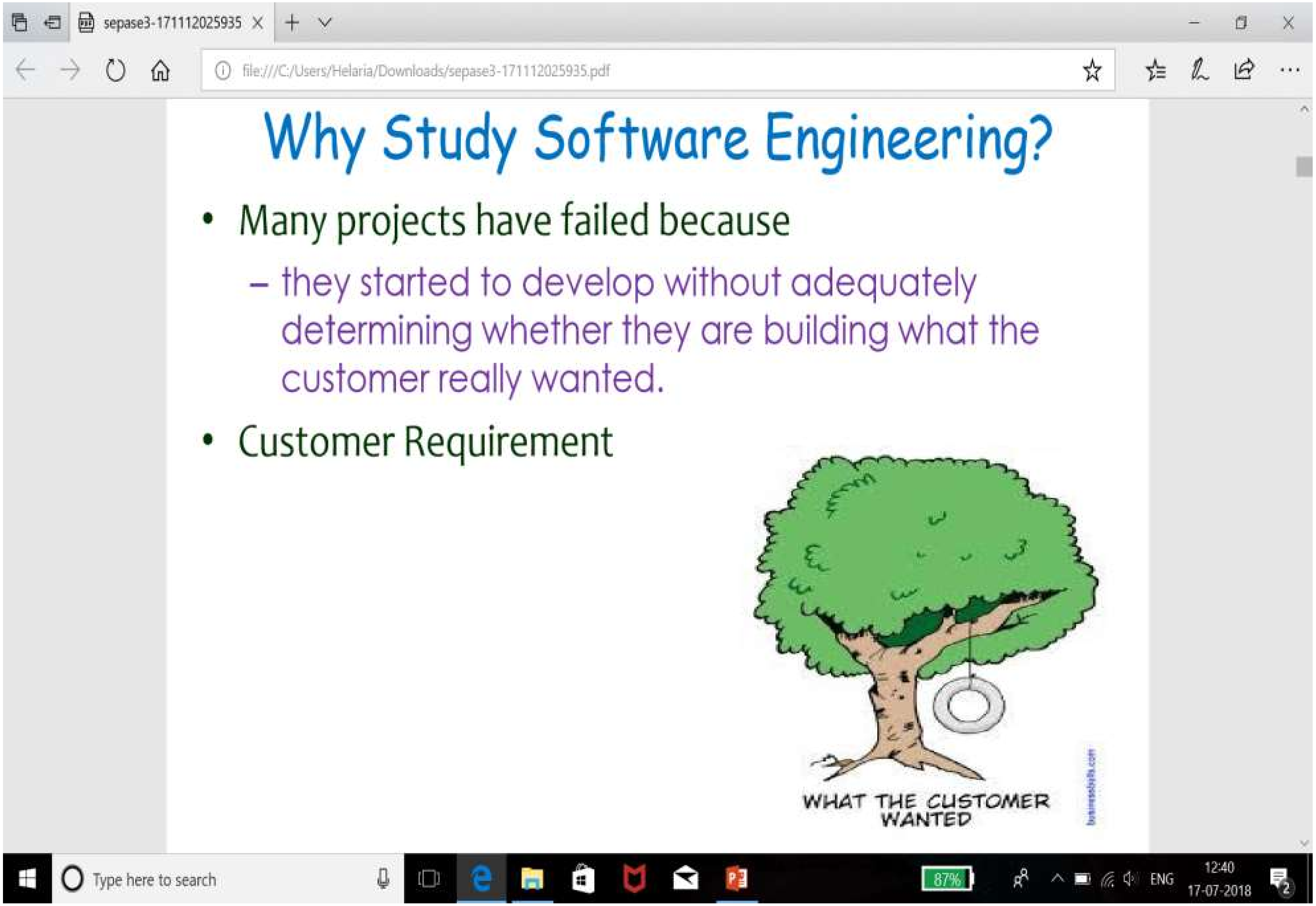
Software Requirements

- Descriptions and specifications of a system

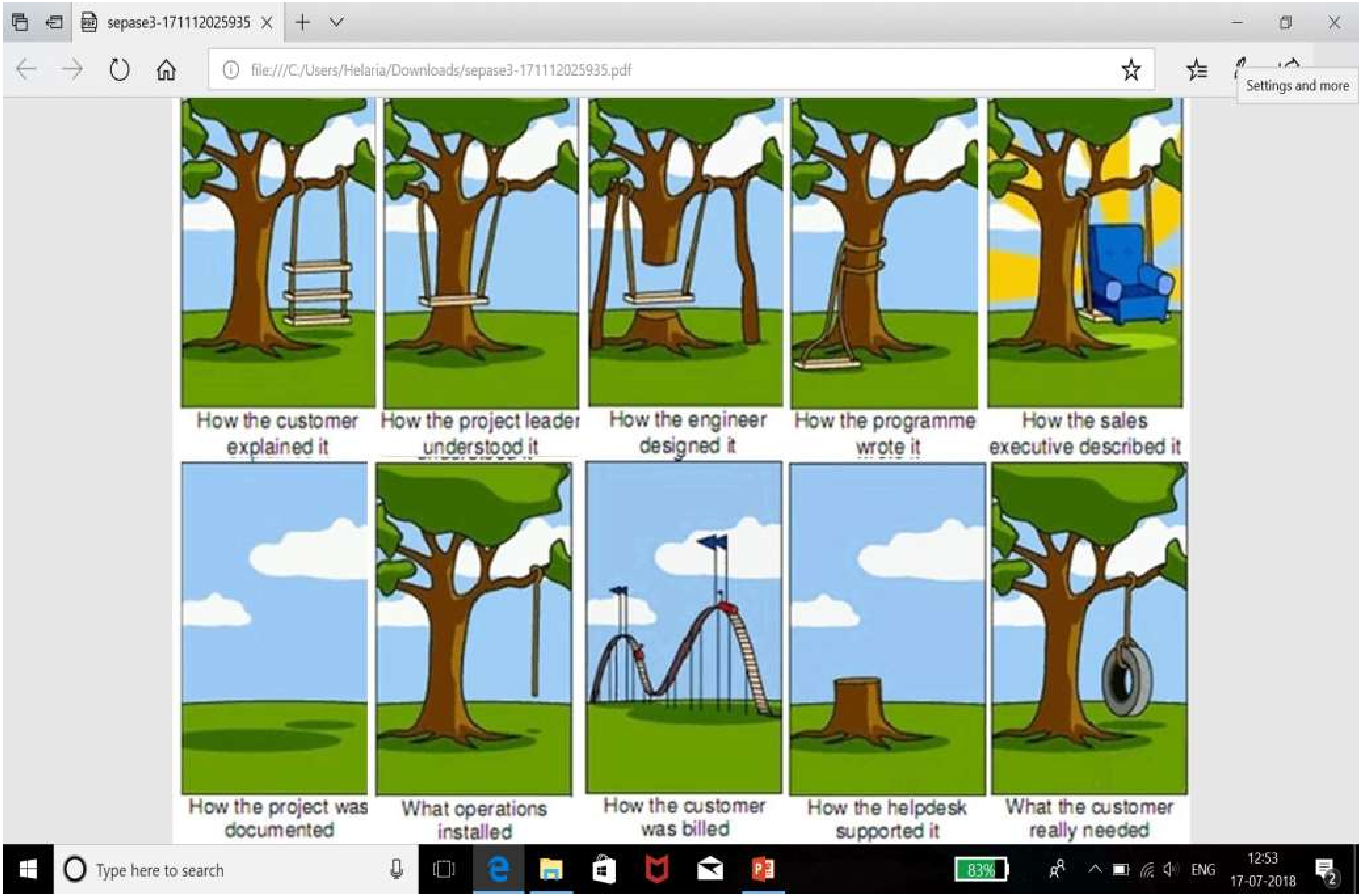
Requirements engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process



Why Study Software Engineering?

- Many projects have failed because
  - they started to develop without adequately determining whether they are building what the customer really wanted.
- Customer Requirement


sepase3-171112025935
file:///C:/Users/Helaria/Downloads/sepase3-171112025935.pdf
Settings and more
How the customer explained it
How the project leader understood it
How the engineer designed it
How the programme wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed
Type here to search
83%
ENG
12:53
17-07-2018

What is a requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification
- Requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation
  - May be the basis for the contract itself - therefore must be defined in detail

  - Both these statements may be called requirements

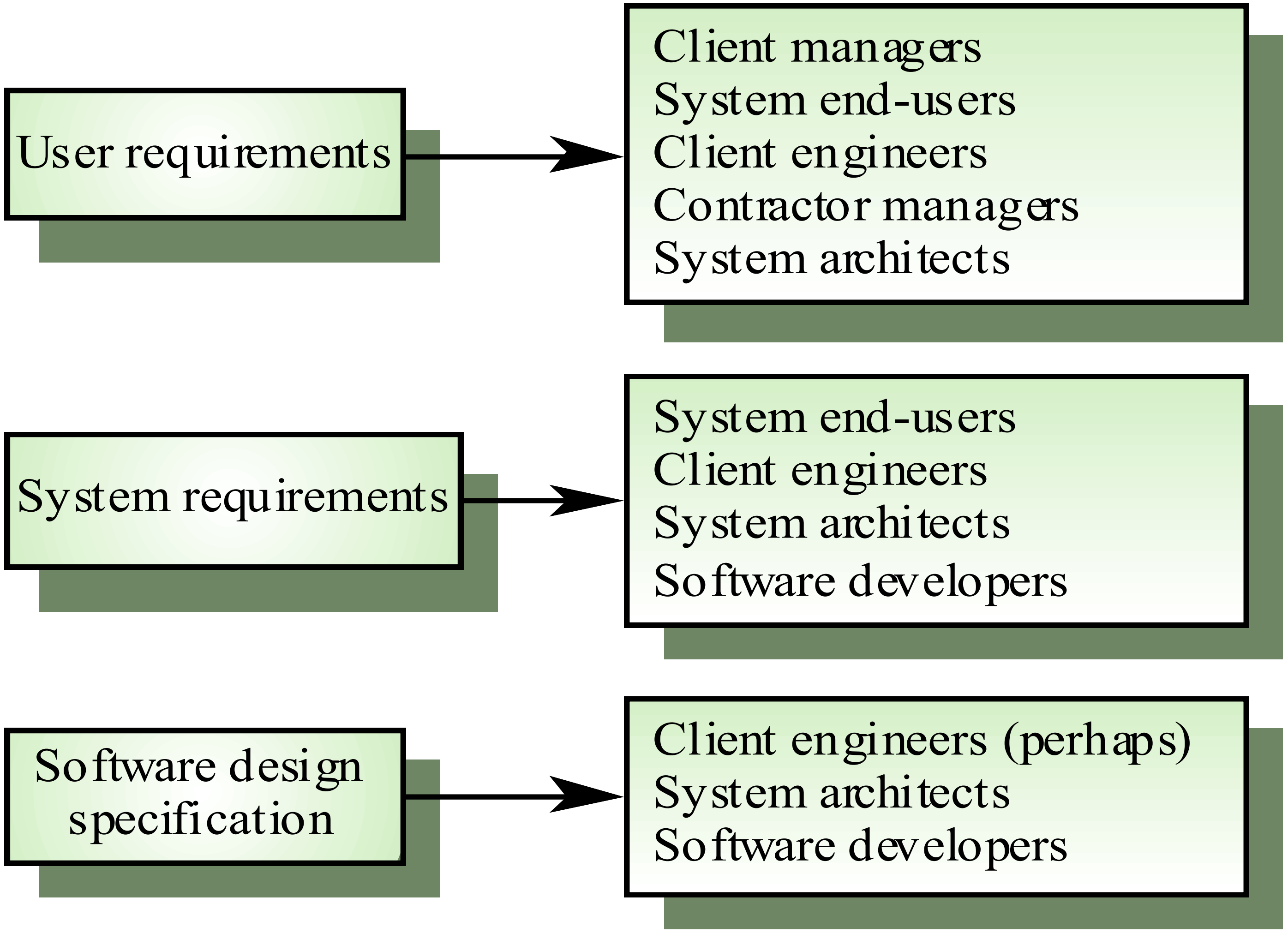
What is a software requirement?

- Its an Abstract description of the system should provide and the constraints under which the system must operate.
- It should only specify the external behaviour of the system.

Types of requirement

- User requirements
    - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers
- System requirements
    - A structured document setting out detailed descriptions of the system services. Written as a contract between client and contractor
- Software specification
    - A detailed software description which can serve as a basis for a design or implementation. Written for developers

Requirements readers



Functional and non-functional requirements

- Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- Non-functional requirements
  - constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

Functional requirements

- Describe functionality or system services
- Depend on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail

Function requirement specify what the product must do in order to specify the basic reason for its existence:

- Specifications of the products functionality.
- Actions that the product must take – check, compute, record and retrieve.
- Derived from the basic purpose of the product.
- Normally its business oriented, rather than technical
- Not measurable or testable at this stage.
- Not a quality.

Examples of functional requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Non-functional requirements (NFR)

- How a system must behave
- NFR specify all the remaining requirements not covered by the FUNCTIONAL REQUIREMENTS.
- Define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language or development method
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless

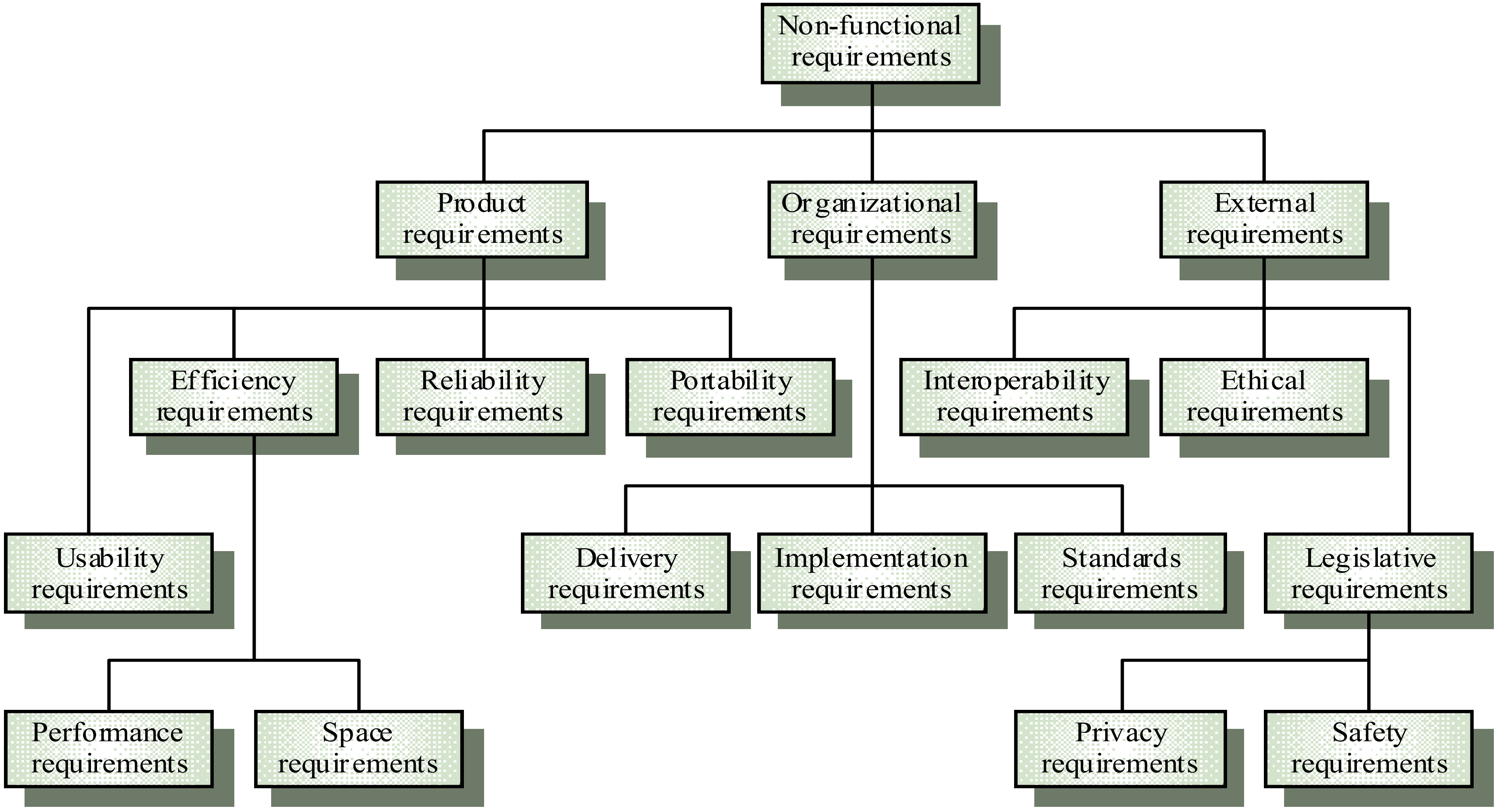
Examples of Non functional requirements the user may want that the product be:

- FAST (the response time be less than a specified time)
- ACCURATE (up to three places after decimal)
- USER FRIENDLY (the input screen be self explanatory)
- ATTRACTIVE.

Non-functional classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements etc.

Non-functional requirement types



User requirements

- Should describe functional and non-functional requirements so that they are understandable by system users who don’t have detailed technical knowledge
- User requirements are defined using natural language, tables and diagrams

Problems with natural language

- Lack of clarity
  - Precision is difficult without making the document difficult to read
- Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up

  - Several different requirements may be expressed together

Database requirement

The database shall support the generation and control of configuration objects; that is, objects which are themselves groupings of other objects in the database.

The configuration control facilities
shall allow access to the objects in a version group by the use of an incomplete name.

Requirements document requirements

- Specify external system behaviour
- Specify implementation constraints
- Easy to change
- Serve as reference tool for maintenance
- Record forethought about the life cycle of the system i.e. predict changes
- Characterise responses to unexpected events

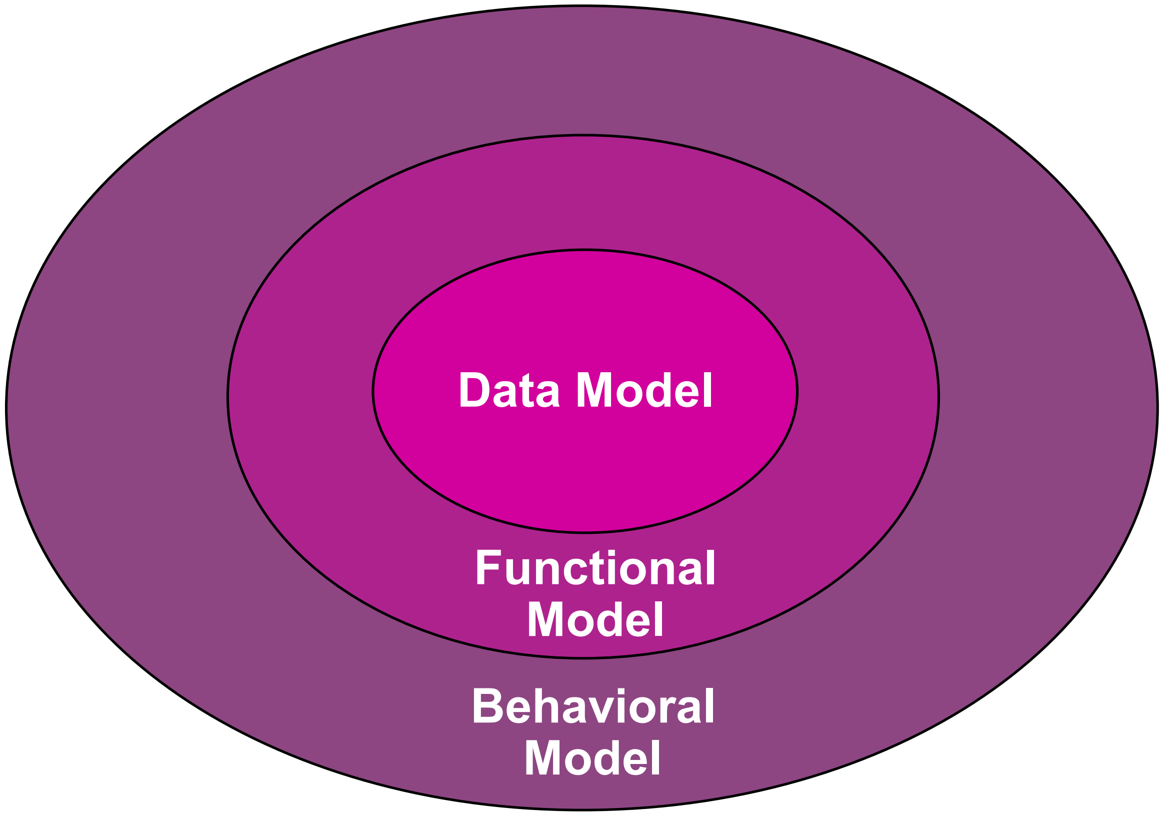
Software Requirements Specification Document

What is an SRS ?

Software Requirements specification (SRS) is a perfect detailed description of the behavior of the system to be developed.

The SRS document is an formal agreement between the developer and customer covering the functional and non-functional requirement of the software to be developed.

Systems Requirements Specification



The SRS is composed of the outer layer of the behavioral model, the functional model, then the data model.

Purpose of SRS document?

- SRS establishes basis of agreement between the user and the supplier.
  - Users needs have to be satisfied, but user may not understand software
  - Developers will develop the system, but may not know about problem domain

- SRS is
  - the medium to bridge the communications gap, and
  - specifies user needs in a manner both can understand

Need for SRS…

- Helps user understand his needs.
  - users do not always know their needs
  - must analyze and understand the potential
  - The requirement process helps clarify needs

- SRS provides a reference for validation of the final product
  - Clear understanding about what is expected.
  - Validation - “ SW satisfies the SRS “

Need for SRS…

- Helps user understand his needs.
  - users do not always know their needs
  - must analyze and understand the potential
  - The requirement process helps clarify needs

- SRS provides a reference for validation of the final product
  - Clear understanding about what is expected.
  - Validation - " SW satisfies the SRS "

Systems Requirements Specification
The main qualities of SRS document

Correct

Complete

Unambiguous

Verifiable

Consistent

Understandable

Modifiable

Traceable

Ranked for importance and/or stability

Systems Requirements Specification

Correct -

specifies every true requirement known at that time and no incorrect specifications - no wrong data

Unambiguous

each requirement has only one interpretation

Complete -

everything included behavior (methods, use cases, systems, subsystems, business rules) and data (objects, attributes

Systems Requirements Specification

Verifiable

is the software built what was specified in the SRS

Consistent

conflicting terms, characteristics, don't conflict with other requirement

Understandable

question: are formal specifications understandable, are informal specifications understandable

Systems Requirements Specification

Modifiable

changing requirements easily modified when specifying, designing, coding, implementing

Traceable

the origin od each requirement can be found

Design Independent

SRS should not specify a particular design

- Ranked for importance/stability
  - Needed for prioritizing in construction
  - To reduce risks due to changing requirements

Components of an SRS

- What should an SRS contain ?
  - Clarifying this will help ensure completeness
- An SRS must specify requirements on
  - Functionality
  - Performance
  - Design constraints
  - External interfaces

Functional Requirements

- Heart of the SRS document; this forms the bulk of the specification
- Specifies all the functionality that the system should support
- Outputs for the given inputs and the relationship between them
- All operations the system is to do
- Must specify behavior for invalid inputs too

Performance Requirements

- All the performance constraints on the software system
- Generally on response time , throughput etc => dynamic
- Capacity requirements => static

Design Constraints

- Factors in the client environment that restrict the choices
- Some such restrictions
  - Standard compliance and compatibility with other systems
  - Hardware Limitations
  - Reliability, fault tolerance, backup req.
  - Security

External Interface

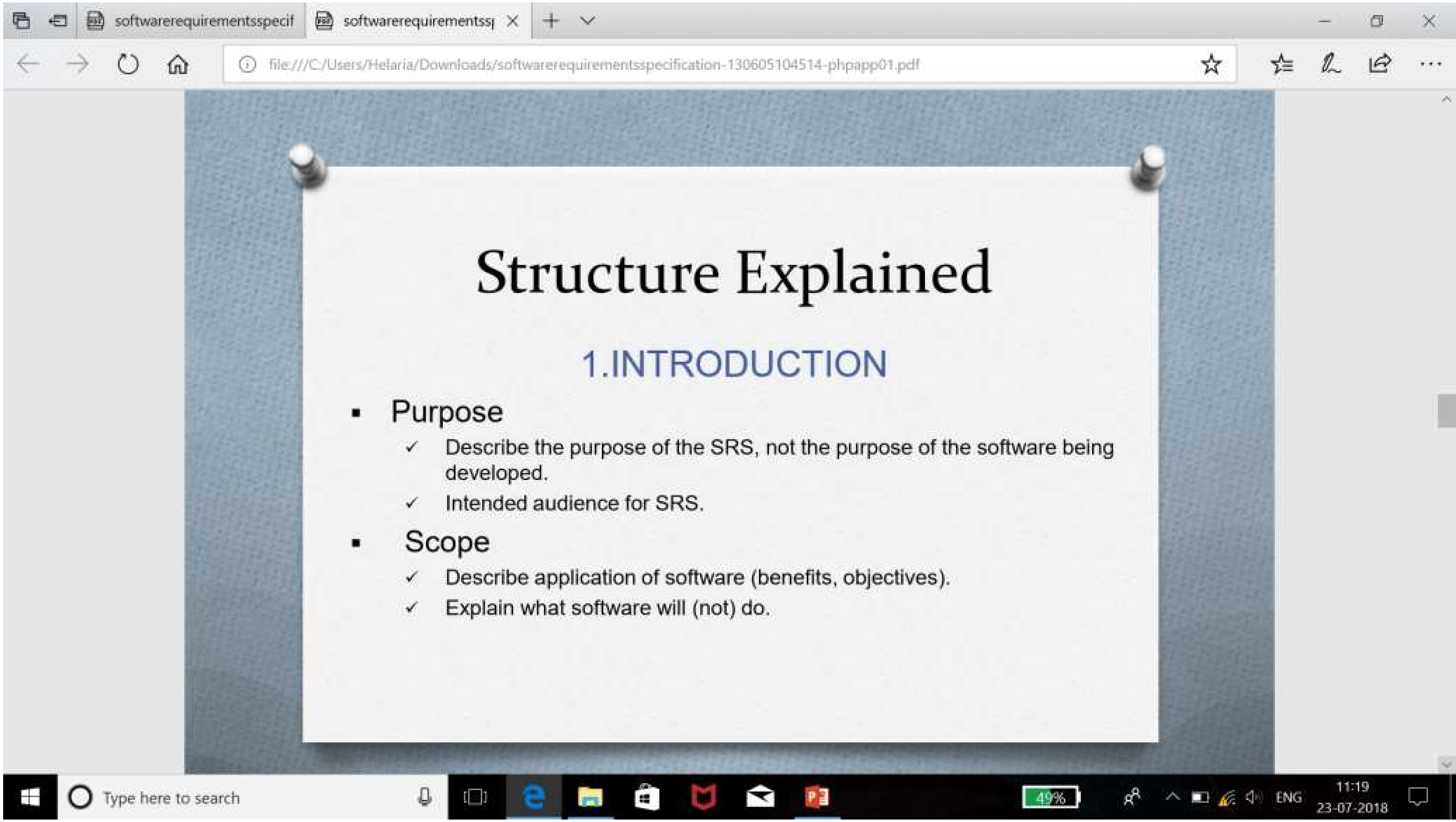
- All interactions of the software with people, hardware, and sw
- User interface most important
- General requirements of “friendliness” should be avoided
- These should also be verifiable

Requirements document structure

1. Introduction
    - 1.1 purpose of the requirement document
    - 1.2 Scope of the product
    - 1.3 definition and abbreviations
    - 1.4 References to supporting documents
    - 1.5 overview of rest SRS

2.General Description
    - 2.1 Product Perspective
    - 2.2 Product characteristics
    - 2.3 User characteristics
    - 2.4 General constraints
    - 2.5 Assumptions and Dependencies

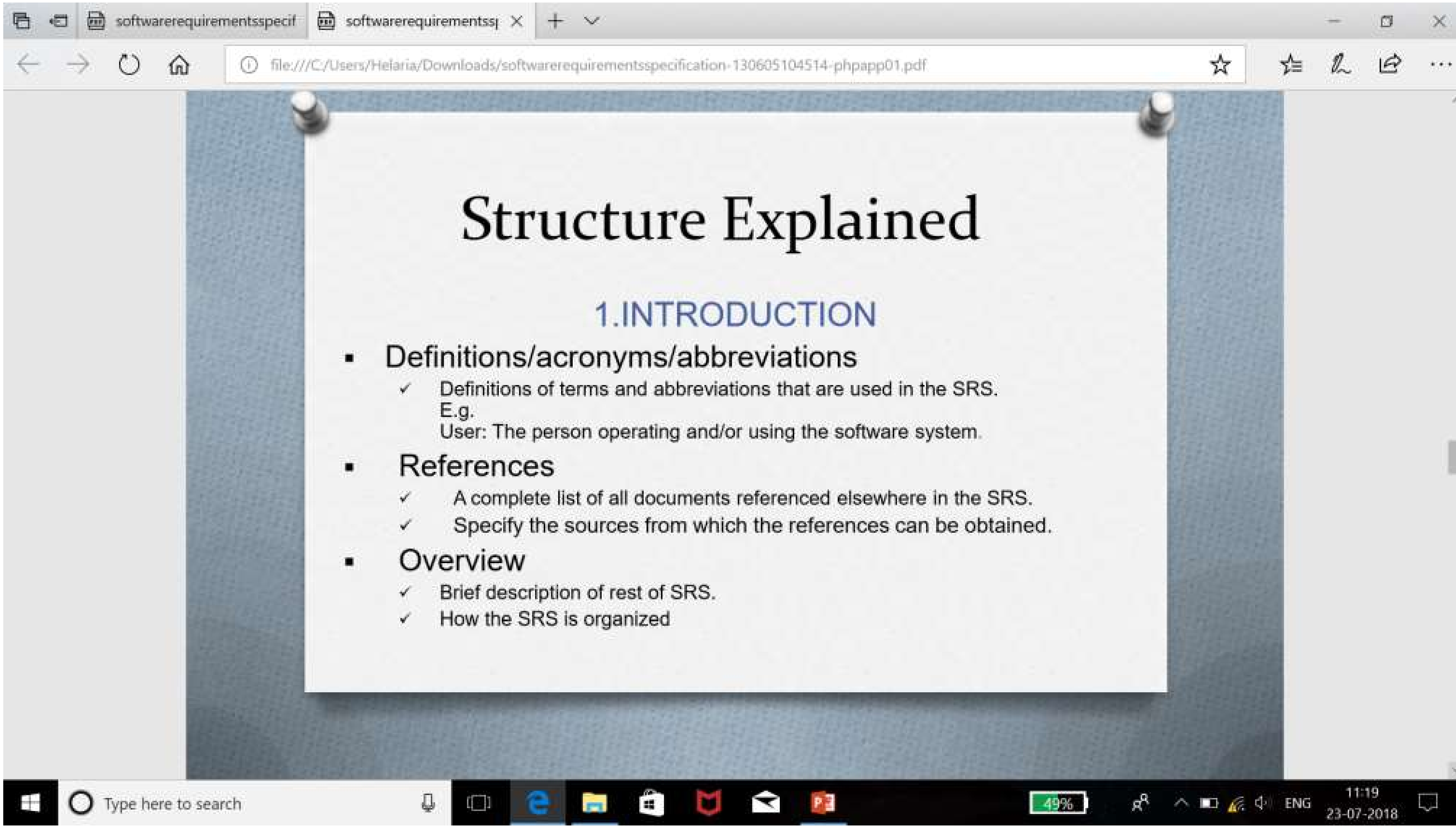
3. Functional Requirement
4. Non-Functional Requirements
5. System Architecture
6. System Models
7. Appendices
    - Data dictionary
    - Screenshots and reports
    - references links


softwarerequirementsspecif
softwarerequirementss
file:///C:/Users/Helaria/Downloads/softwarerequirementsspecification-130605104514-phpapp01.pdf
Type here to search
49%
ENG
11:19
23-07-2018

Structure Explained

1.INTRODUCTION

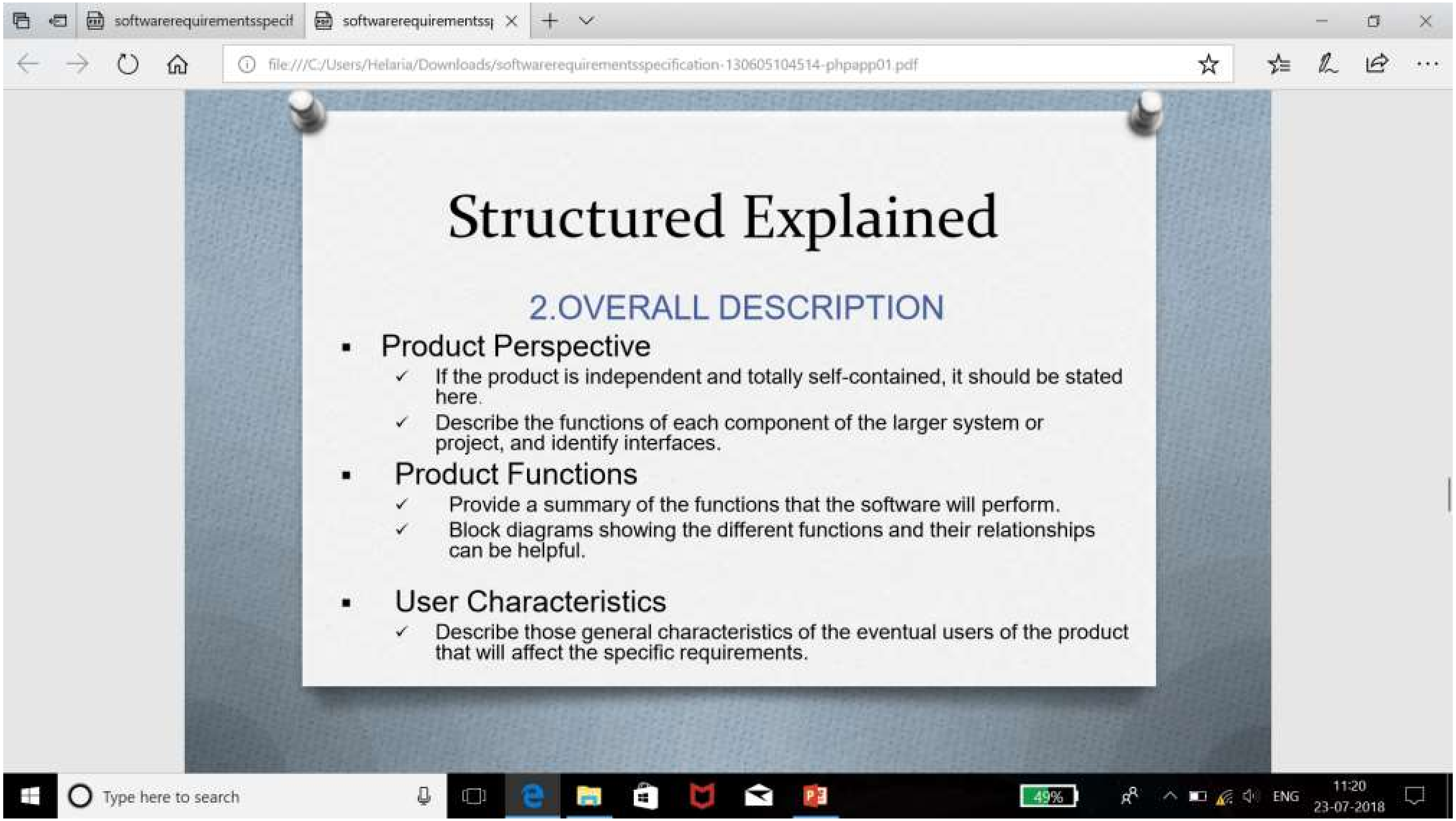
- Purpose
  - ✓ Describe the purpose of the SRS, not the purpose of the software being developed.
  - ✓ Intended audience for SRS.
- Scope
  - ✓ Describe application of software (benefits, objectives).
  - ✓ Explain what software will (not) do.


softwarerequirementsspecif
softwarerequirementss
file:///C:/Users/Helaria/Downloads/softwarerequirementsspecification-130605104514-phpapp01.pdf
Type here to search
49%
ENG
11:19
23-07-2018

Structure Explained

1.INTRODUCTION

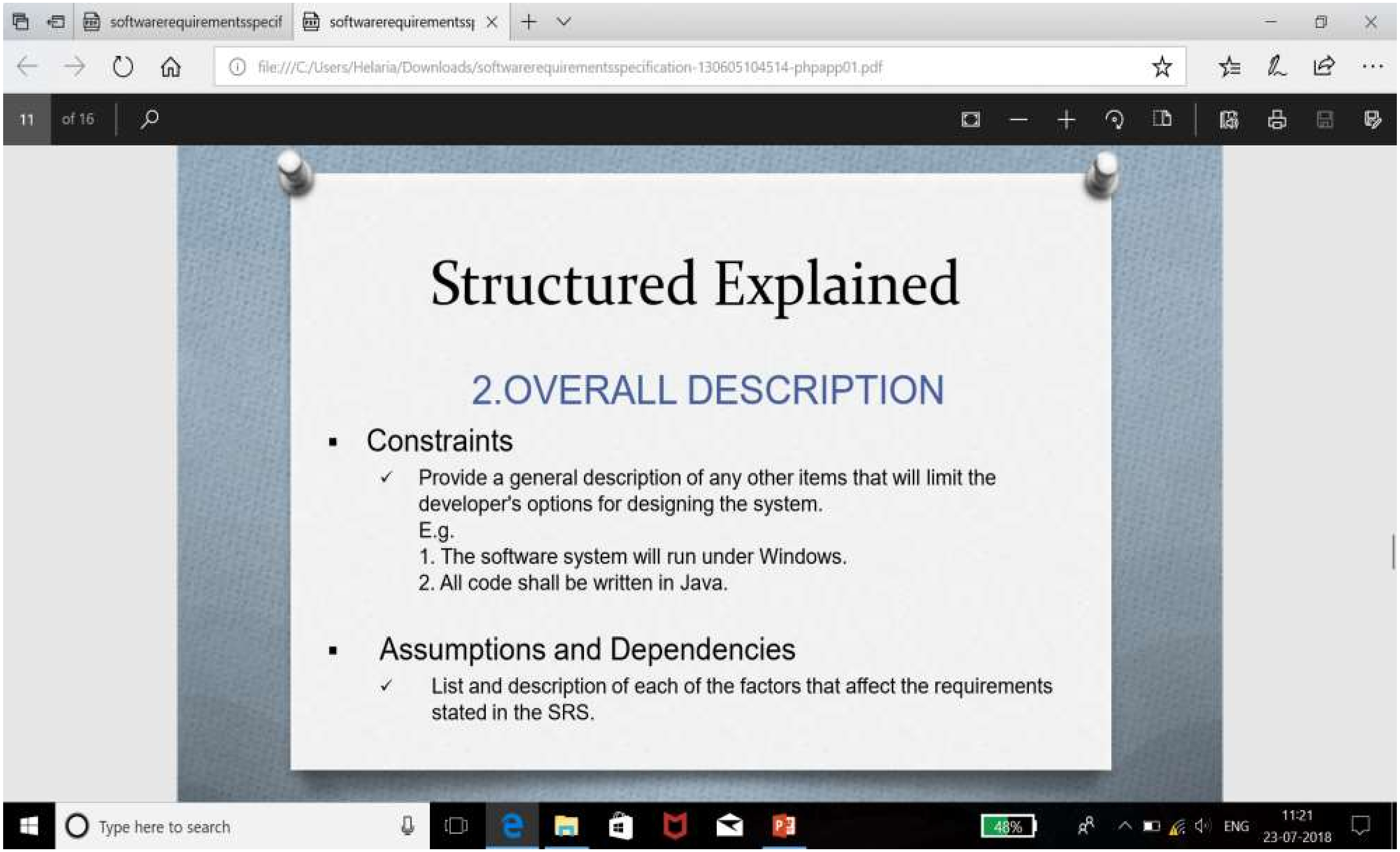
- Definitions/acronyms/abbreviations
  - ✓ Definitions of terms and abbreviations that are used in the SRS.
    E.g.
    User: The person operating and/or using the software system.
- References
  - ✓ A complete list of all documents referenced elsewhere in the SRS.
  - ✓ Specify the sources from which the references can be obtained.
- Overview
  - ✓ Brief description of rest of SRS.
  - ✓ How the SRS is organized


softwarerequirementsspecif
softwarerequirementss
file:///C:/Users/Helaria/Downloads/softwarerequirementsspecification-130605104514-phpapp01.pdf
Type here to search
49%
ENG
11:20
23-07-2018

Structured Explained

2.OVERALL DESCRIPTION

- Product Perspective
  - ✓ If the product is independent and totally self-contained, it should be stated here.
  - ✓ Describe the functions of each component of the larger system or project, and identify interfaces.
- Product Functions
  - ✓ Provide a summary of the functions that the software will perform.
  - ✓ Block diagrams showing the different functions and their relationships can be helpful.
- User Characteristics
  - ✓ Describe those general characteristics of the eventual users of the product that will affect the specific requirements.


softwarerequirementsspecif
softwarerequirementss
file:///C:/Users/Helaria/Downloads/softwarerequirementsspecification-130605104514-phpapp01.pdf
11
of 16
Type here to search
48%
ENG
11:21
23-07-2018

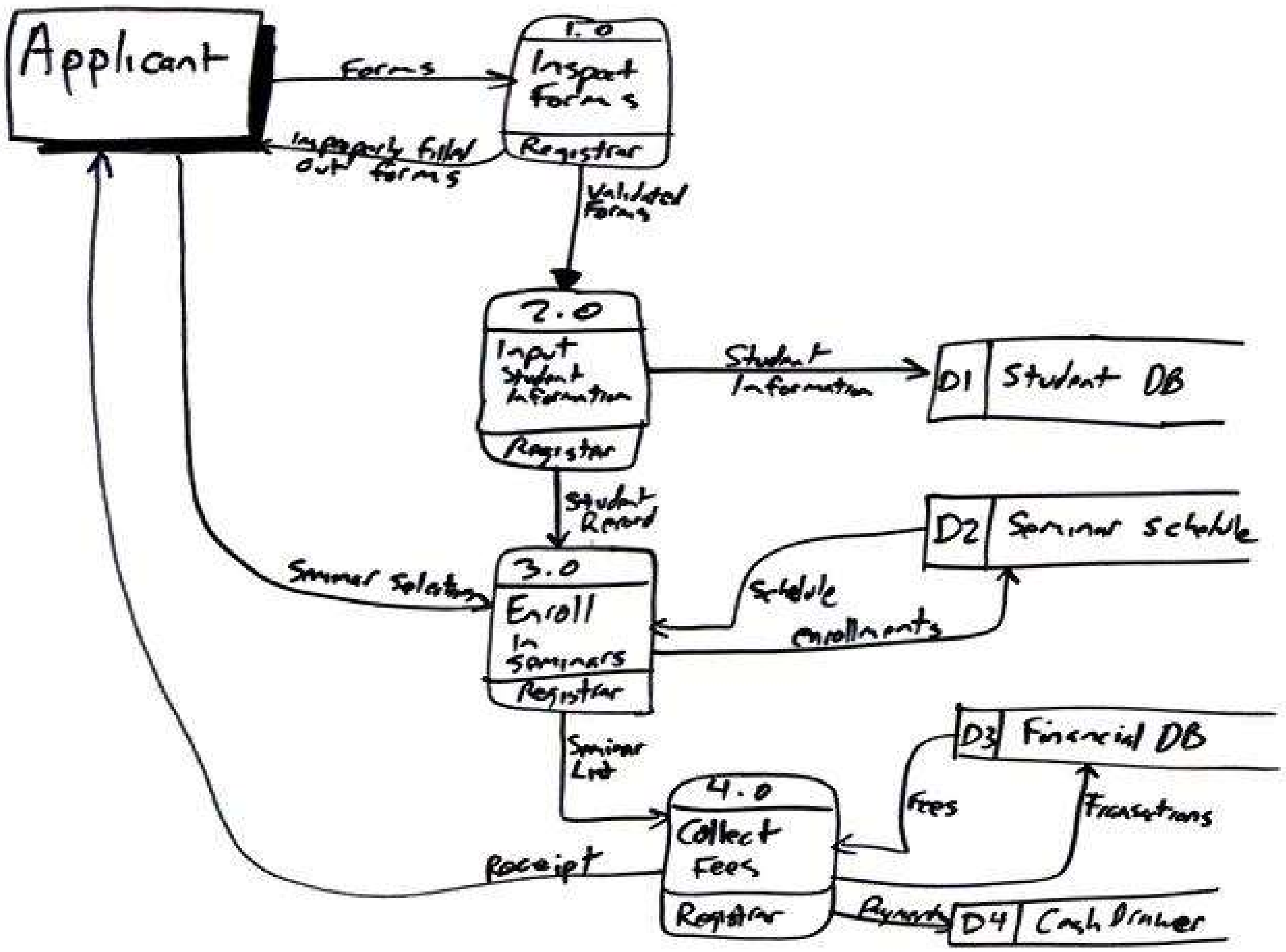
Structured Explained

2.OVERALL DESCRIPTION

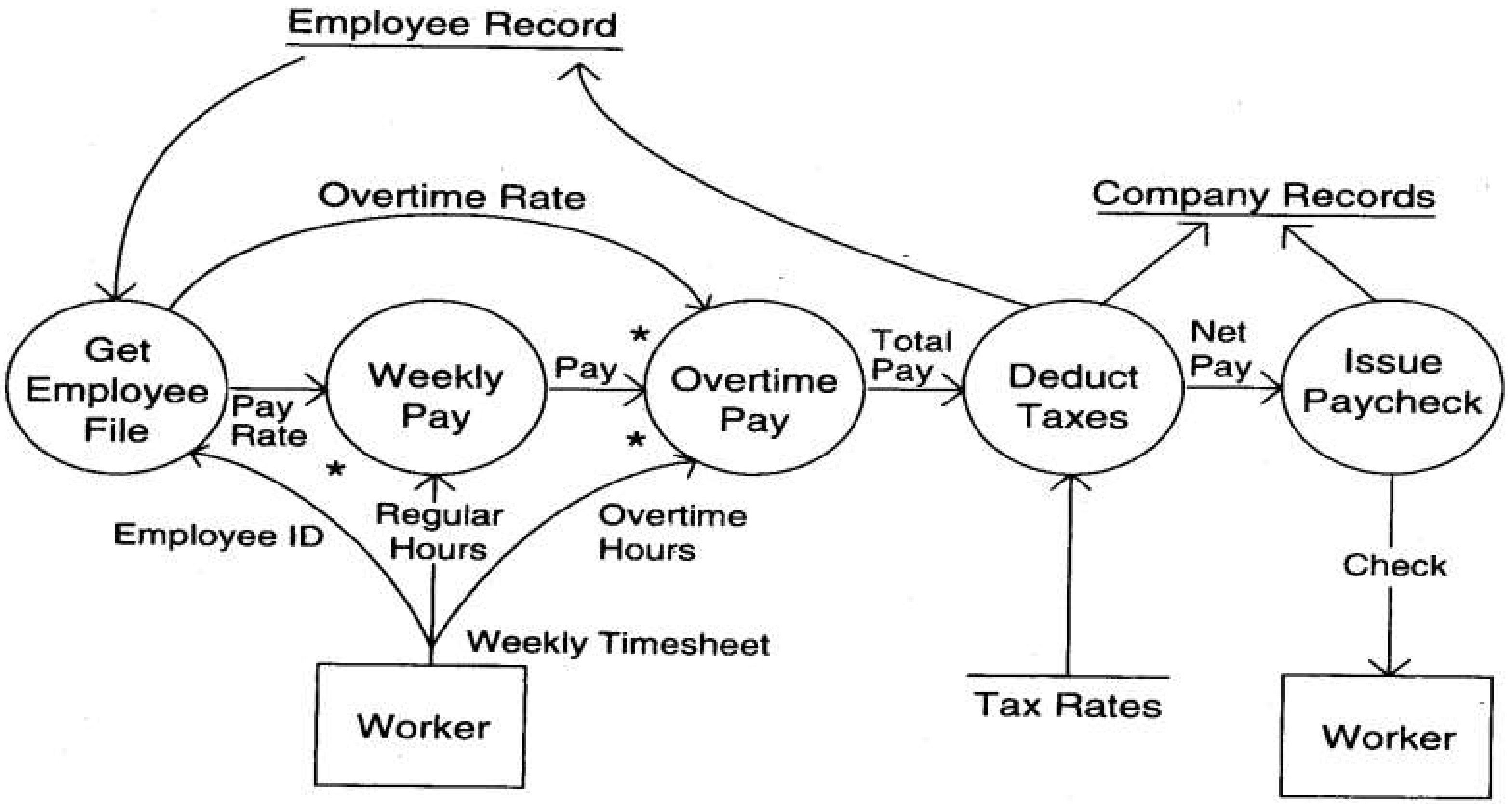
- Constraints
  - ✓ Provide a general description of any other items that will limit the developer's options for designing the system.
    E.g.
    1. The software system will run under Windows.
    2. All code shall be written in Java.

- Assumptions and Dependencies
  - ✓ List and description of each of the factors that affect the requirements stated in the SRS.

Example DFD: Enrolling in a University



DFD Example


Employee Record
Overtime Rate
Company Records
Get Employee File
Pay Rate
Weekly Pay
Pay
*
*
*
Overtime Pay
Total Pay
Deduct Taxes
Net Pay
Issue Paycheck
Employee ID
Regular Hours
Overtime Hours
Weekly Timesheet
Check
Worker
Tax Rates
Worker

REQUIREMENTS ENGINEERING PROCESS (REP)

- The processes used for requirements engineering vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- The goal of this stage of the software engineering process is to help create and maintain a system requirements document.

REQUIREMENTS ENGINEERING PROCESS (REP)

- REP involves creating and maintaining system requirements document

- REP is divided into 4

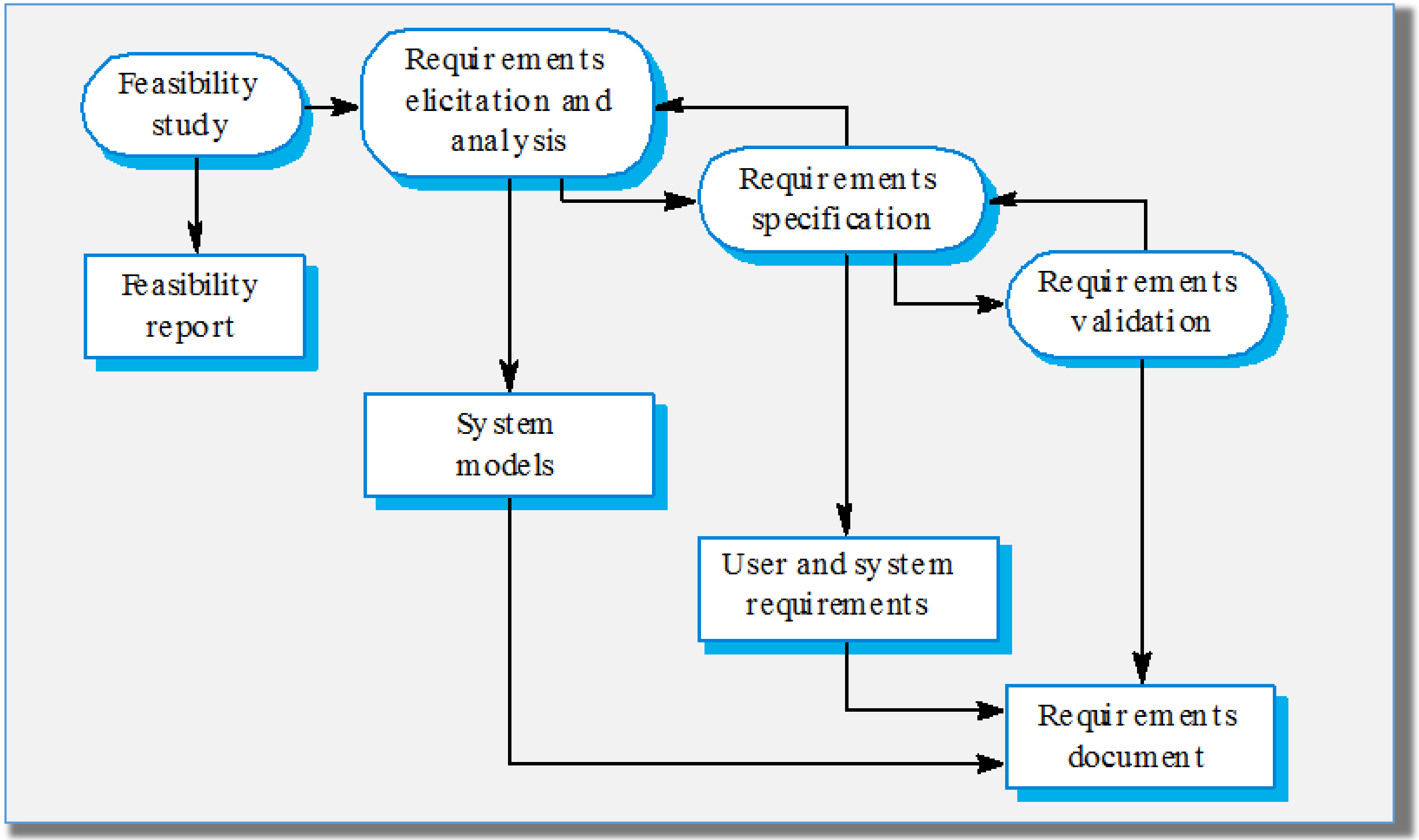
  Feasibility study

  Requirement Elicitation and analysis

  Requirement Specification

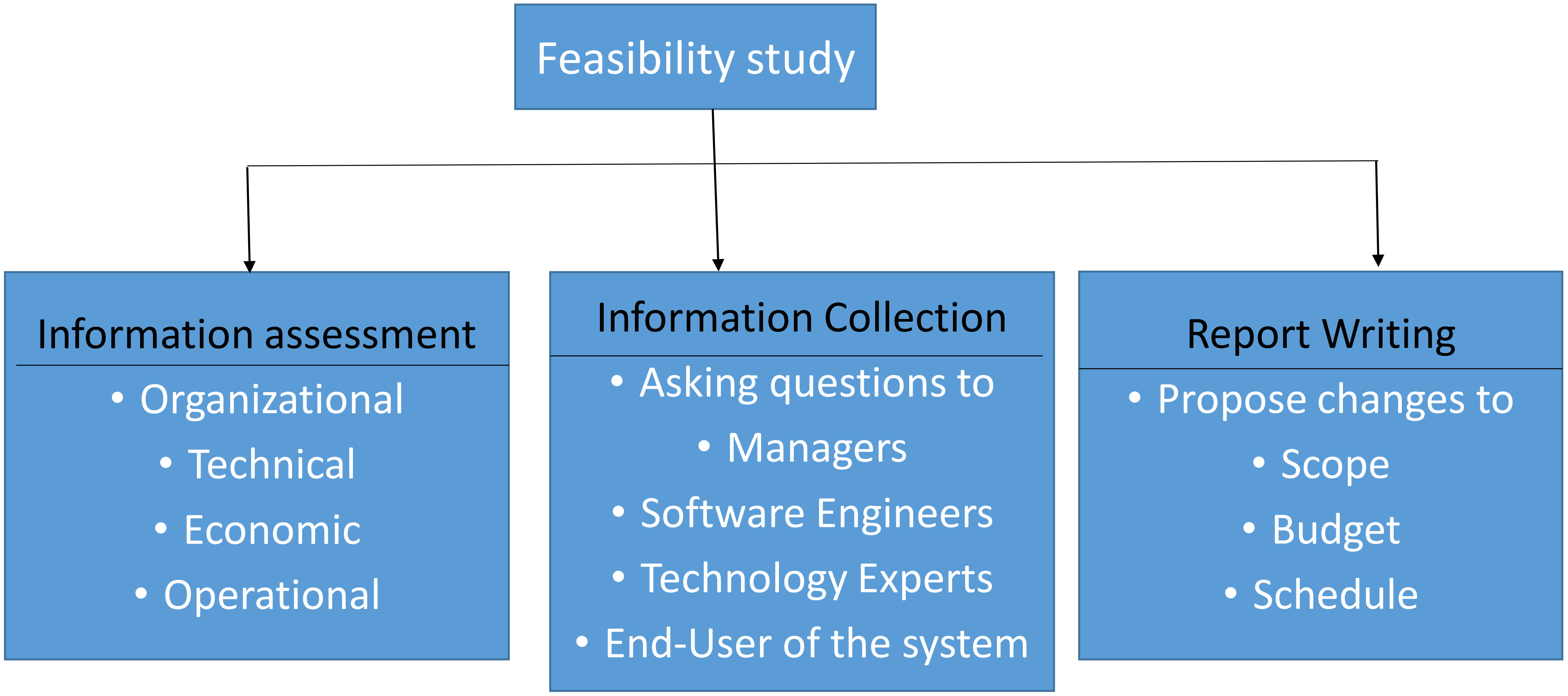
  Requirement validation

The Requirements Engineering Process



Feasibility Studies

- A feasibility study is to access whether the proposed system is economically and technically feasible
- The result of a feasibility study would be a report that recommends the feasibility of carrying on with the requirement engineering process.

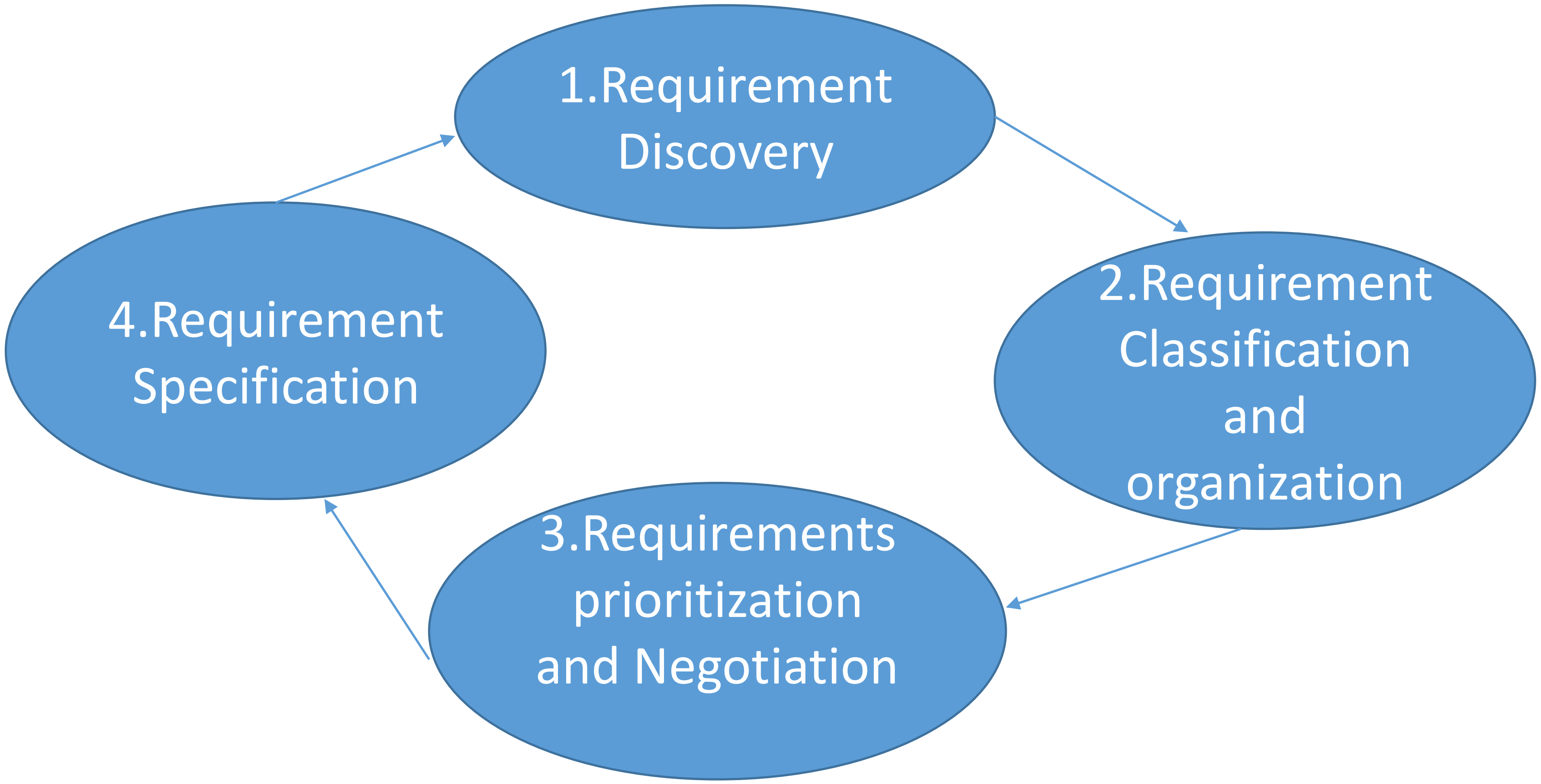

Feasibility study
Information assessment
• Organizational
• Technical
• Economic
• Operational
Information Collection
• Asking questions to
• Managers
• Software Engineers
• Technology Experts
• End-User of the system
Report Writing
• Propose changes to
• Scope
• Budget
• Schedule

Feasibility Study Implementation

- Based on information assessment (what is required), information collection and report writing.
- Questions for people in the organisation
  - What if the system wasn’t implemented?
  - What are current process problems?
  - How will the proposed system help?
  - What will be the integration problems?
  - Is new technology needed? What skills?
  - What facilities must be supported by the proposed system?

Elicitation and Analysis

- Sometimes called requirements **elicitation or requirements discovery.**
- Involves technical staff working with customers to find out about the **application domain**, **the services that the system should provide** and the **system's operational constraints**.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders.*


1.Requirement Discovery
2.Requirement Classification and organization
3.Requirements prioritization and Negotiation
4.Requirement Specification

TECHNIQUES USED:

View Point oriented Elicitation (3 types of generic view point)

Interactor

Viewpoint of people or system That directly interact with the System

indirect

Viewpoint of the stake holders who donot use the sys themselves but who influence the requirements

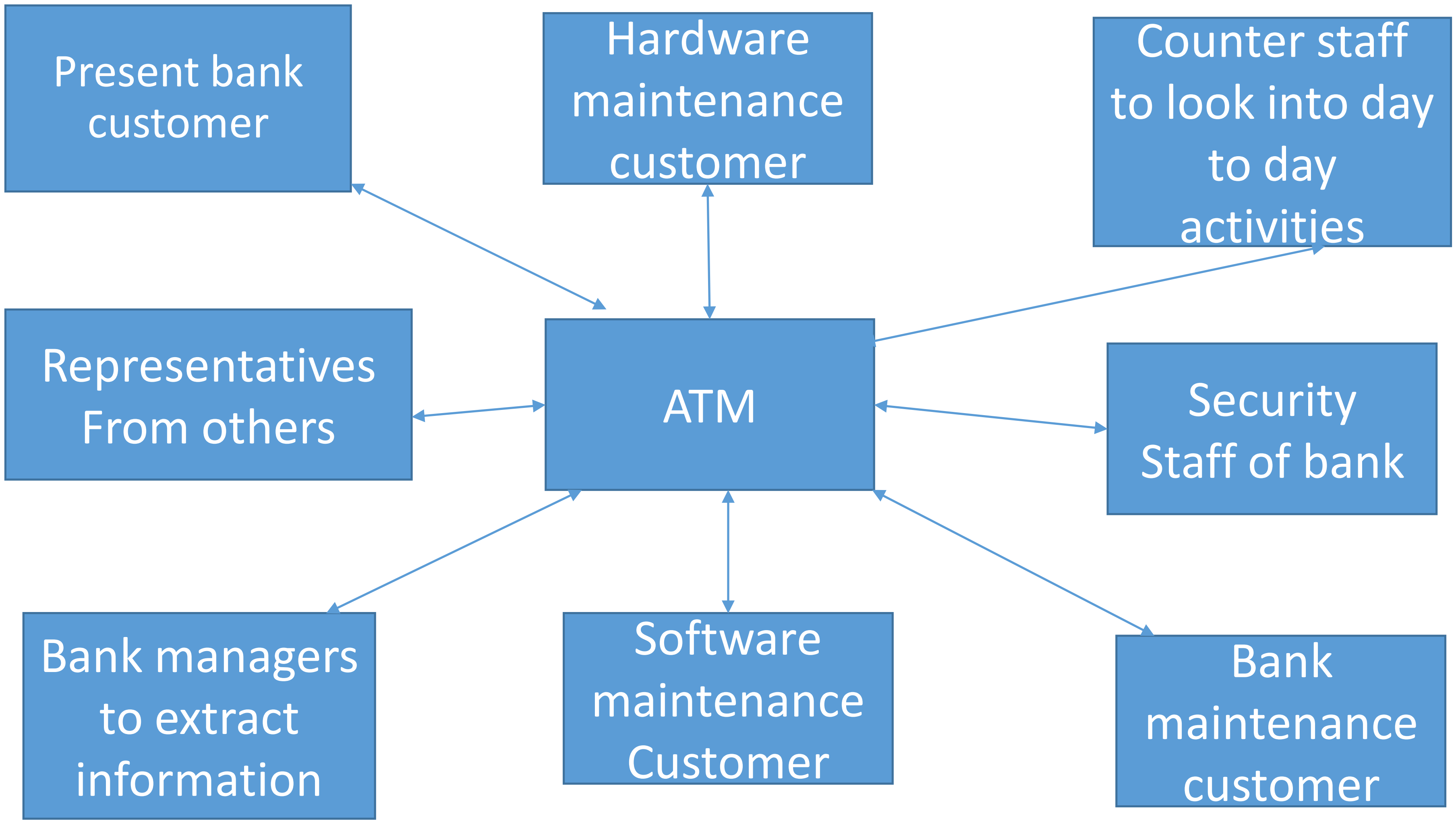
Domain

Represent the characteristics and Constraints of the domain which influence the sys requirement

Problems of Requirements Analysis

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements
  - Example, staff → easy of use, management → highest security
  - Patients → change appointments easily, management → plan staff resourcing, reduce costs
- Organisational and political factors may influence the system requirements (Data protection)
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change.

Example of Viewpoint


Present bank customer
Hardware maintenance customer
Counter staff to look into day to day activities
Representatives From others
ATM
Security Staff of bank
Bank managers to extract information
Software maintenance Customer
Bank maintenance customer

Interviewing

- In formal or informal interviewing, the RE team puts questions to stakeholders about the system that they use and the system to be developed.
- There are two types of interview
  - Closed interviews where a pre-defined set of questions are answered.
  - Open interviews where there is no pre-defined agenda and a range of issues are explored with stakeholders.
- Ideally, interviewers should be open-minded, willing to listen to stakeholders and should not have pre-conceived ideas.

Scenarios

- There are effectively test cases running in a given situation
- So for example:
  - Try and withdraw cash with stolen credit card
  - Try and withdraw cash but machine has low cash stock
  - Withdraw cash with card number 3456123245677
  - Etc.
- Scenarios are very important as they
  - Show the developer by example what will happen given certain conditions
  - They can be used as a basis to test the software
  - Make things very clear and reduce ambiguity

Ethnography

- In ethnography, a social scientist spends a considerable amount of time observing and analysing how people **actually work**.
- People do not have to explain or articulate their work.
- Social and organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Focused Ethnography

- Developed in a project studying the air traffic control process
- Combines ethnography with prototyping
- Prototype development results in unanswered questions which focus the ethnographic analysis.
- The problem with ethnography is that it studies existing practices which may have some historical basis which is no longer relevant.

Scope of Ethnography

- Requirements that are derived from the way that people actually work rather than the way in which process definitions suggest that they ought to work.
  - People may have “short cuts” or use their previous knowledge and experience to better perform their role which may not be evident.
- **As an example**, an air traffic controller may switch off a conflict alert alarm detecting flight intersections. Their strategy is to ensure these planes are moved apart before problems arise and the alarms can distract them.

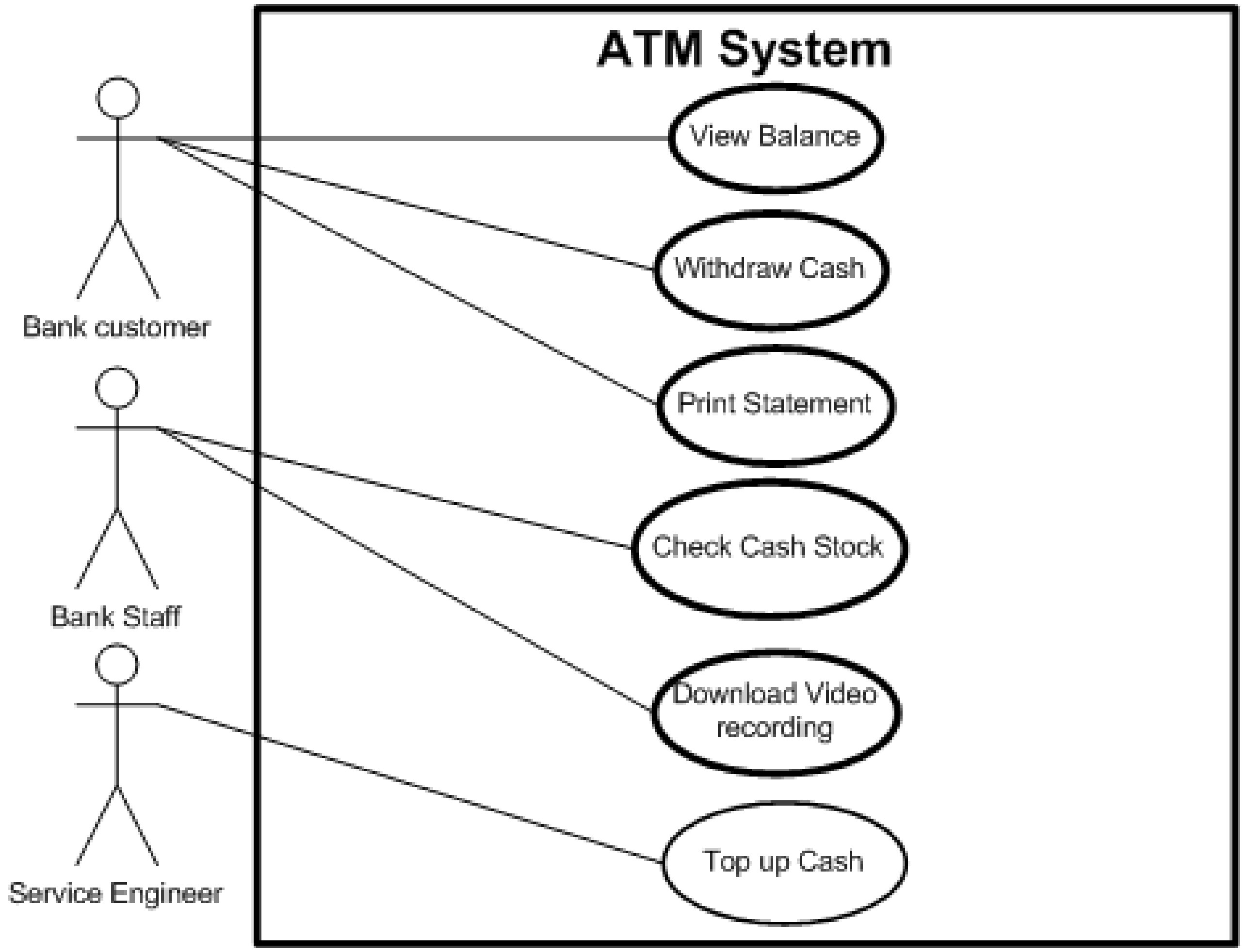
Scope of Ethnography

- Requirements that are derived from cooperation and awareness of other people’s activities.
  - People do not work in isolation and may share information and use dialogue with colleagues to inform decisions.
- **Using the previous scenario**, air traffic controllers may use awareness of colleagues work to predict the number of aircraft entering their sector and thus require some visibility of adjacent sector.

ATM machine

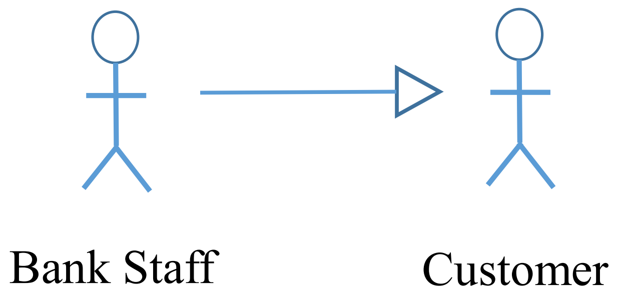
- Actors
  - Customers
  - Bank staff
  - ATM service engineer
- Use cases
  - Withdraw cash
  - Check balance
  - Add cash to machine
  - Check security video recording

Example - ATM Use Case Diagram



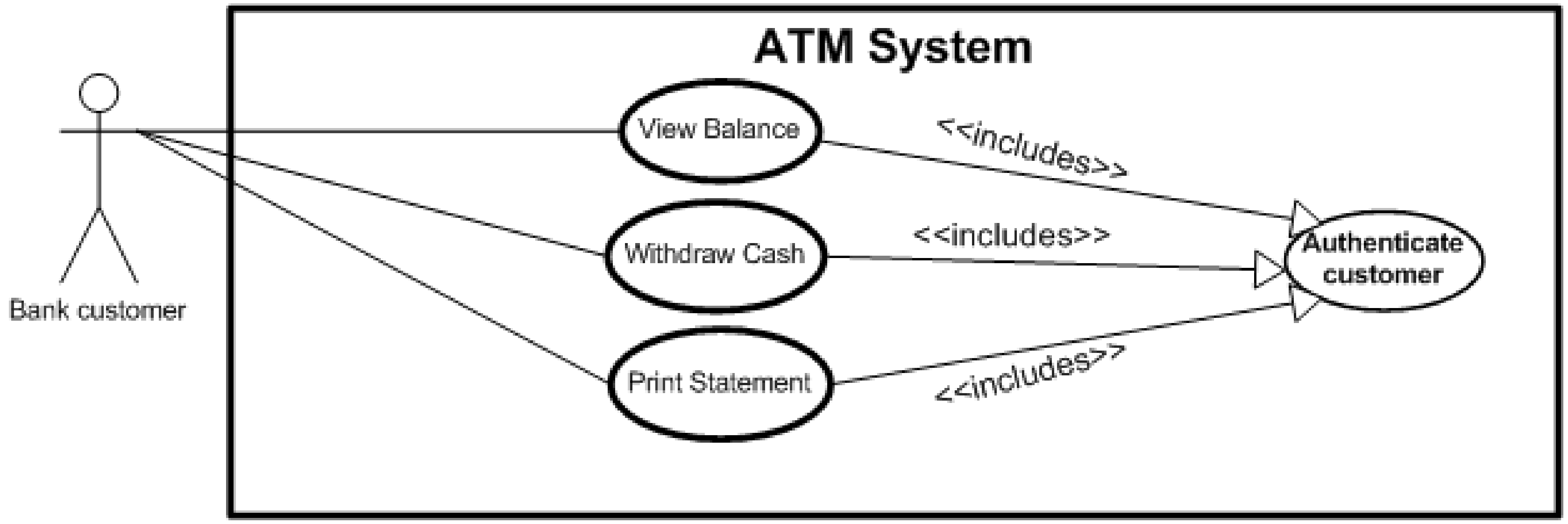
Advanced Use Case Diagrams

- **Inheritance** can be used between actors to show that all use cases of one actor are available to the other:



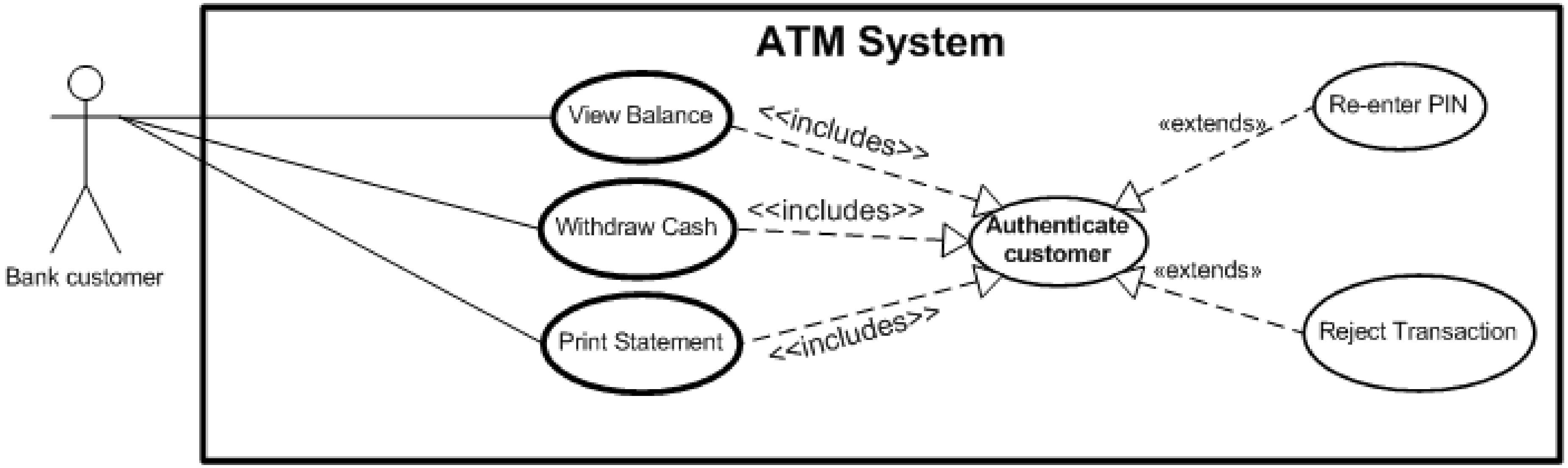
Include Relations

- 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.



SOFTWARE SYSTEM MODELING

- System models – Abstract descriptions of systems whose requirements are being analysed
- SYSTEM MODELS ARE GRAPHICAL representation that describes business processes, the problem to be solved and the system that is to be developed.

Objectives

- To explain why the context of a system should be modelled as part of the requirements engineering process
- To describe behavioural modelling, data modelling and object modelling
- To introduce some of the notations used in the Unified Modeling Language (UML)
- To introduce formal methods and formal modeling approaches

SYSTEM PERSPECTIVE

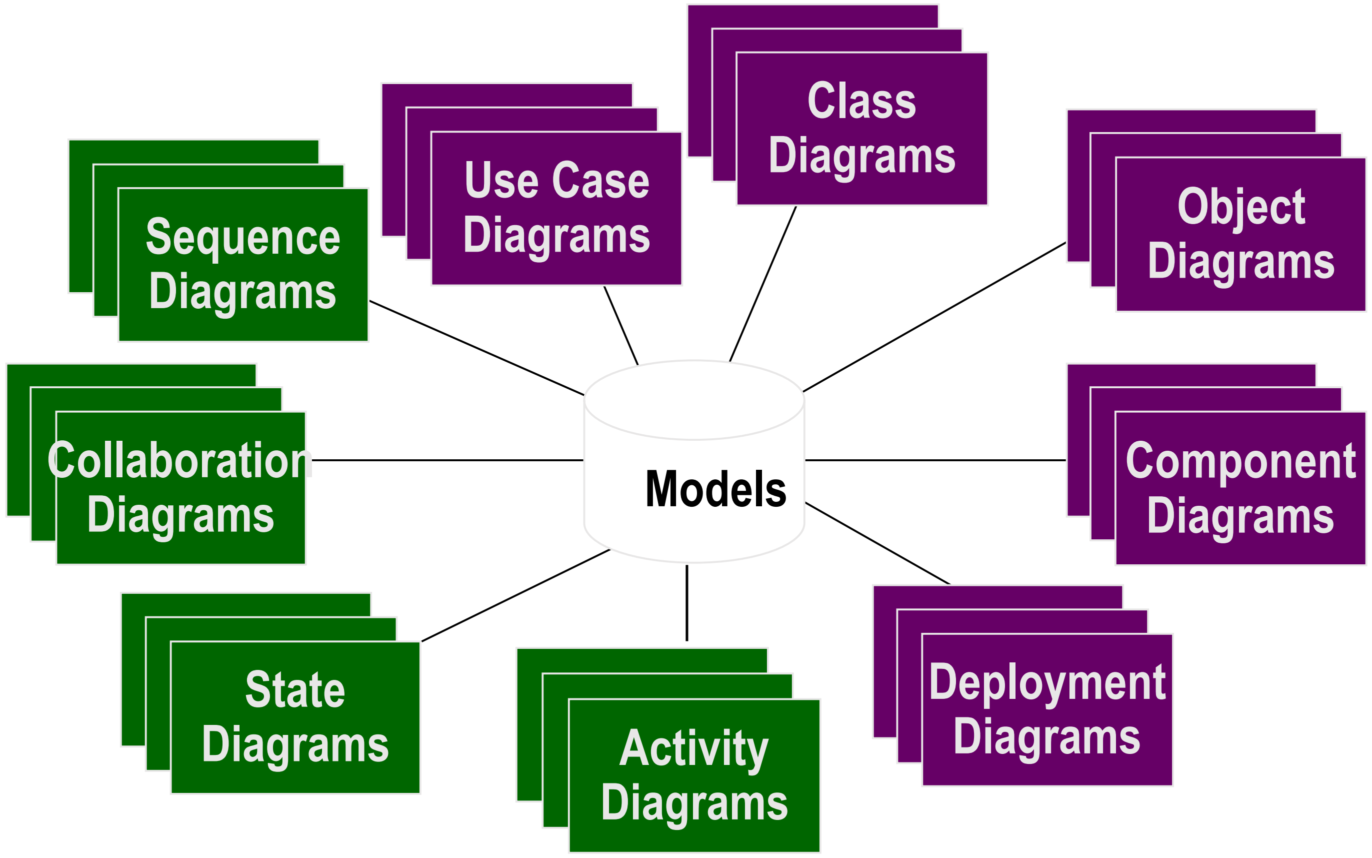
- Different models present the system from different perspective
- An External Perspective ------- shows the sys environment
- An Behavioral Perspective------ shows the dynamic behavior of the sys
- An Structural Perspective------ architecture of the sys is modelled
- An Interaction Perspective------ shows the interaction between system and its environment.

Different types of model (based on abstraction)

1. Data flow model : shows how the data is processed at different stage
2. Composition model : Shows how entities are composed of other entities this is also known as aggregation.
3. Architectural model : shows the principal sub-system that makes sys
4. Classification model : Also known as Inheritance, shows how entities have common characteristics
5. Stimulus – Response model : shows how the system reacts to internal or external events.

The Unified Modeling Language Diagram types:

- UML is a modelling language for object oriented modelling:



Software modeling and models

- Software modeling helps the engineer to understand the functionality of the system
- Models are used for communication among stakeholders
- Different models present the system from different perspectives
  - External perspective showing the system’s context or environment
  - Process models showing the system development process as well as activities supported by the system
  - Behavioural perspective showing the behaviour of the system
  - Structural perspective showing the system or data architecture

DIFFERENT TYPES OF SYSTEM MODELS

Context model ---------------- process model

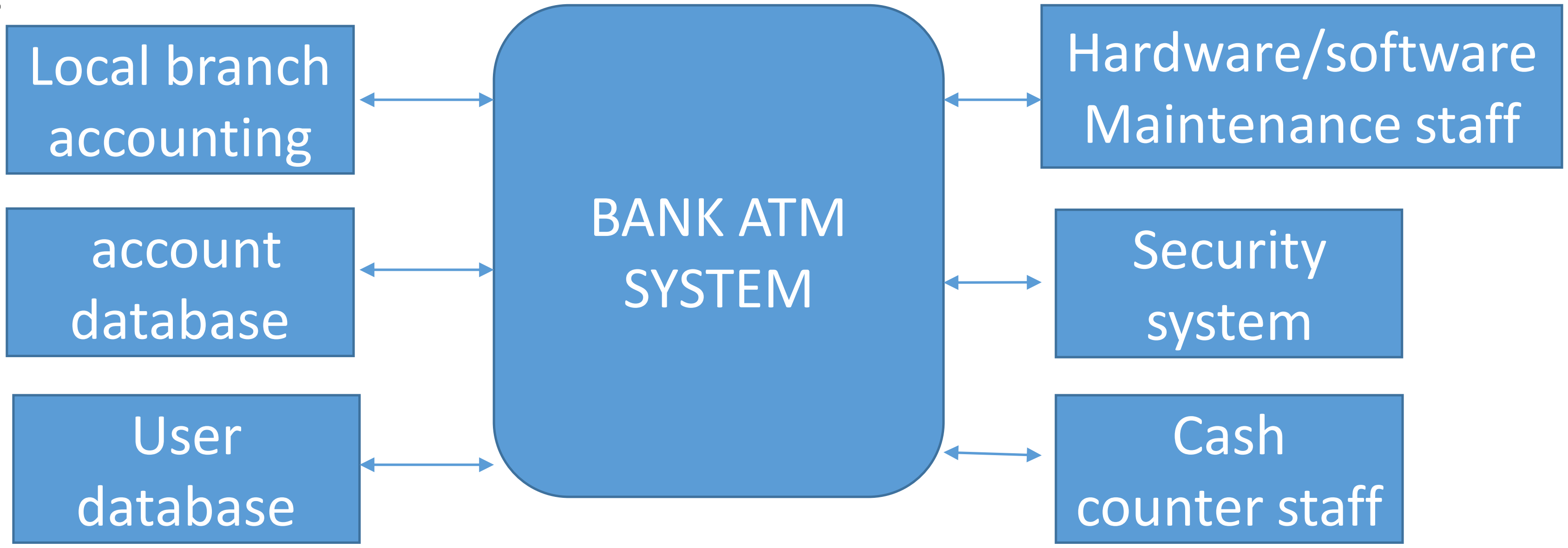
Behavioral model ---------------- dataflow model and state machine model

Data model ------------------------ data dictionaries

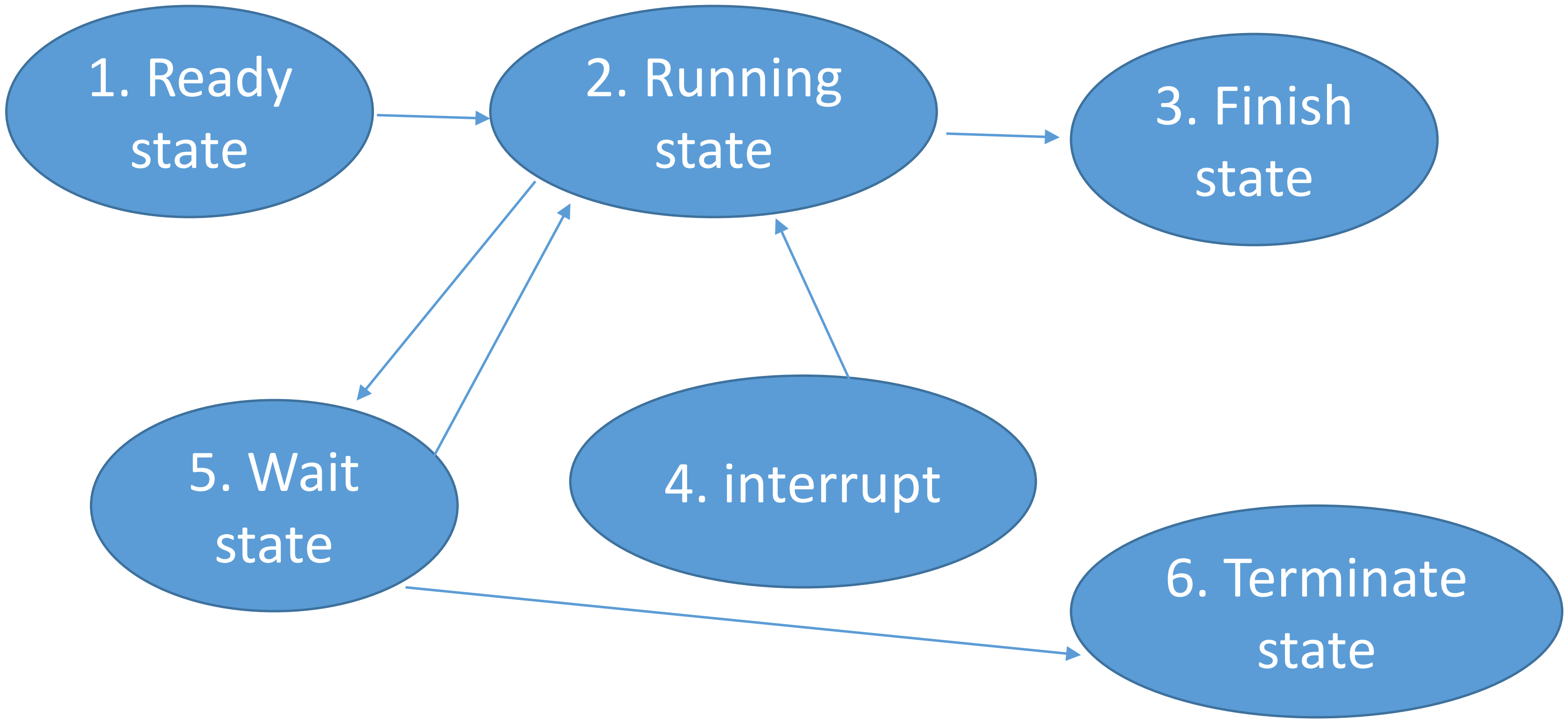
Object model ----------------------- Inheritance model, object aggregation and object behavior model

CONTEXT MODEL

- Its an architectural model and helps to put boundary of the system
- If the boundary is not clear then it may pose technical and managerial problems.
- After the boundary is defined, the dependencies of the system on its environment can be clarified
- Ex:



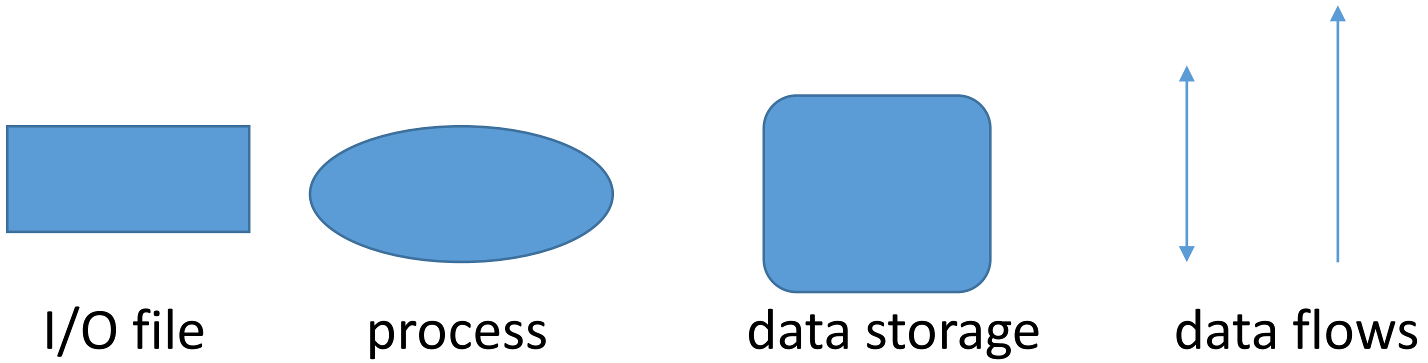
Process Model



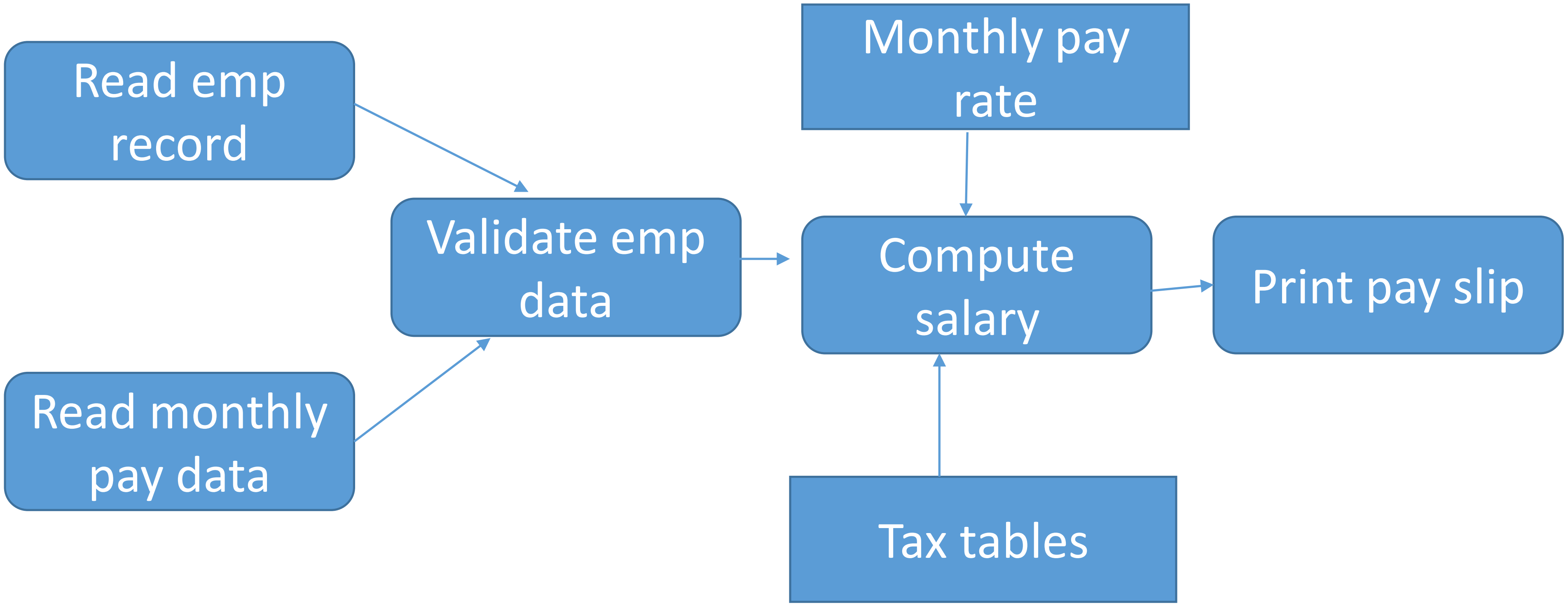
BEHAVIORAL MODEL

- DATA FLOW MODEL : easy for the user to understand , Clearly represents how data is represented.

- Symbol are:

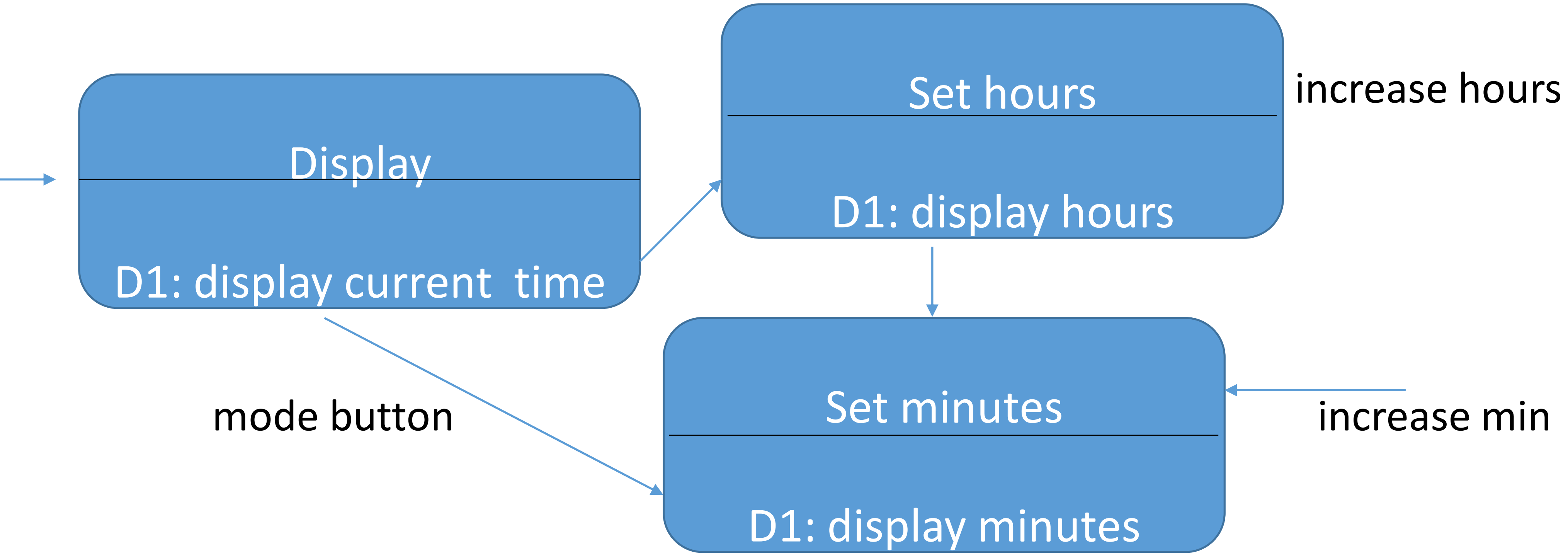


DFD for pay roll system



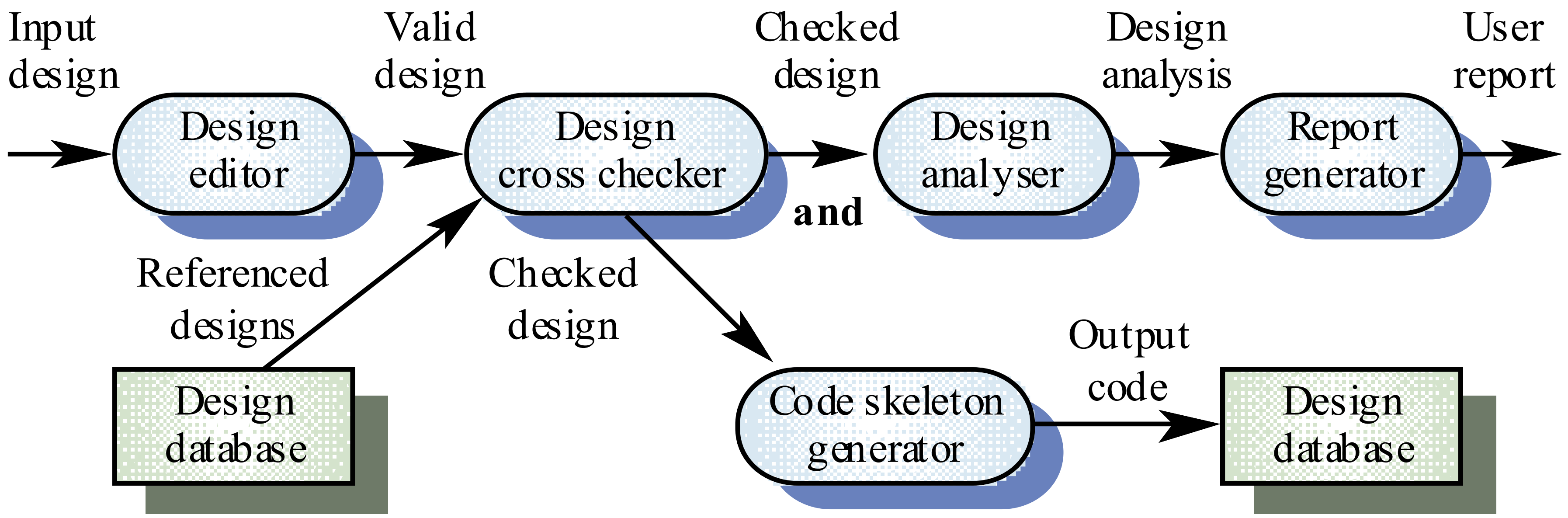
State machine model / Event model

- It describes how the system reacts to events. Events model represent their behaviour.
- This model is used for modelling real time system



Behavioral models – Data Processing

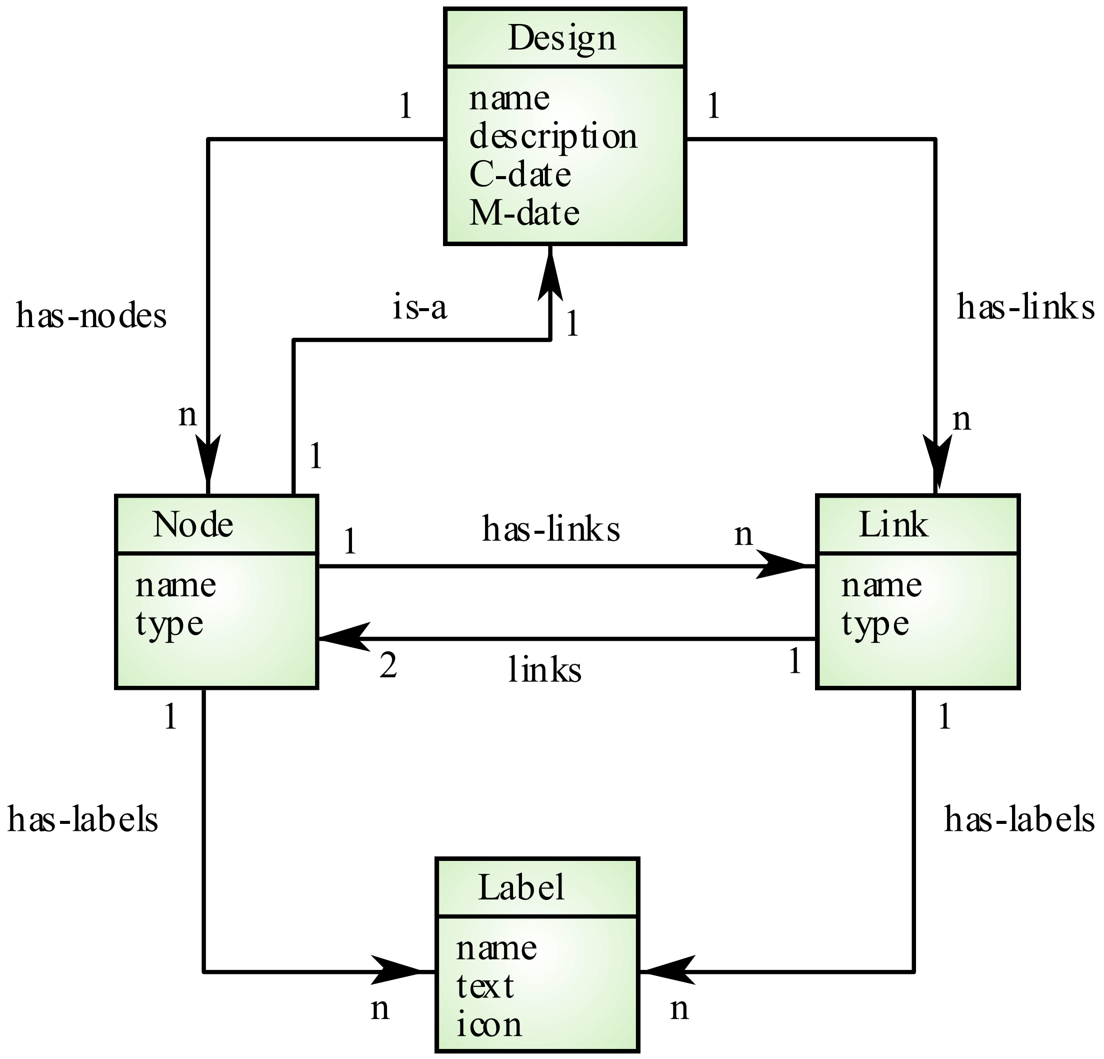
CASE toolset data flow diagram (DFD)



DATA MODEL

- Data models are the logical structure of the data.
- Data model show system entities, their attributes (properties) and the relationships between them.
- This is called ER model its used in database design and relational database schemas.

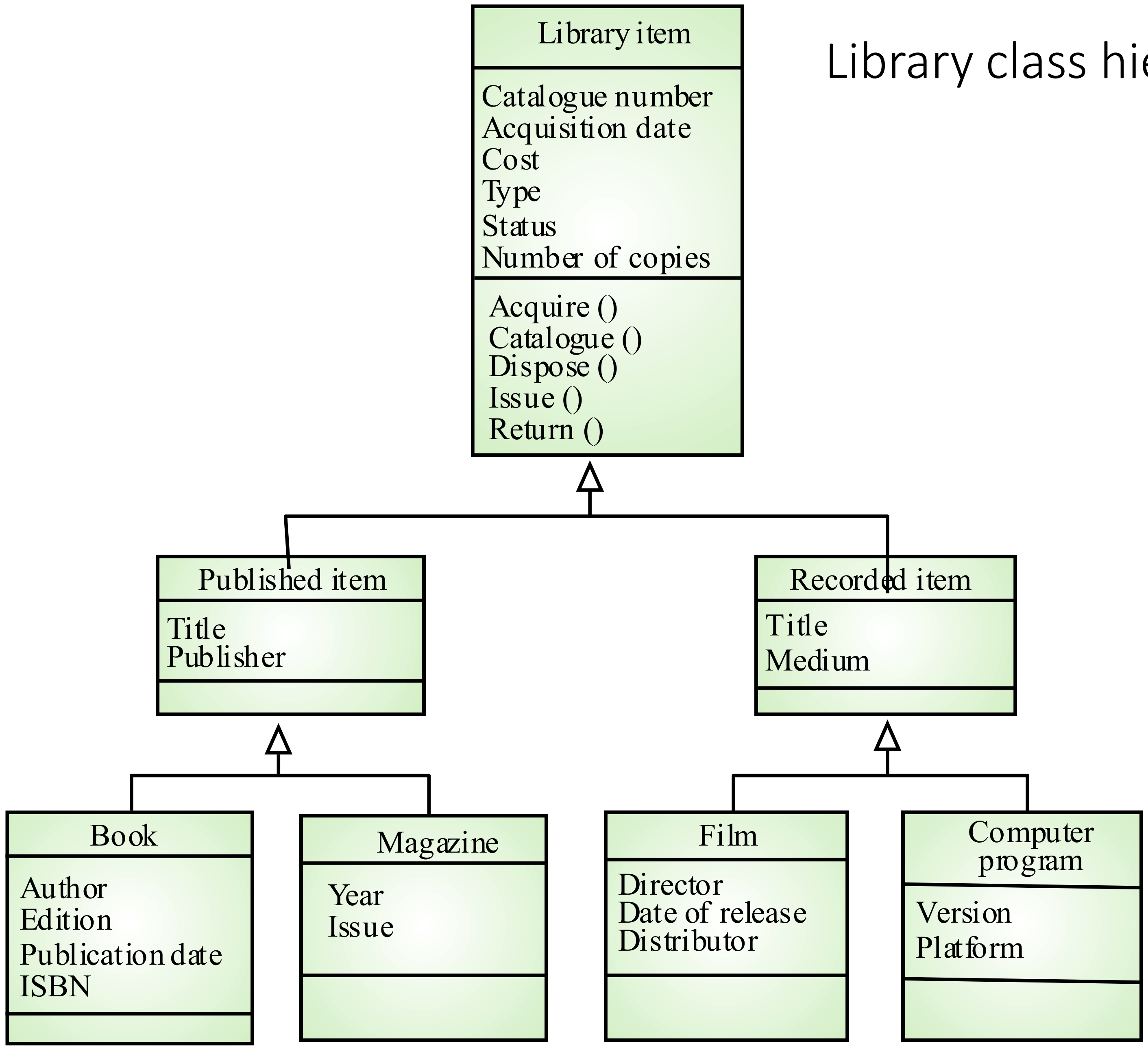
Data or ER models



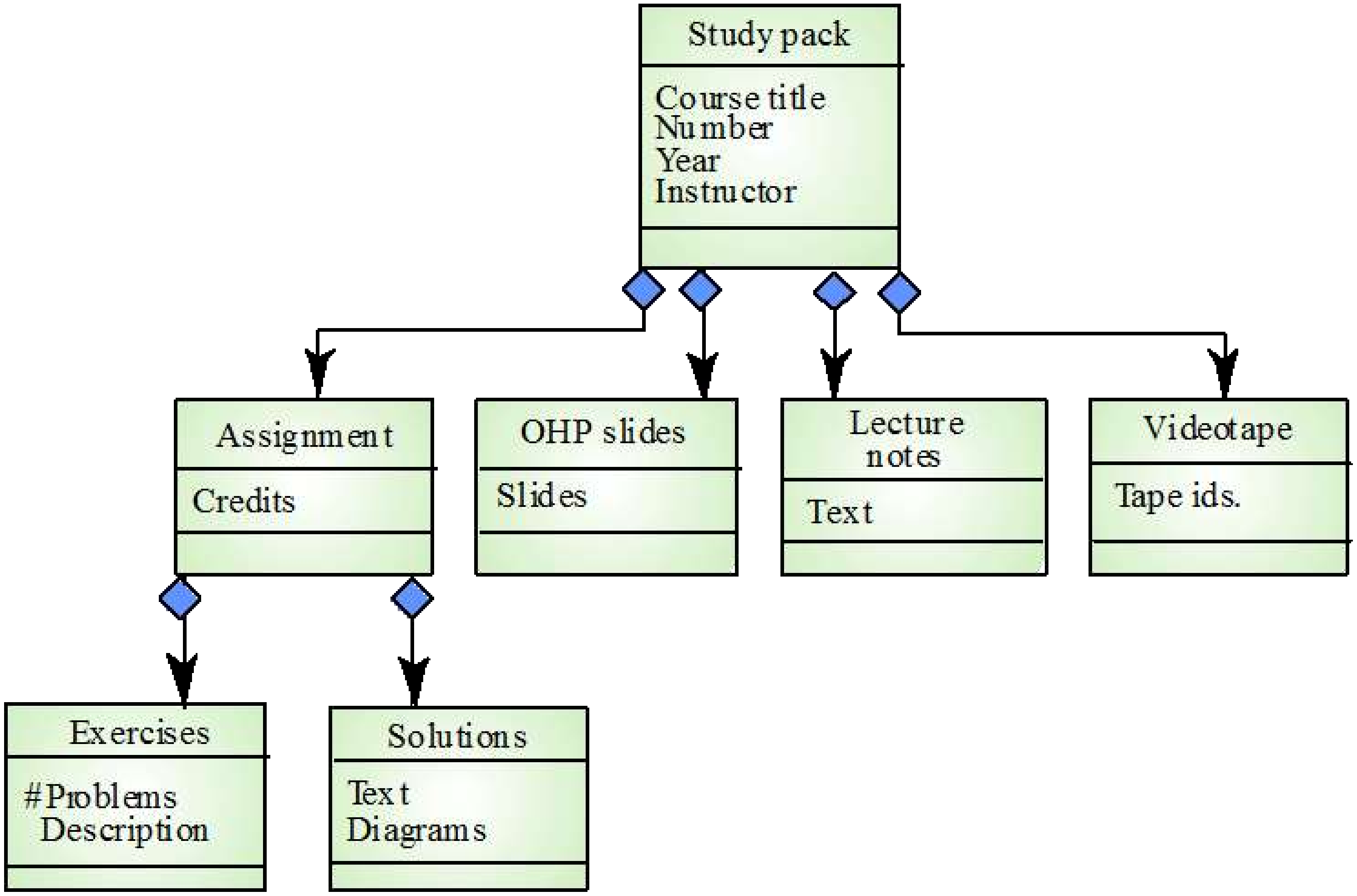
Object models

- Object models describe the system in terms of object classes
- An object class is an abstraction over a set of objects with common attributes and the services (operations) provided by each object
- Various object models may be produced
    - Inheritance models
    - Aggregation models
    - Interaction models

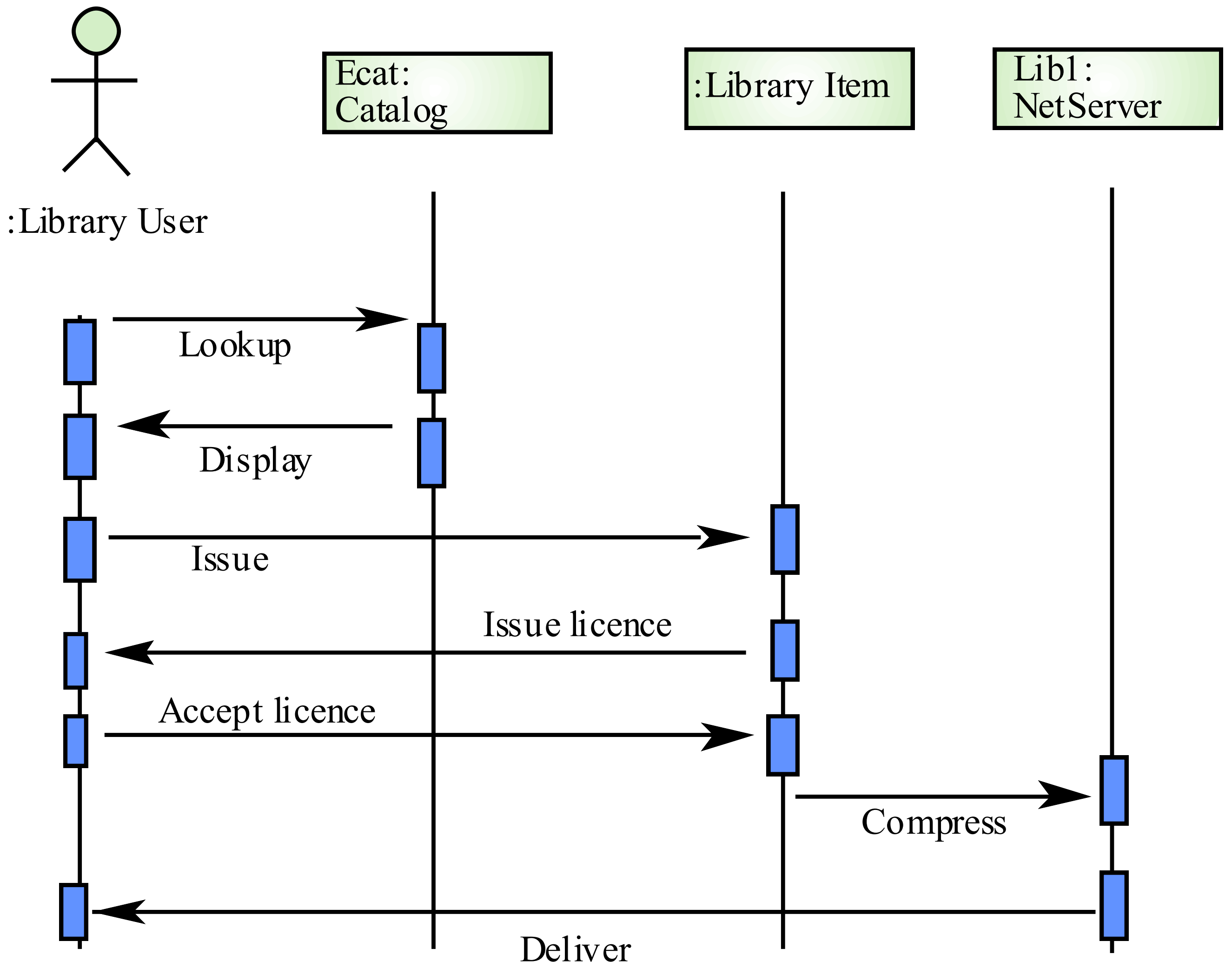
Library class hierarchy



Object aggregation



Object interaction



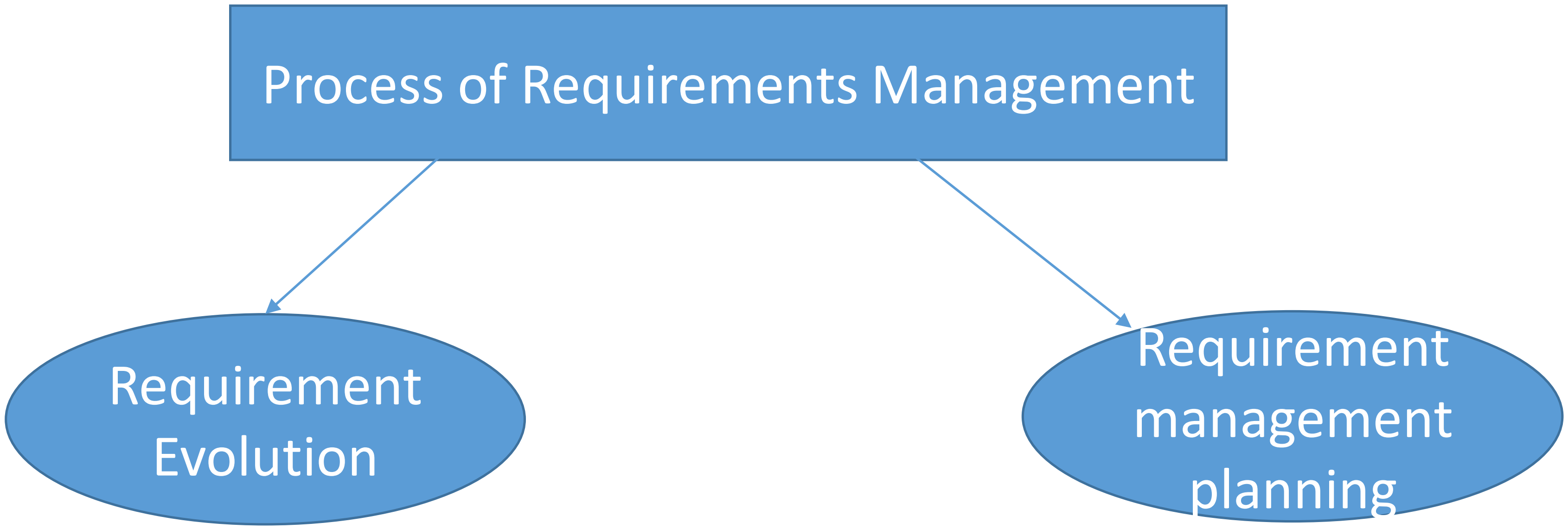
REQUIREMENT VALIDATION

Requirement Validation Techniques are:

- Requirement Reviews
- Prototyping
- Test Case Generation
- Automated Consistency Analysis (Case tools)

REQUIREMENT MANAGEMENT

- It’s the process of understanding and controlling changes to system Requirements.


Process of Requirements Management
Requirement Evolution
Requirement management planning

Requirement Evolution

$$\begin{tikzcd}
\text{Initial understanding of the problem} \arrow[d] \\
\text{Initial Requirement} \arrow[d] \\
\text{Changed understanding of problem} \arrow[d] \\
\text{Changed requirements}
\end{tikzcd}$$