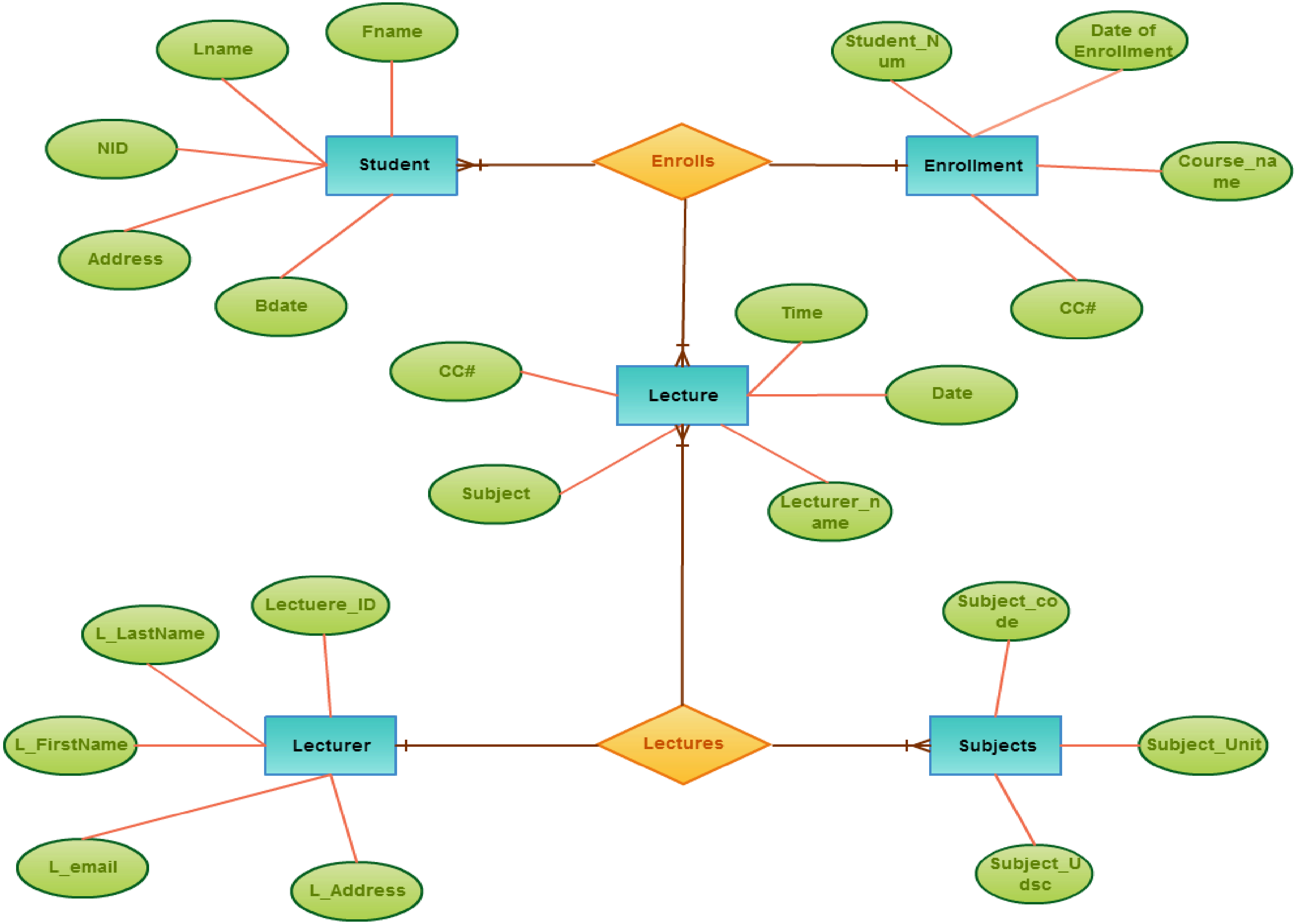
ER diagram

- Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.
- Following are the main components and its symbols in ER Diagrams:
- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

ER Diagram Symbols



ER DIAGRAM FOR STUDENT ENROLLMENT SYSTEM



Modeling for WebApps

- **Content Analysis:** the content is identified, including text, graphics and images, video, and audio data.
- **Interaction Analysis:** The manner in which the user interacts with the WebApp is described in detail.
- **Functional Analysis:** The usage scenarios (use-cases) created as part of interaction analysis define the operations that will be applied to WebApp content and imply other processing functions.
- **Configuration Analysis:** The environment and infrastructure in which the WebApp resides are described in detail. Server-client side.

Software Requirements Specification

- Software Requirement Specification (SRS) completely describes what the proposed without describing how software will do it.
- It contains:
- a document that software should do
- a complete information description
- a detailed functional description
- a representation of system behavior
- an indication of performance requirements and design constraints
- appropriate validation criteria
- other information suitable to requirements
- SRS is also helping the clients to understand their own needs.

Characteristics of a Good SRS

- SRS should be accurate, complete, efficient, and of high quality
- so that it does not affect the entire project plan.
- An SRS is said to be of high quality when the developer and user easily understand the prepared document.
- Characteristics of a Good SRS:
- **Correct**
- SRS is correct when all user requirements are stated in the requirements document.
- Note that there is no specified tool or procedure to assure the correctness of SRS.
- **Unambiguous**
- SRS is unambiguous when every stated requirement has only one interpretation.

Characteristics of a Good SRS

- **Complete**
- SRS is complete when the requirements clearly define what the software is required to do.
- **Ranked for Importance/Stability**
- All requirements are not equally important, hence each requirement is identified to make differences among other requirements.
- Stability implies the probability of changes in the requirement in future.
- **Modifiable**
- The requirements of the user can change, hence requirements document should be created in such a manner that those changes can be modified easily.
- **Traceable**
- SRS is traceable when the source of each requirement is clear and facilitates the reference of each requirement in future.

Characteristics of a Good SRS

- **Verifiable**
- SRS is verifiable when the specified requirements can be verified with a cost-effective process to check whether the final software meets those requirements.
- **Consistent**
- SRS is consistent when the subsets of individual requirements defined do not conflict with each other.

Standard Template

- **Front Page**
- **Software Requirements Specification for**
- **<Project> Version <no.>**
- **Prepared by <author> <organization>**
- **<date created>**
- **Table of Contents**
- **Revision History**

Standard Template

- **1. Introduction**
- 1.1 Purpose
- 1.2 Document Conventions
- 1.3 Intended Audience and Reading Suggestions
- 1.4 Project Scope
- 1.5 References
- **2. Overall Description**
- 2.1 Product Perspective
- 2.2 Product Features
- 2.3 User Classes and Characteristics
- 2.4 Operating Environment
- 2.5 Design and Implementation Constraints

Standard Template

- 2.6 User Documentation
- 2.7 Assumptions and Dependencies
- **3. System Features**
- 3.1 System Feature 1
- 3.2 System Feature 2 (and so on)
- **4. External Interface Requirements**
- 4.1 User Interfaces
- 4.2 Hardware Interfaces 4.3 Software Interfaces
- 4.4 Communications Interfaces

Standard Template

- **5. Other Nonfunctional Requirements**
- 5.1 Performance Requirements
- 5.2 Safety Requirements
- 5.3 Security Requirements
- 5.4 Software Quality Attributes
- **6. Other Requirements Appendix A: Glossary**
- **Appendix B: Analysis Models Appendix C: Issues List**

Problems Without SRS

- Without developing the SRS document, the system would not be properly implemented according to customer needs.
- Software developers would not know whether what they are developing is what exactly is required by the customer.
- Without SRS, it will be very difficult for the maintenance engineers to understand the functionality of the system.
- It will be very difficult for user document writers to write the users' manuals properly without understanding the SRS.

References

- https://www.startertutorials.com/uml/uml-diagrams-library-management-system.html
- https://www.freeprojectz.com/dfd/hospital-management-system-dataflow-diagram
- https://www.uml-diagrams.org/examples/online-shopping-use-case-diagram-example.html
- https://www.studentprojects.live/studentprojectreport/project-srs/library-management-system-project-srs-document/

Assignment

- What is SRS? What are the characteristics of good SRS? Explain with example(hospital management system, college management).(GTU 2013,2017).
- List and explain requirement engineering task. Or requirement engineering process. (GTU212,2014,2016)
- What is relationship? Explain Cardinality and modality with examples.(GTU2015)
- Explain control flow model with example.(GTU 2013) (state chart)
- Explain use case with example(library management system).
- Explain activity and swim lane with example(billing counter in shopping mall).
- Explain DFD with example.
- Explain class diagram of library management system.
- Explain functional and non functional requirement of Library management system.