جامعة الملك سعود
King Saud University
College of Computer & Information Sciences

Requirements Engineering

Chapter 4 (Sommerville)

Fall Semester 2021
1st Semester 1443 H

Topics

- Requirements engineering processes
- Requirements elicitation and analysis
- Requirements specification
  - User and System Requirements
  - Functional and non-functional requirements
  - Requirements in Agile development
  - The software requirements document (SRS)
- Requirements validation
- Requirements management

Requirements Engineering Process

Software Specification (Requirements Engineering)	Software Design & Development	Software Validation	Software Evolution
Customers and engineers **define** the software that is to be produced and the **constraints** on its operation.	The software is **designed** and **programmed**.	The software is **checked** to ensure that it is what the customer requires.	The software is **modified** to reflect changing customer and market requirements

خطوة أساسية

The 4 Fundamental SW Process Activities

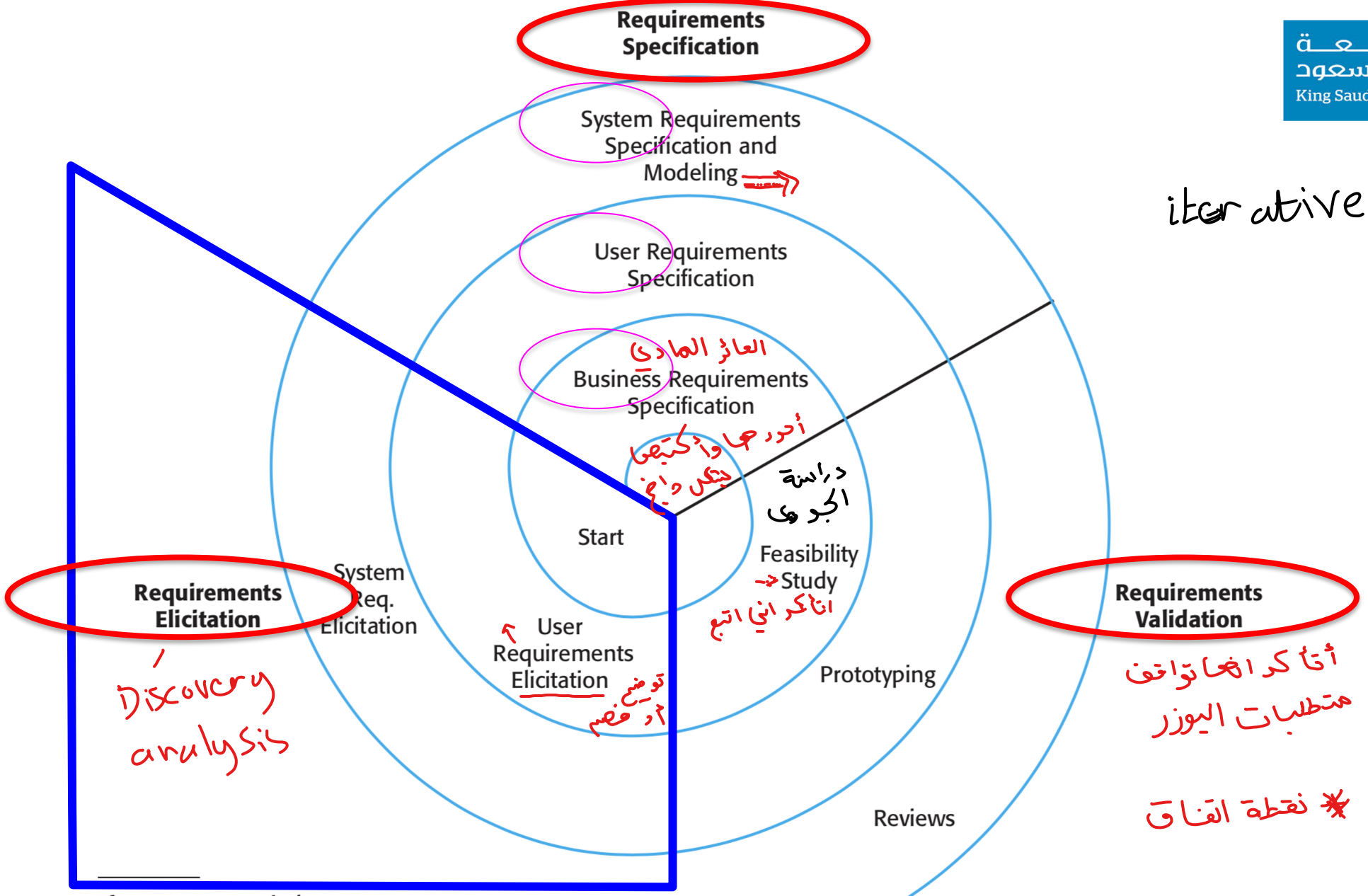


Figure 4.12 A spiral view of the requirements engineering process

iterative process

Discovery analysis

*At the end

Definitions

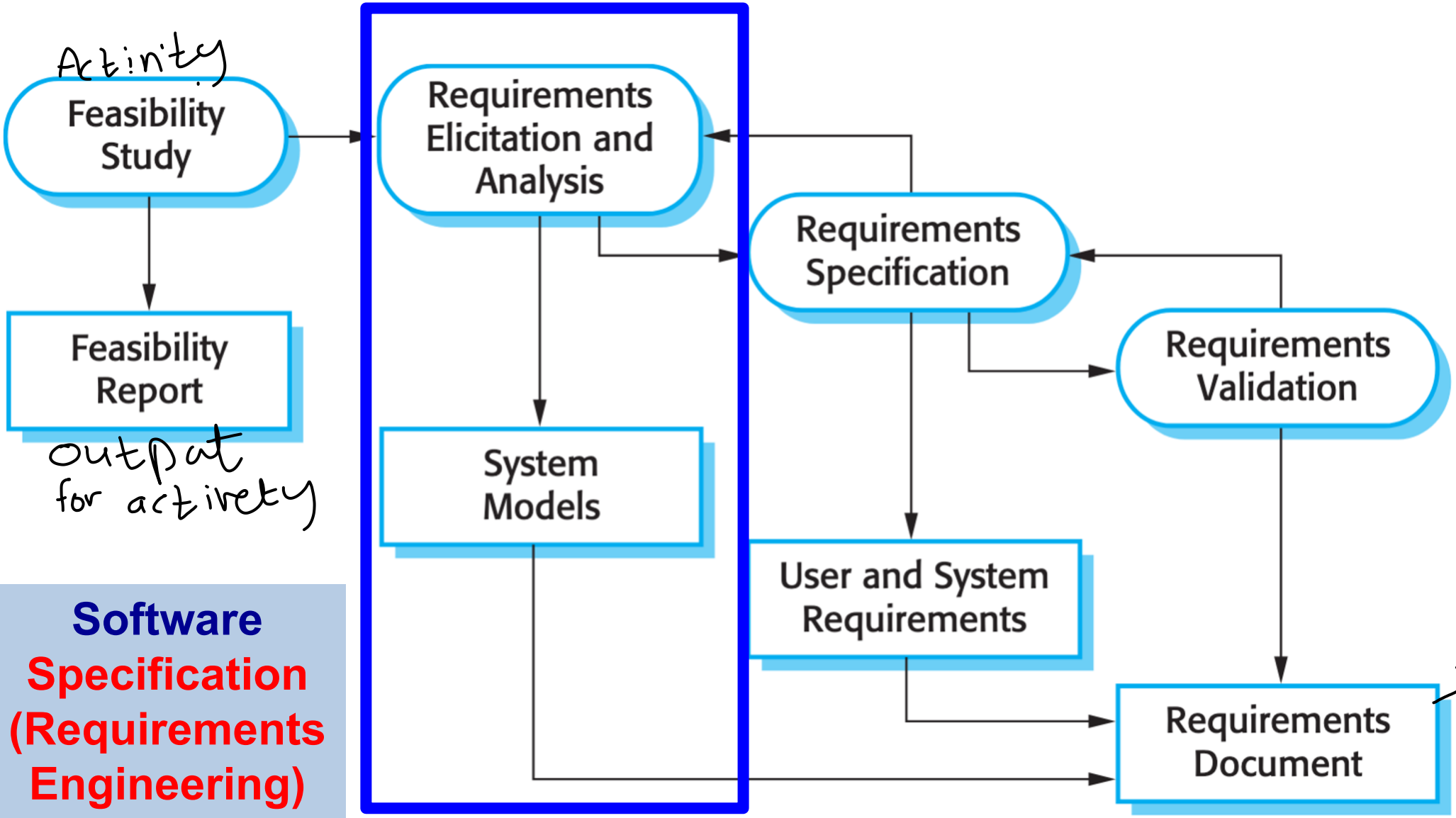
Requirements

The requirements for a system are the descriptions of what the system should do— ***the services*** that it provides and the ***constraints*** on its operation. They reflect the ***needs of customers*** for a system that serves a certain purpose such as controlling a device, placing an order, or finding information.

Requirements Engineering

The process of finding out, analyzing, documenting and checking these services and constraints.

Requirements Elicitation and Analysis



Activity

output for activity

output for the 3 activity

Software Specification (Requirements Engineering)

System Stakeholders ⇒

أي اد يأثر على ديزان السوفتوير يعتبر Stakeholder

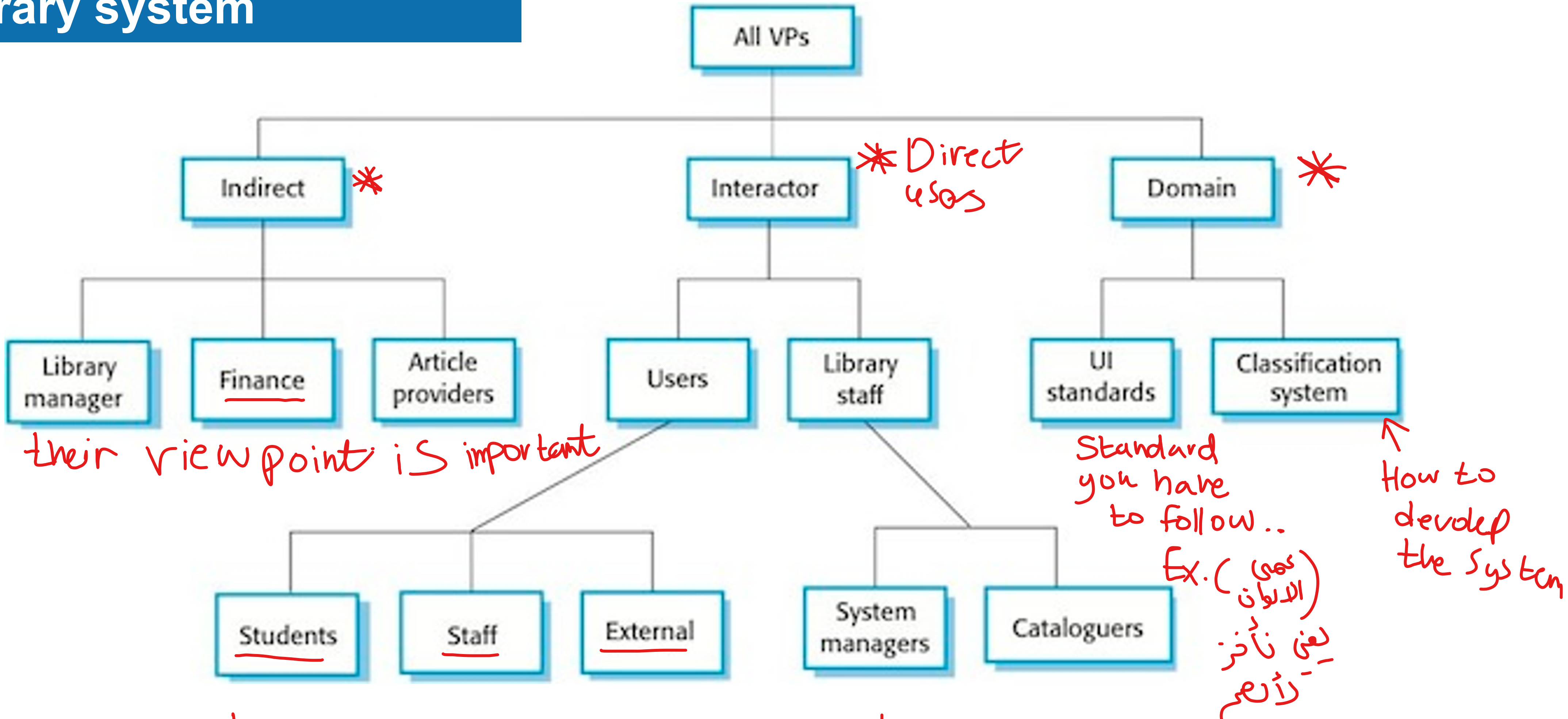
- A system stakeholder is *anyone who should have some direct or indirect influence on the system requirements*.
- Stakeholders include end-users who will interact with the system and anyone else in an organization who will be affected by it.
- Other system stakeholders might be:
  - engineers who are developing or maintaining other related systems,
  - business managers, ⇒
  - domain experts ⇒ المجال

* في منهم Direct unDirect

* أهم Stakeholder ⇐ end users

Example:

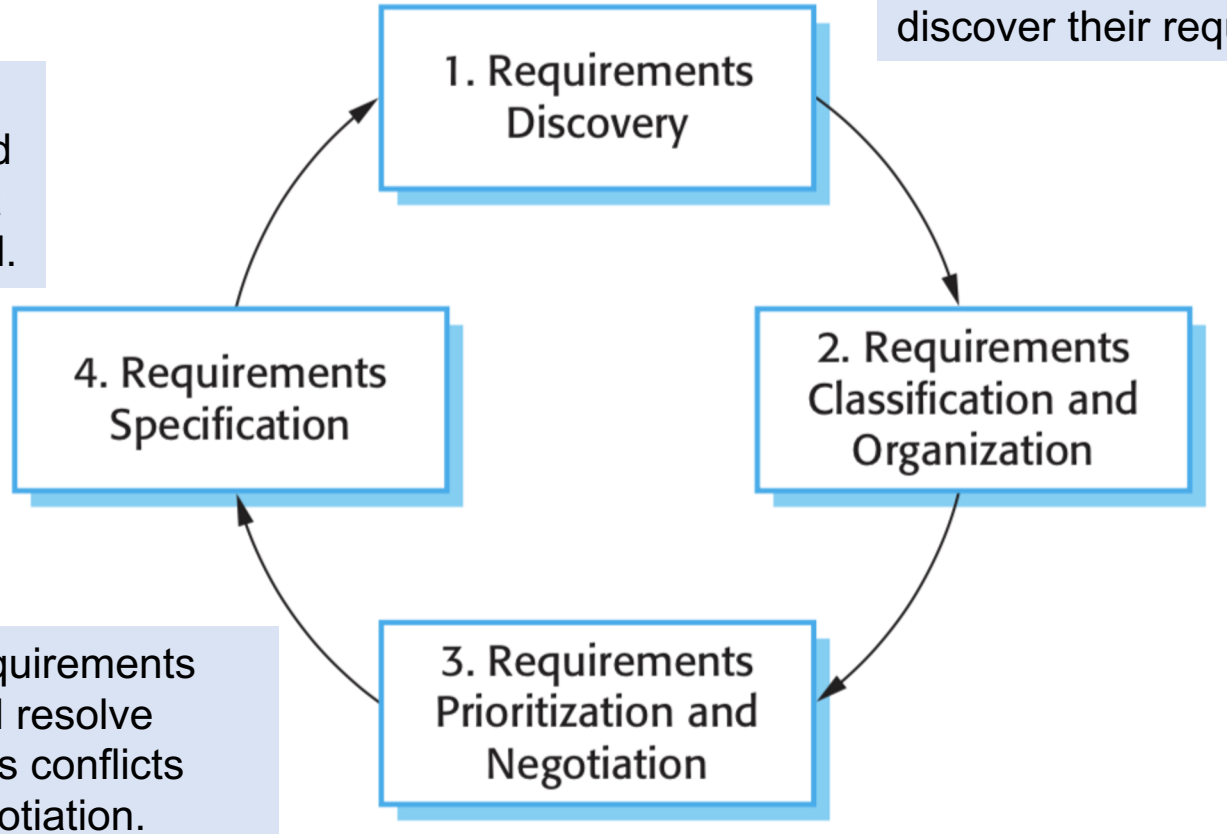
Viewpoint hierarchy for library system



the most important the end users

Requirements Elicitation and Analysis

Interact with stakeholders of the system to discover their requirements. (Elicitation)



Group related requirements, and organize them into coherent clusters. The most common way of grouping requirements is to use a *model of the system architecture* to identify subsystems and to associate requirements with each subsystem.

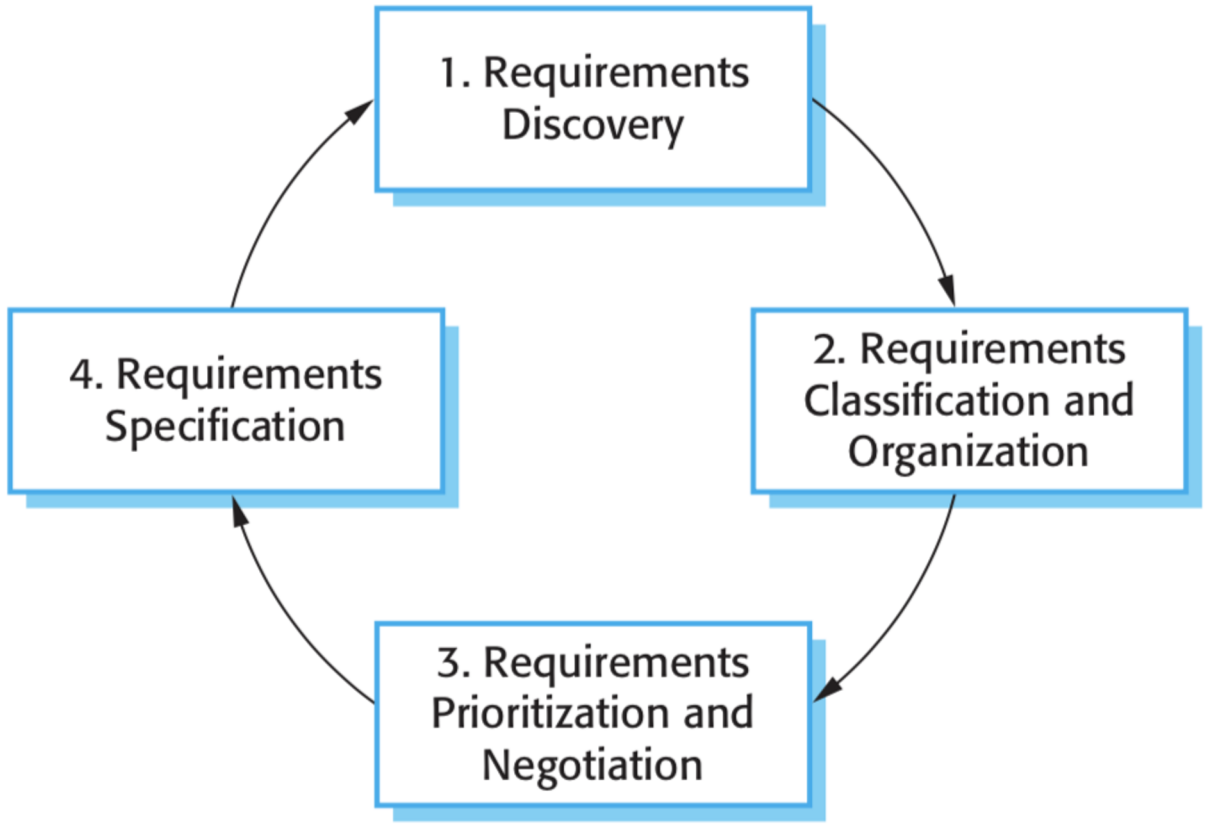
Prioritize requirements and find and resolve requirements conflicts through negotiation.

Document Requirements and input into the next round of the spiral.

Why is it difficult to elicit requirements from stakeholders?

- Stakeholders often don’t know what they want.
- Stakeholders in a system naturally express requirements in their own terms and with implicit knowledge of their own work.
- Different stakeholders have different requirements and they may express these in different ways.
- Different stakeholders have different views on the importance and priority of requirements and, sometimes, these views are conflicting.

Requirements Discovery (Elicitation)



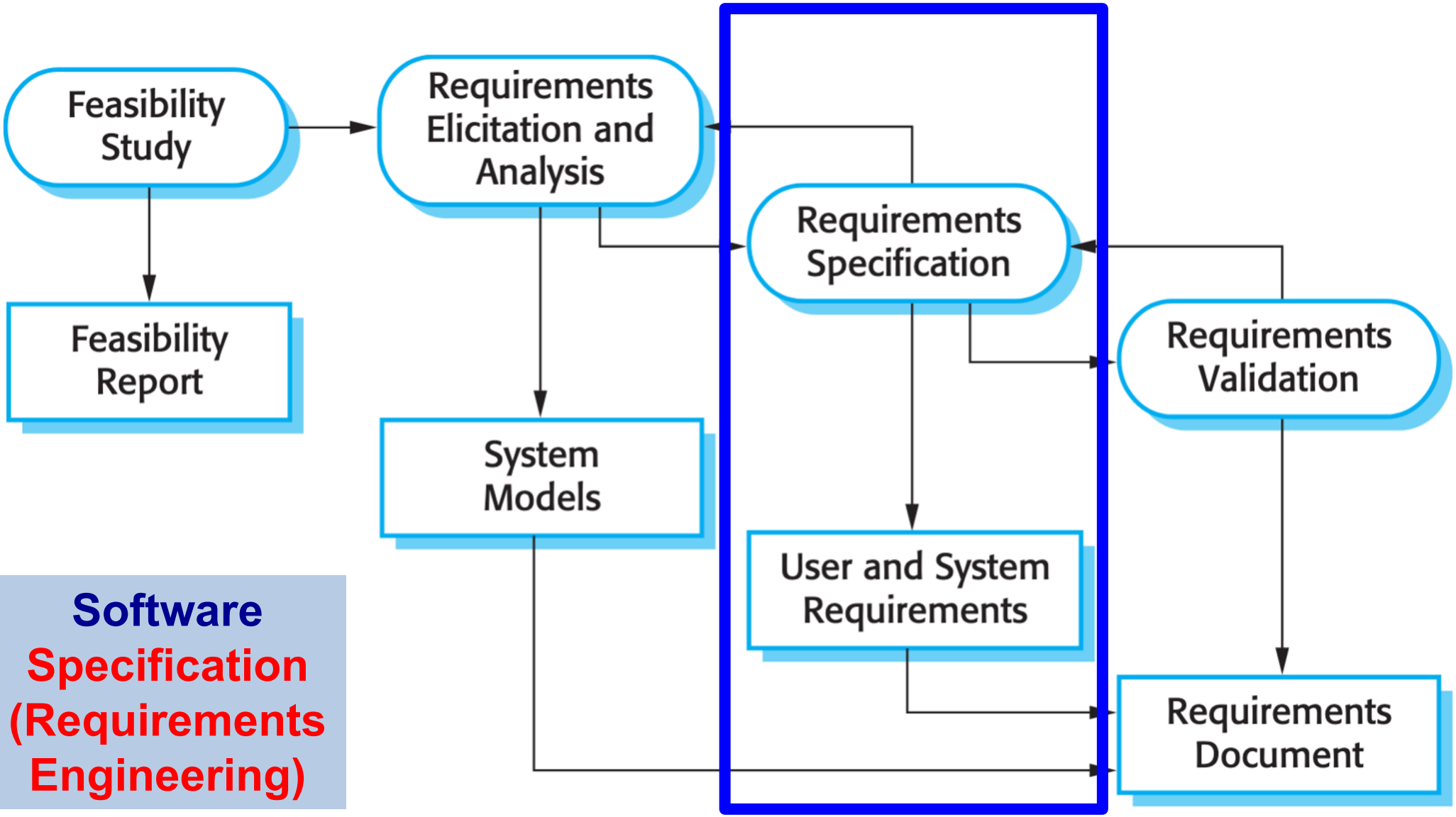
Sources of Information

- Documentation
- System Stakeholders
- Specification of similar systems

Requirements Discovery Methods

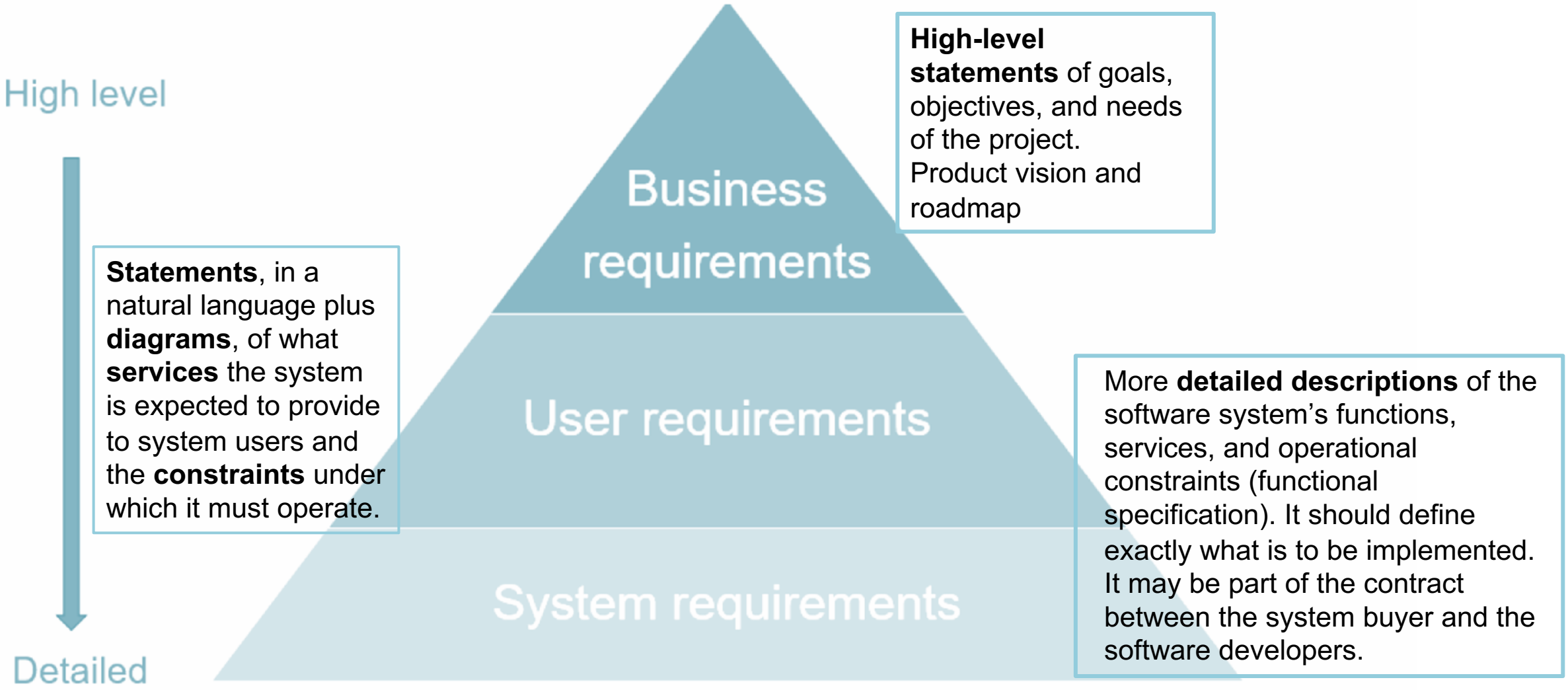
- Interviews
- Questionnaires
- Observation
- Workshops
- Focus groups
- Scenarios (use cases)
- Prototypes to help stakeholders understand what the system will be like.

Requirements Specification



Software Specification (Requirements Engineering)

Requirements classification



High-level statements of goals, objectives, and needs of the project.
Product vision and roadmap

Statements, in a natural language plus **diagrams**, of what **services** the system is expected to provide to system users and the **constraints** under which it must operate.

More **detailed descriptions** of the software system's functions, services, and operational constraints (functional specification). It should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers.

https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements/

Functional and Non-Functional Requirements

Functional Requirements (FR)

Statements of:

- Services the system should provide
- How the system should react to particular inputs.
- How the system should behave in particular situations.

Non-Functional Requirements (NFR)

Constraints on the services or functions offered by the system:

- Constraints on the development process
- Quality constraints

Apply to the system as a whole, rather than individual system features or services

Non-Functional Requirements

- Non-functional requirements are requirements that are not directly concerned with the specific services delivered by the system to its users.
- They may relate to emergent system properties such as reliability, response time, and store occupancy.
- They are often more critical than individual functional requirements. Failing to meet a non-functional requirement can mean that the whole system is unusable.

Example: Library System

A library system that provides a single interface to a number of databases of articles in different libraries. Students can search for, download and print these articles for personal study.

- The student shall be able to search either all of the initial set of databases or select a subset from it. (Functional/ User Requirement)
- The system shall provide pdf viewer for the user to read documents in the document store. (Functional/ System Requirement)
- The system shall allocate a unique identifier (ORDER_ID) for each article purchased. (Functional/ System Requirement)
- The system shall allow the student to register by entering their username and password. (Functional/ System Requirement)

Example: Non Functional Requirements (NFR)

- The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.
- *The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (problematic!)*
- *Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use.*

Example of part of the structure for a requirement document (showing just some high-level headings)

1 Prepare Aircraft for Flight
- 1.1 Fueling
- 1.2 Cleaning
- 1.3 Loading baggage
- 1.4 Loading meals & supplies
- 1.5 Loading passengers
- 1.6 Pre-flight Checks
  - 1.6.1 Instrument Failure
  - 1.6.2 De-icing

2 Operate the Flight
- 2.1 Taxiing to runway
- 2.2 Takeoff
- 2.3 Flight
  - 2.3.1 Control
  - 2.3.2 Navigation
  - 2.3.3 Passenger Service
- 2.4 Landing
- 2.5 Docking

3 Maintain Aircraft
- 3.1 Daily Servicing
- 3.2 Major Servicing

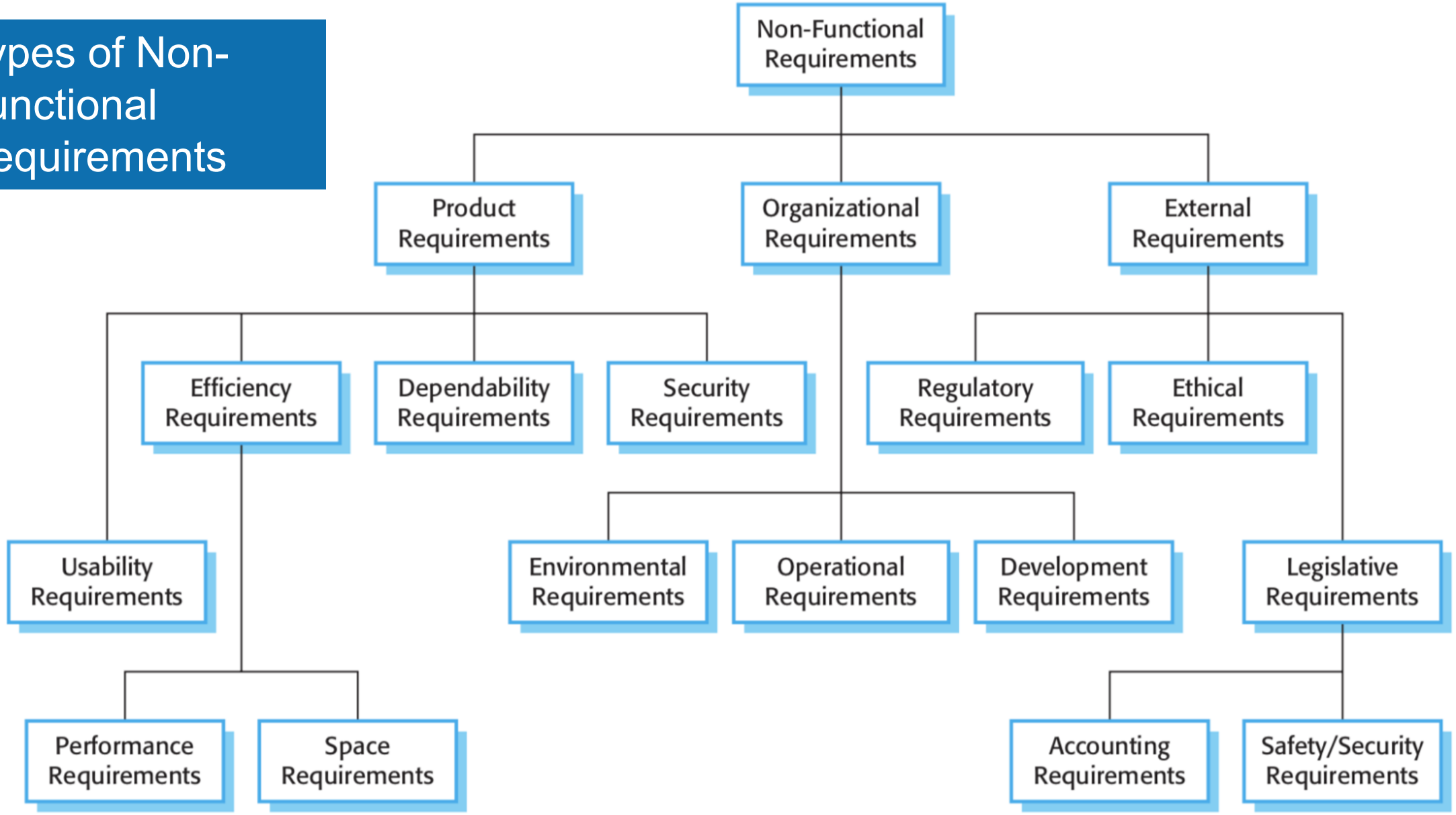
Requirements headings form a natural sequence of activities

Significant exceptions are listed where they would be detected

Main headings are broken down into their own lists of activities

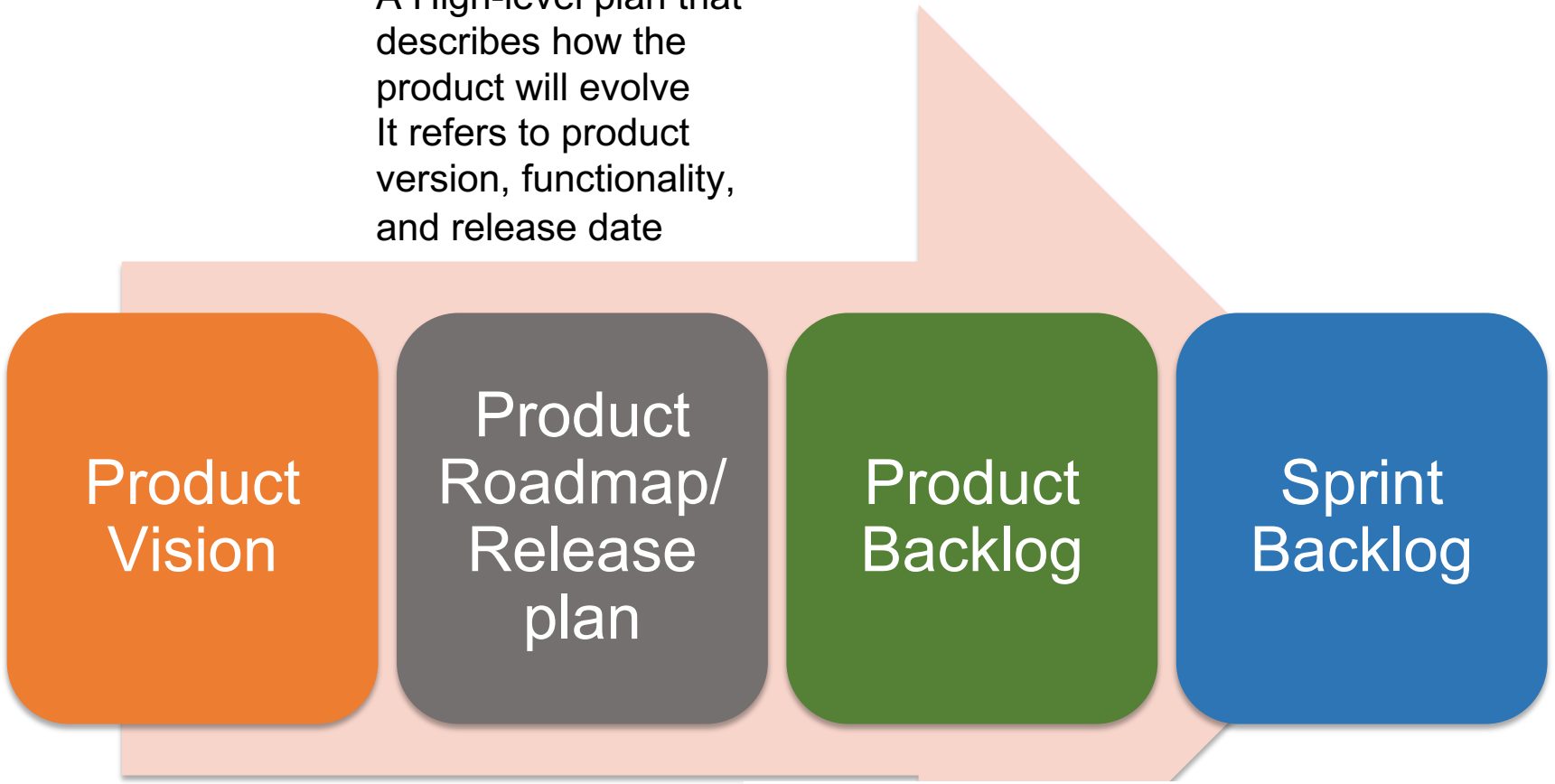
Related activities are grouped logically

Types of Non-Functional Requirements



Requirements in Agile

A High-level plan that describes how the product will evolve
It refers to product version, functionality, and release date



- User Stories selected by The Team
- Will be built in next Sprint
- Fully Estimated
- Divided into Tasks

to make dessision

A combined list of all desired work, including epics, user focused stories, technical work, features & ideas

The Product Vision

The product vision is “a brief statement of the desired future state that would be achieved by developing and deploying a product. A good vision should be simple to state and provide a coherent direction to the people who are asked to realize it.”

Rubin, K. S. (2017). Essential Scrum: A practical guide to the most popular agile process. Upper Saddle River, NJ: Addison-Wesley.

The Product Vision

For _____ (target customer) Users

Who _____ (statement of the need or opportunity) needs

The _____ (product name) **is a** _____ (product category) website / mopile App

That _____ (key benefit, compelling reason to buy) - Feature

Unlike _____ (primary competitive alternative)

Our Product _____ (statement of primary differentiation) - ايش يميز البرودكت

Product Vision for a Shopping Mall

For	people
Who	want an easy way to buy grocery items
The Bee Shopping Mall	is a web-based grocery mall
That	allows consumers to buy items from the web
Unlike	existing solutions that require consume to buy from physical stores
Our product	will provide a totally web-based shopping experience

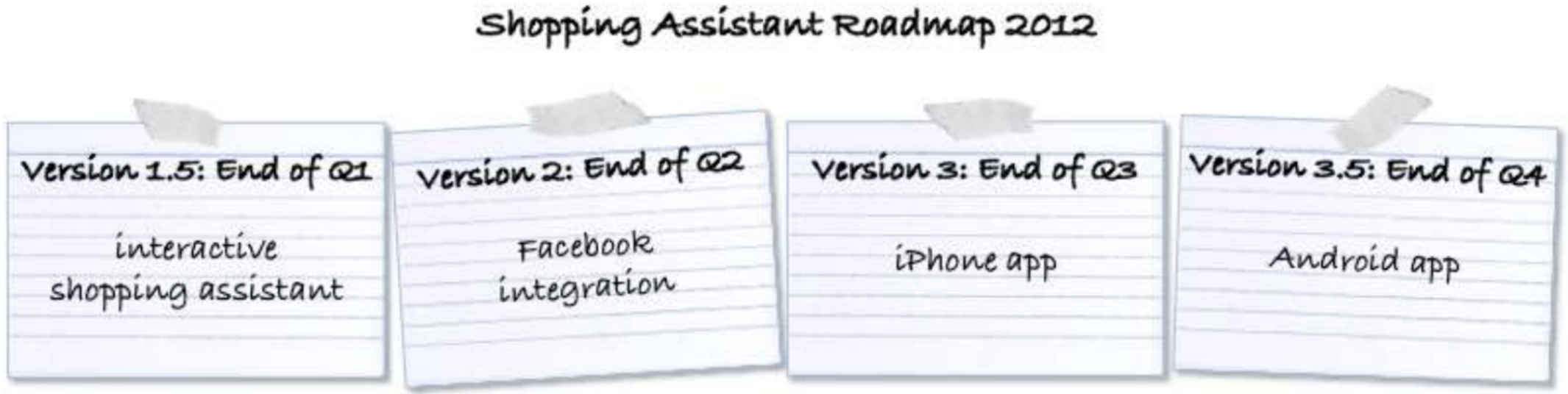
The Product Roadmap

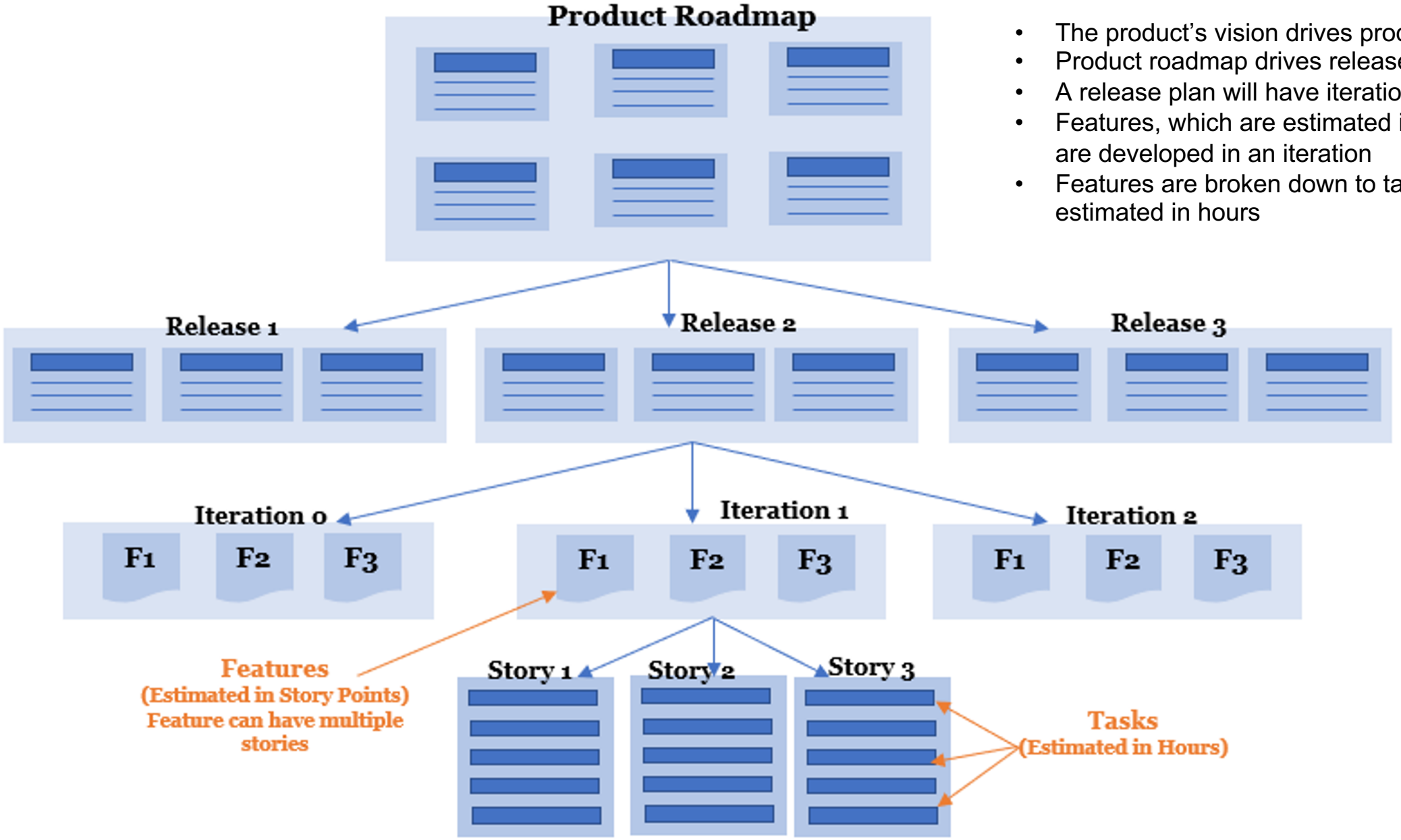
A product roadmap is "a description of the incremental nature of how a product will be built and delivered over time."

1 يصف
2 احد

Rubin, K. S. (2017). Essential Scrum: A practical guide to the most popular agile process. Upper Saddle River, NJ: Addison-Wesley.

Product Roadmap

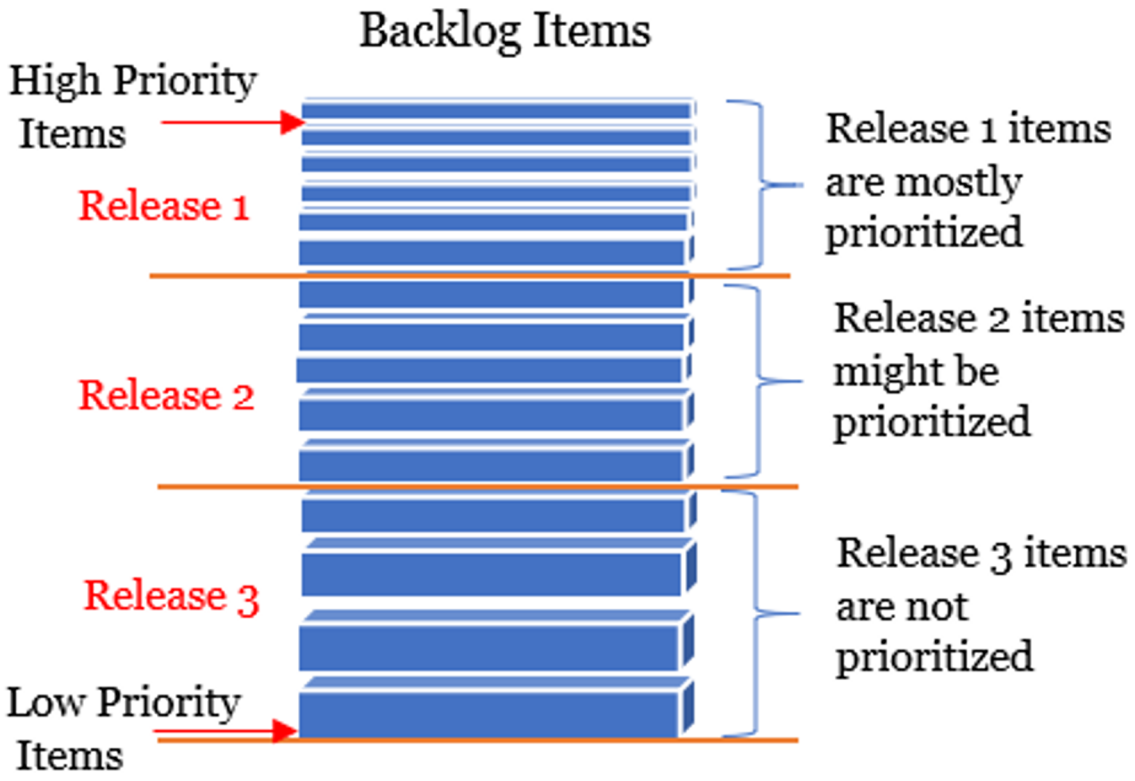




- The product's vision drives product roadmap
- Product roadmap drives release plans
- A release plan will have iterations
- Features, which are estimated in story points, are developed in an iteration
- Features are broken down to tasks, which are estimated in hours

https://www.mpug.com/agile-release-planning-lets-break-it-down/

Product Backlog



User Stories

A **user story** is a short description of a feature told from the perspective of the user of the system. They follow a simple template:

As a < *type of user* >, **I want to** < *some goal* > **so that** < *some reason* >

- Who are we building it for, who the user is? — ***As a <type of user>***
- What are we building, what is the intention? — ***I want <some goal>***
- Why are we building it, what value it brings for the user? — ***So that <some reason>***

INVEST (User Story Criteria)

- **I**ndependent – they can be developed in any sequence and changes to one User Story don't affect the others.
- **N**egotiable – it's up for the team to decide how to implement them; there is no rigidly fixed workflow.
- **V**aluable – each User Story delivers a detached unit of value to end users.
- **E**stimable – it's quite easy to guess how much time the development of a User Story will take.
- **S**mall – it should go through the whole cycle (**designing, coding, testing**) during one sprint.
- **T**estable – there should be clear acceptance criteria to check whether a User Story is implemented appropriately.

The Software Requirments Document

- Sometimes called **The Software Requirements Specification or SRS**.
- It is an **official statement of what the system developers should implement**.
- It should include both **the user requirements** for a system and a detailed specification of **the system requirements**.
- The **level of detail** that you should include in a requirements document depends on **the type of system** that is being developed and the **development process** used.

Organization of the SRS document

- Preface
- Introduction
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System models
- System evolution
- Appendices
- Index

Requirements should be:

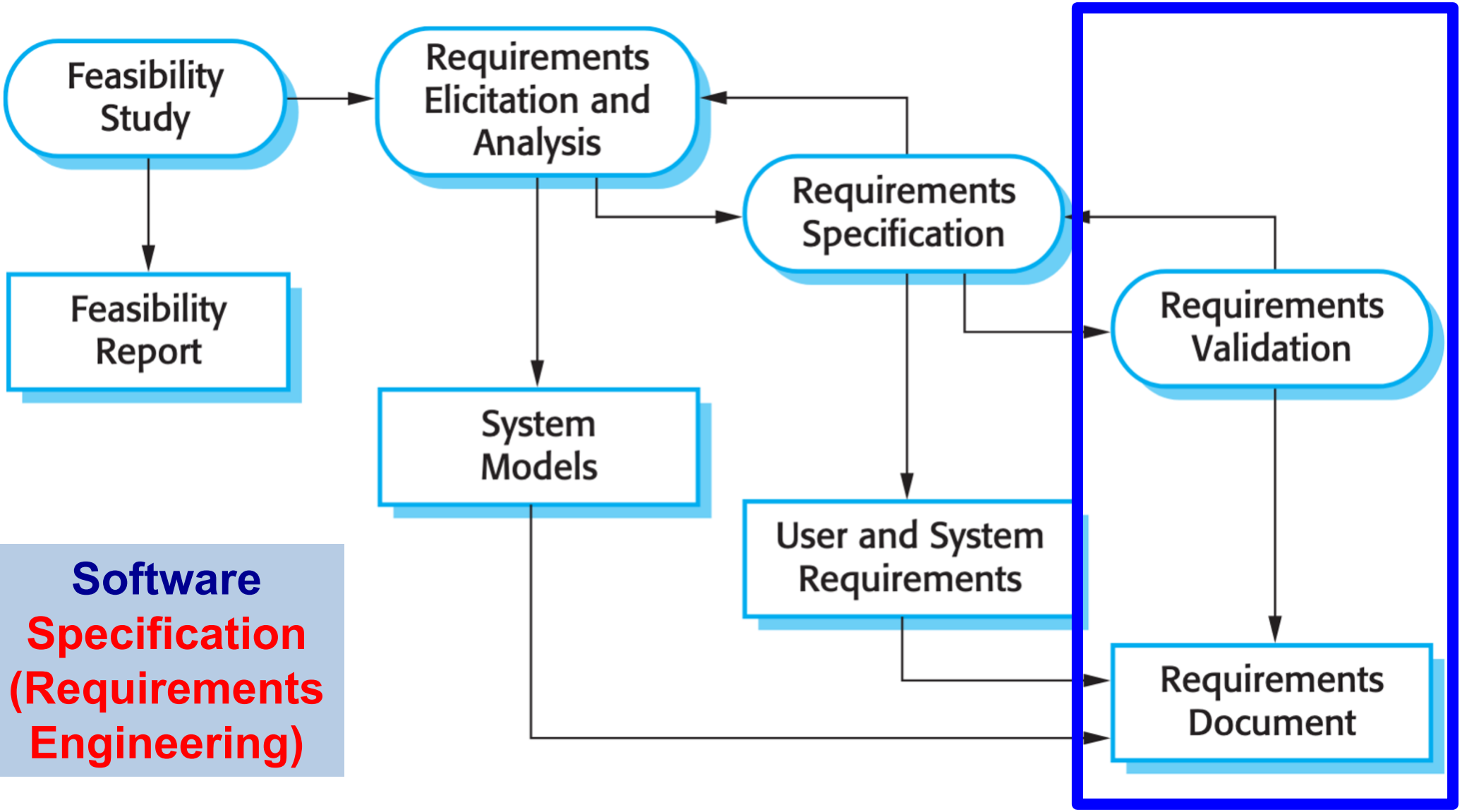
Written in natural language.
Easy to understand.
Written using numbered sentences (1, 1.1,1.2, etc).
Each sentence should express one requirement.

Requirements should include all user and system requirements.
Should cover all the functional and non-functional requirements.

A requirement is ambiguous if a single reader can interpret a requirement in **more than one way**, or when several readers can interpret a requirement **differently from one another**.

Contain no conflicts between any set of requirements.
Examples of conflict include: differences in terminologies used at separate places,
logical conflicts like time period of report generation, etc.

Requirements Validation



Software Specification (Requirements Engineering)

Requirements Validation

- Requirements validation is the process of checking that requirements actually define the system that the customer really wants.
- It is concerned with finding problems with the requirements.
- Requirements validation is important because errors in a requirements document can lead to extensive rework costs when these problems are discovered during development or after the system is in service.

Validation Checks

Requirements in the document should not conflict. That is, there should not be contradictory constraints or different descriptions of the same system function.

The functions proposed by stakeholders should be aligned with what the system needs to perform

Validity checks

Consistency checks

Completeness checks

The requirements document should include requirements that define all functions and the constraints intended by the system user.

Realism checks

Verifiability checks

The requirements should be checked to ensure that they can actually be implemented

You should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement.

Requirements validation techniques

- *Requirements reviews*
  - The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.
- *Prototyping*
  - In this approach to validation, an executable model of the system in question is demonstrated to end-users and customers. They can experiment with this model to see if it meets their real needs.
- *Test-case generation*
  - Requirements should be testable. If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsidered.

Requirements Management

Requirements Management

- Requirements management is the process of understanding and controlling changes to system requirements.
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes.
- You need to establish a formal process for making change proposals and linking these to system requirements.
- You should start planning how to manage changing requirements during the requirements elicitation process.

Thank you!