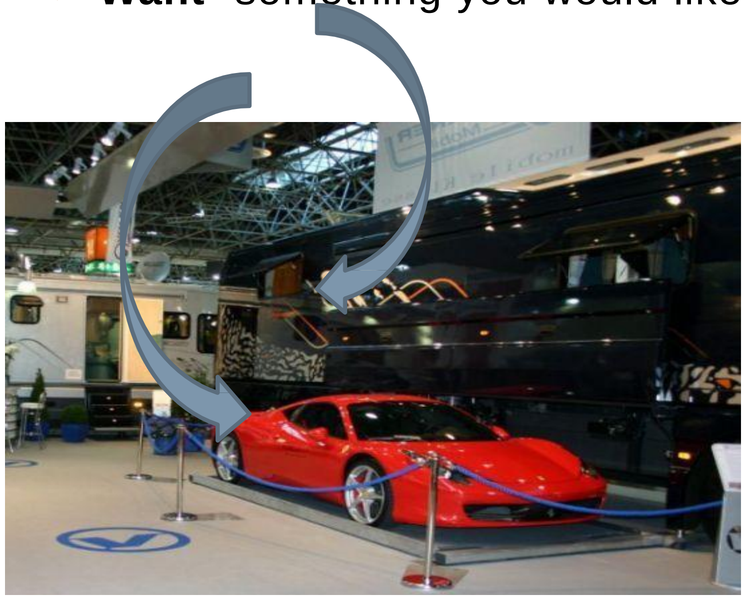
requirements
engineering
process
system
systems
software
lifetime
first
use
most
waterfall
change
term
continues
extensive
publication
probably
documenting
maintains
development
Computer
Alan
report
bibliography
1990s
computer-based
existing
establishment
new
RE
traditional
Society
subject
Software
conference
set
technical
specification
tutorial
model
presented
documented
general
outcome
fast
document
discovering
activities
covers
maintaining
Requirements
IEEE
involved

Requirement

- Something required, something wanted or needed
  - Webster’s dictionary
- There is a huge difference between *wanted* and *needed* and it should be kept in mind all the time
  - ✓ **Need**- something you have to have
  - ✓ **Want** -something you would like to have



Requirement Engineering

- Requirements are ... A specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.

[Sommerville 1997]

Software Requirements

- A complete description of *what* the software system will do without describing *how* it will do it is represented by the software requirements

- Software requirements are ***complete specification of the desired external behavior*** of the software system to be built

Response of software against the input

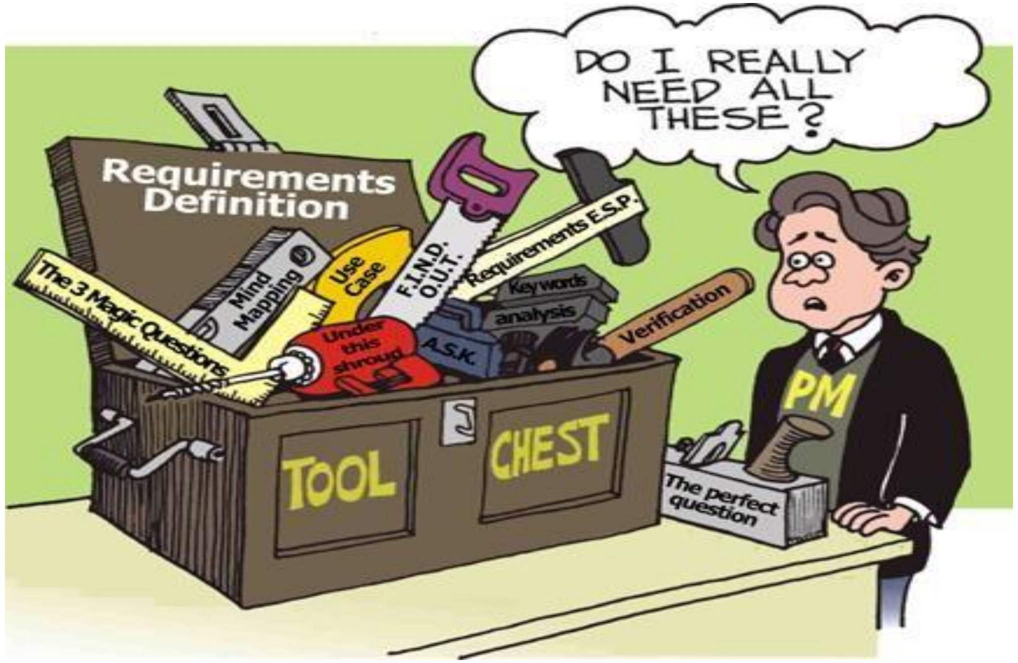
- Software requirements may be:
  - Abstract statements of services
  - Detailed mathematical functions
  - Part of the bid of contract
  - The contract itself
  - Part of the technical document, which describes a product

Requirement [7]

Can be constraint

functionality

- A condition or capability that must be met or possessed by a system...to satisfy a contract, standard, specification, or other formally imposed document
  - IEEE Std 729

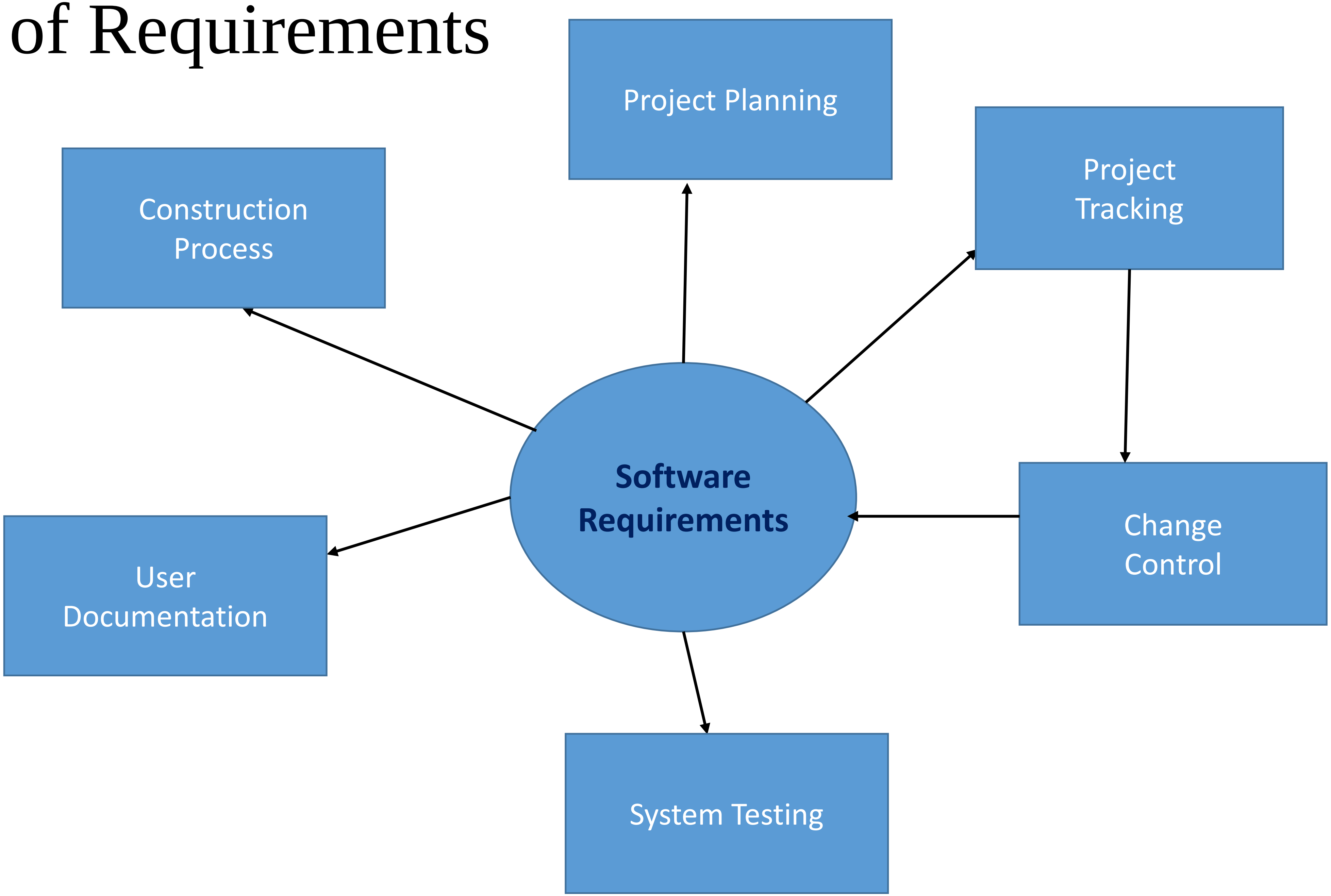


The Goal of Software Development

- The goal of software development is to develop quality software—on time and on budget—that meets customers' real needs.

- A software is good if it ***MEETS STAKEHOLDERS EXPECTATIONS***:
  - it is (at least) correct, reliable, maintainable, user- friendly …
  - the total cost it incurs over all phases of its life cycle is minimal and within the budget

Role of Requirements



Project Planning

Construction Process

Project Tracking

Software Requirements

User Documentation

Change Control

System Testing

Importance of Software Requirements

The hardest single part of building a software system is deciding what to build….. No other part of the work so cripples the resulting system if done wrong. No other part is difficult to rectify later. (Fred Brooks)

The Root Causes of Project Success and Failure

- The first step in resolving any problem is to understand the root causes.

most

- The 1994 Standish Group survey study noted the three commonly cited factors that caused projects to be "challenged":
  - ✓ **Lack of user input: 13 percent of all projects**
  - ✓ **Incomplete requirements and specifications: 12 percent of all projects**
  - ✓ **Changing requirements and specifications: 12 percent of all projects**

”I am so busy!..

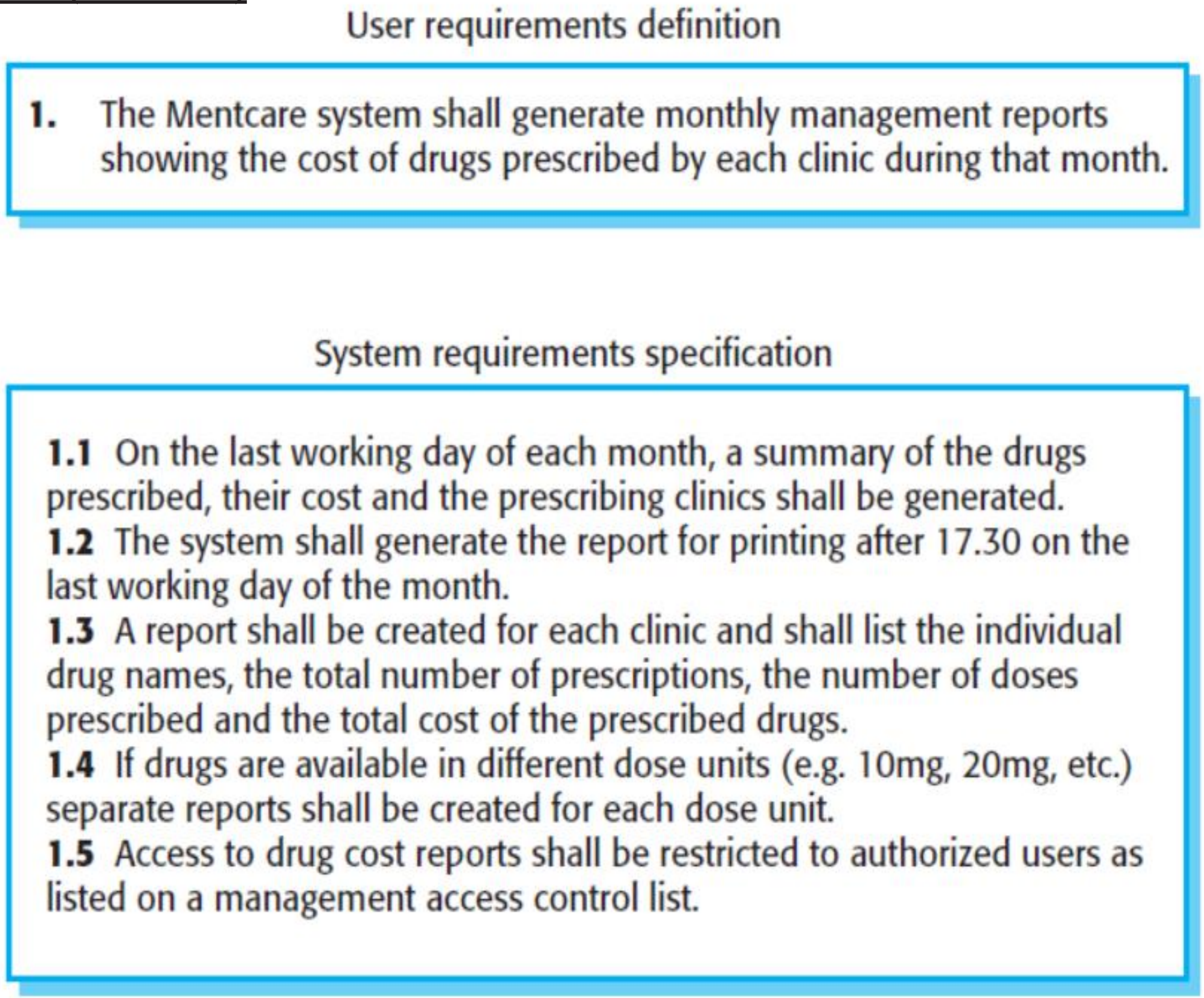
..to waste time with requirements”

User requirements and System requirements

User requirements are statements, in a natural language plus diagrams (conceptual models), of **what services** the system is expected to provide to system users and the **constraints** under which it must operate. The user requirements may vary from broad statements of the system features required to detailed, precise descriptions of the system functionality.

System requirements are **more detailed** descriptions of the software system’s **functions**, **services**, and operational **constraints**. The system requirements document (sometimes called a functional specification) should define exactly what is to be implemented. **It may be part of the contract between the system buyer and the software developers.**

Mental health care patient information system (Mentcare)



User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

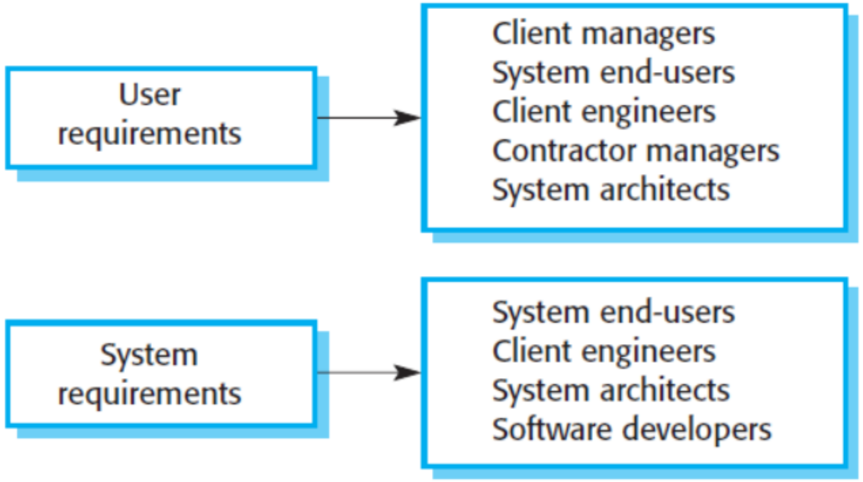
1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

- You need to write requirements at **different levels of detail** because different types of readers use them in different ways.

- The **readers of the user requirements** are not usually concerned with how the system will be implemented and may be **managers** who are not interested in the detailed facilities of the system.

- The **readers of the system requirements** need to know **more precisely** what the system will do because they are **concerned** with **how it will support the business processes** or because they are **involved** in the system **implementation**.

In figure there are examples of system stakeholders. Many stakeholders like end-user, manager,



System stakeholders for the Mentcare system include:

Mental health care patient information system (Mentcare)

1. **Patients** whose information is recorded in the system and relatives of these patients.

2. **Doctors** who are responsible for assessing and treating patients.

3. **Nurses** who coordinate the consultations with doctors and administer some treatments.

4. **Medical receptionists** who manage patients’ appointments.

5. **IT staff** who are responsible for installing and maintaining the system.

6. A **medical ethics manager** who must ensure that the system meets current ethical guidelines for patient care.

7. **Health care managers** who obtain management information from the system.

8. **Medical records staff** who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.

Levels of Requirements

- Business Requirements
- User Requirements
- Functional Requirements
- Non-Functional Requirements

Business Requirements

- These are used to state the high-level business objective of the organization or customer requesting the system or product

- They are used to document main system features and functionalities without going into their nitty-gritty details

- They are captured in a document describing the project vision and scope

User Requirements

- User requirements add further detail to the business requirements

- User requirements are statements of what services the system is expected to provide to system users and the constraints under which it must operate.

Functional Requirements

- These are statements of services the system should provide, **how the system should react to particular inputs**, and **how the system should behave in particular situations**

- Depend on the type of software being developed

- When expressed as user requirements, functional requirements should be written in natural language so that system users and managers can understand them.

- Functional system requirements expand the user requirements and are written for system developers. They should describe the system functions, their inputs and outputs, and exceptions in detail.

- **In some cases, the functional requirements may also explicitly state what the system should not do**

Functional Requirements

- Functional requirements are the statements and services that system should provide in two clearly described external behaviors
  - ✓ Reaction to particular input (e.g give some input to software and it produces a result)
  - ✓ Behavior in particular situations (e.g If I click on an exit button then system behaves in a particular way)

Functional Requirements

- Functional Requirements can be stated from either ***static or dynamic*** perspective
  - ✓ The ***dynamic*** perspective describes the behavior of the system in ***terms of results***
  - ✓ The ***static*** perspective describes the ***functions performed by each entity*** and the way each interacts with other entities and the environment

- Abnormal behavior is also documented as functional requirements in the form of exception handling

- Functional requirements should be complete and consistent

- Customers and developers usually focus all their attention on functional requirements

Mentcare (Example)

1. A user shall be able to search the appointments lists for all clinics.
2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
3. Each staff member using the system shall be uniquely identified by his or her eight-digit employee number.

Non-Functional Requirements

- These are **constraints** on the services or functions offered by the system

- They include **timing constraints**, constraints on the **development process**, and constraints imposed by **standards**

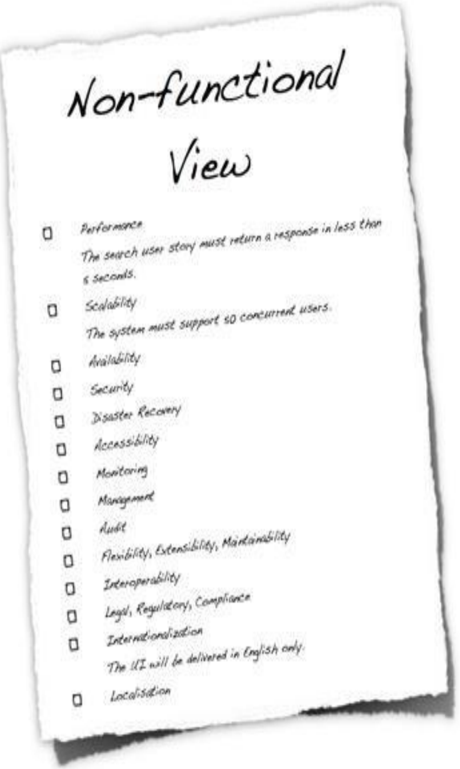
- Non-functional requirements often apply to the system as a whole, rather than individual system features or services

- For example, to ensure that **performance requirements** are met in an **embedded system**, you may have to organize the system to **minimize communications** between components.

Non-Functional Requirements

- Non-functional requirements are often called **qualities** of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements".

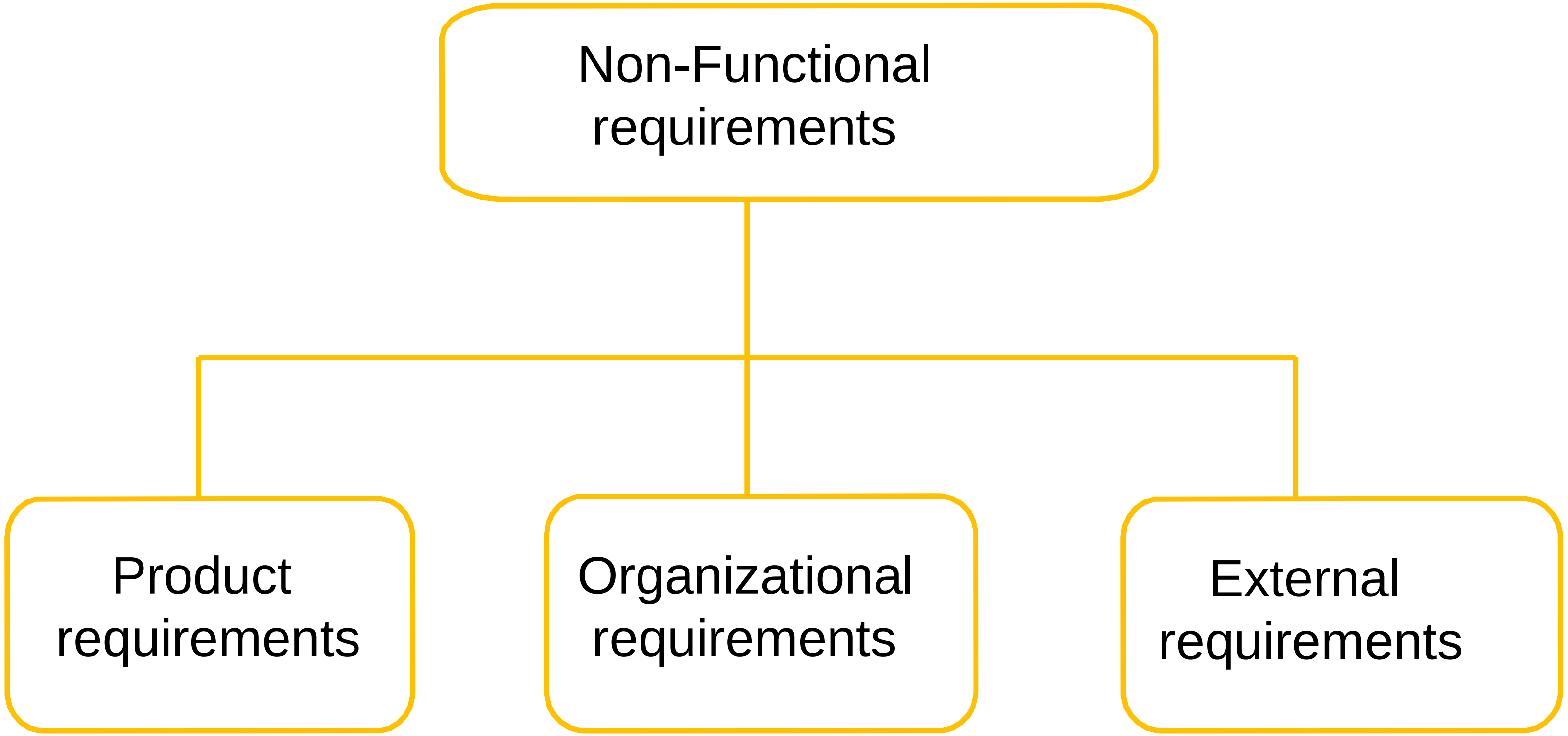
- Informally these are sometimes called the "ilities", from attributes like stability and portability.



Non-functional View

- □ Performance
  The search user story must return a response in less than 5 seconds.
- □ Scalability
  The system must support 50 concurrent users.
- □ Availability
- □ Security
- □ Disaster Recovery
- □ Accessibility
- □ Monitoring
- □ Management
- □ Audit
- □ Flexibility, Extensibility, Maintainability
- □ Interoperability
- □ Legal, Regulatory, Compliance
- □ Internationalization
  The UI will be delivered in English only.
- □ Localisation

Non-Functional Requirements


Non-Functional requirements

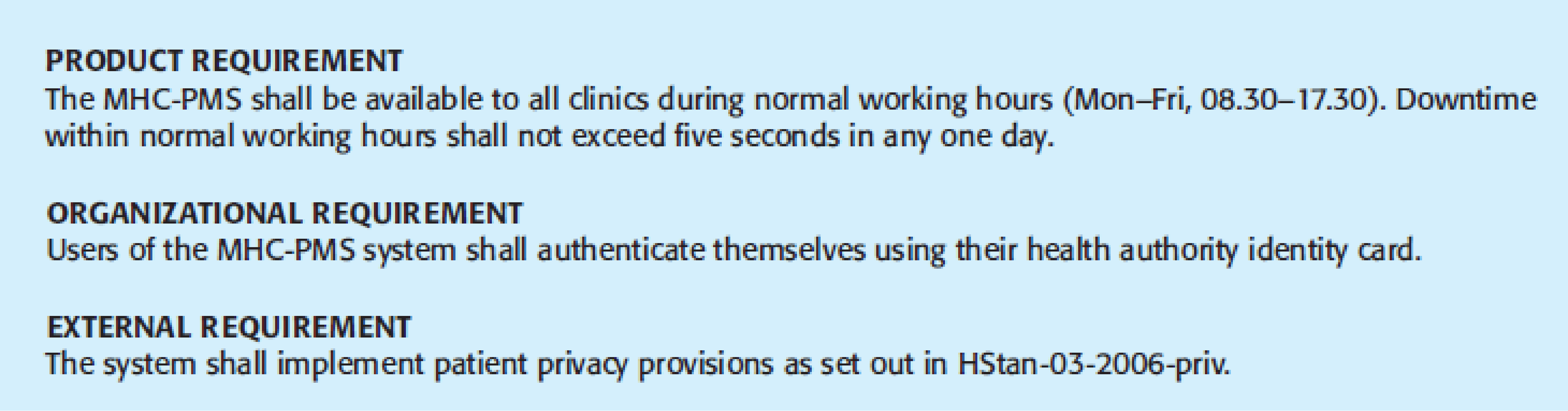
Product requirements

Organizational requirements

External requirements

Non-Functional Requirements

Types	**Explanation**
Product requirements:	specify that the delivered product *must behave* in a particular way *e.g. execution speed, reliability, etc.*
Organizational requirements:	are a consequence of *organizational policies* and procedures *e.g. process standards used, implementation requirements, etc.*
External requirements:	arise from *factors which are external* to the system and its development process *e.g. interoperability requirements, legislative requirements, etc.*

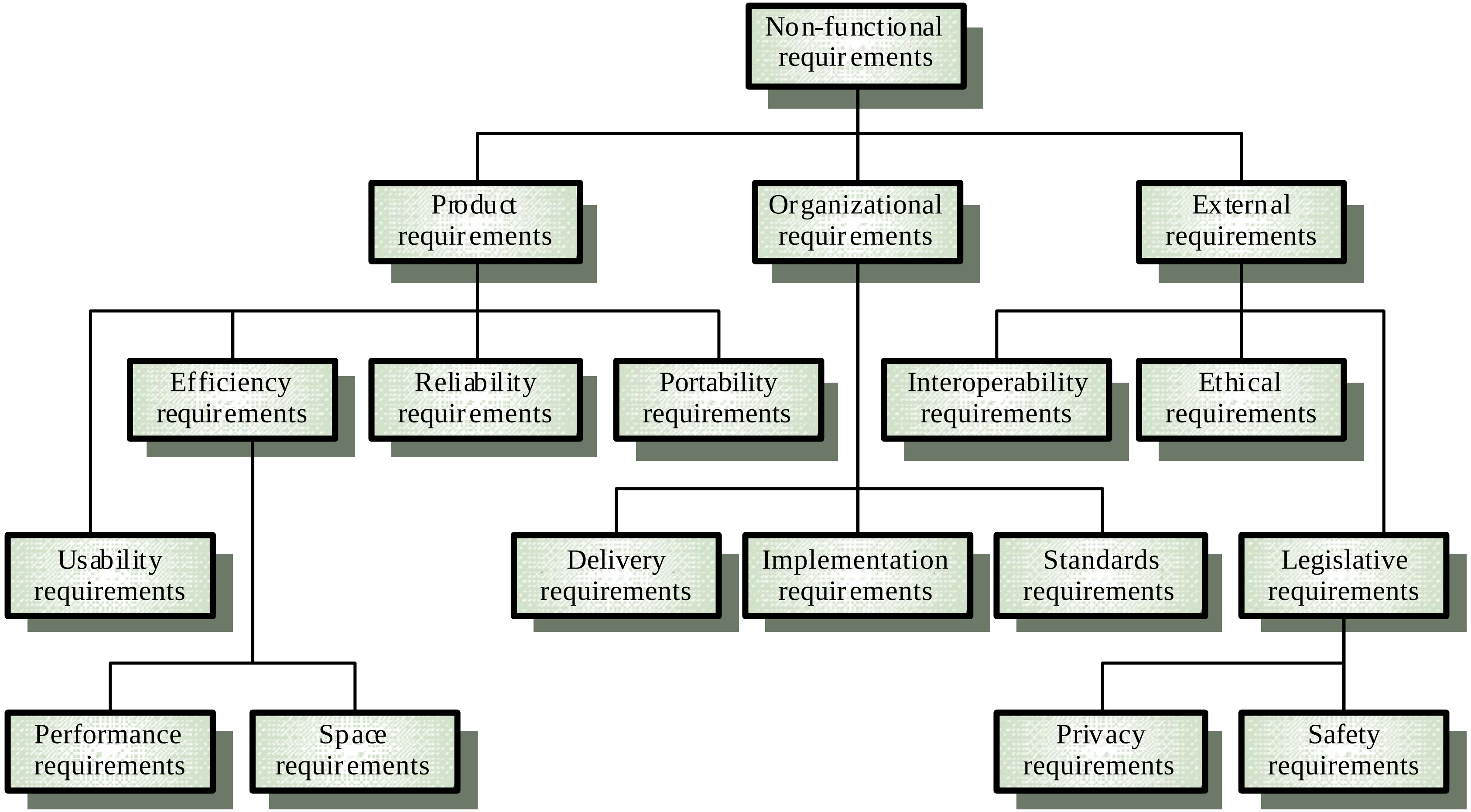


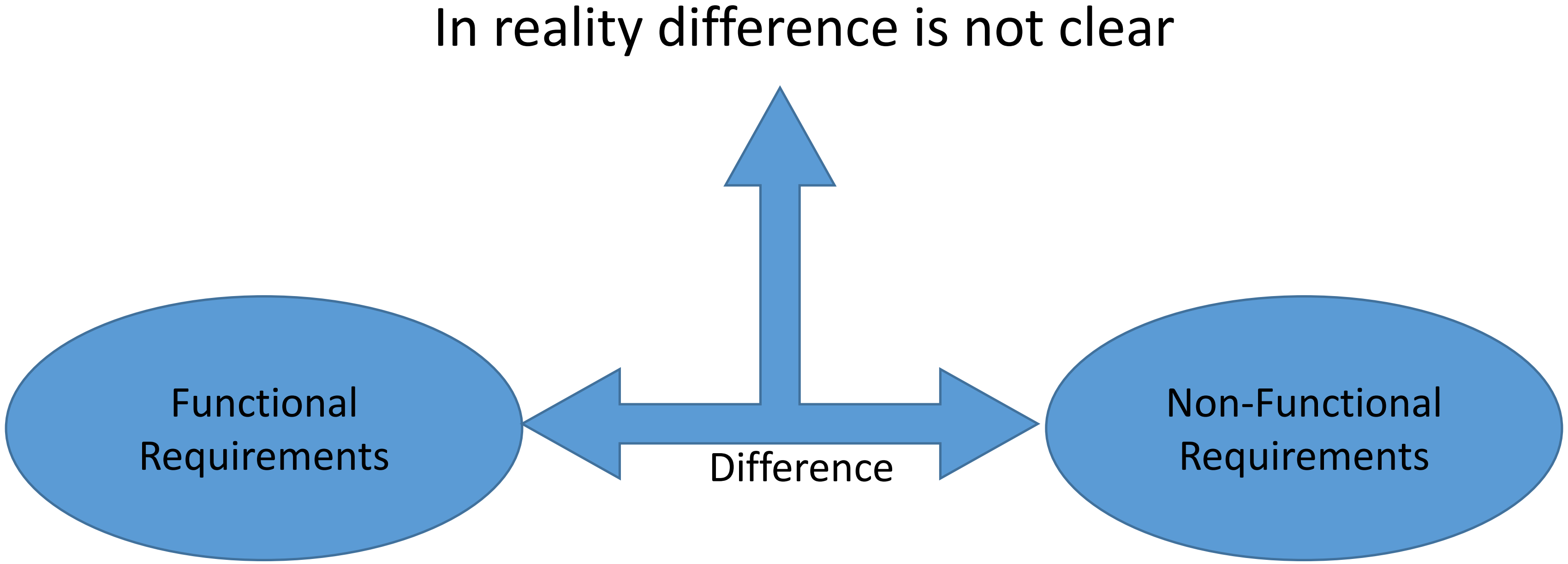
PRODUCT REQUIREMENT
The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

ORGANIZATIONAL REQUIREMENT
Users of the MHC-PMS system shall authenticate themselves using their health authority identity card.

EXTERNAL REQUIREMENT
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Non-Functional Requirements





In reality difference is not clear

Functional Requirements

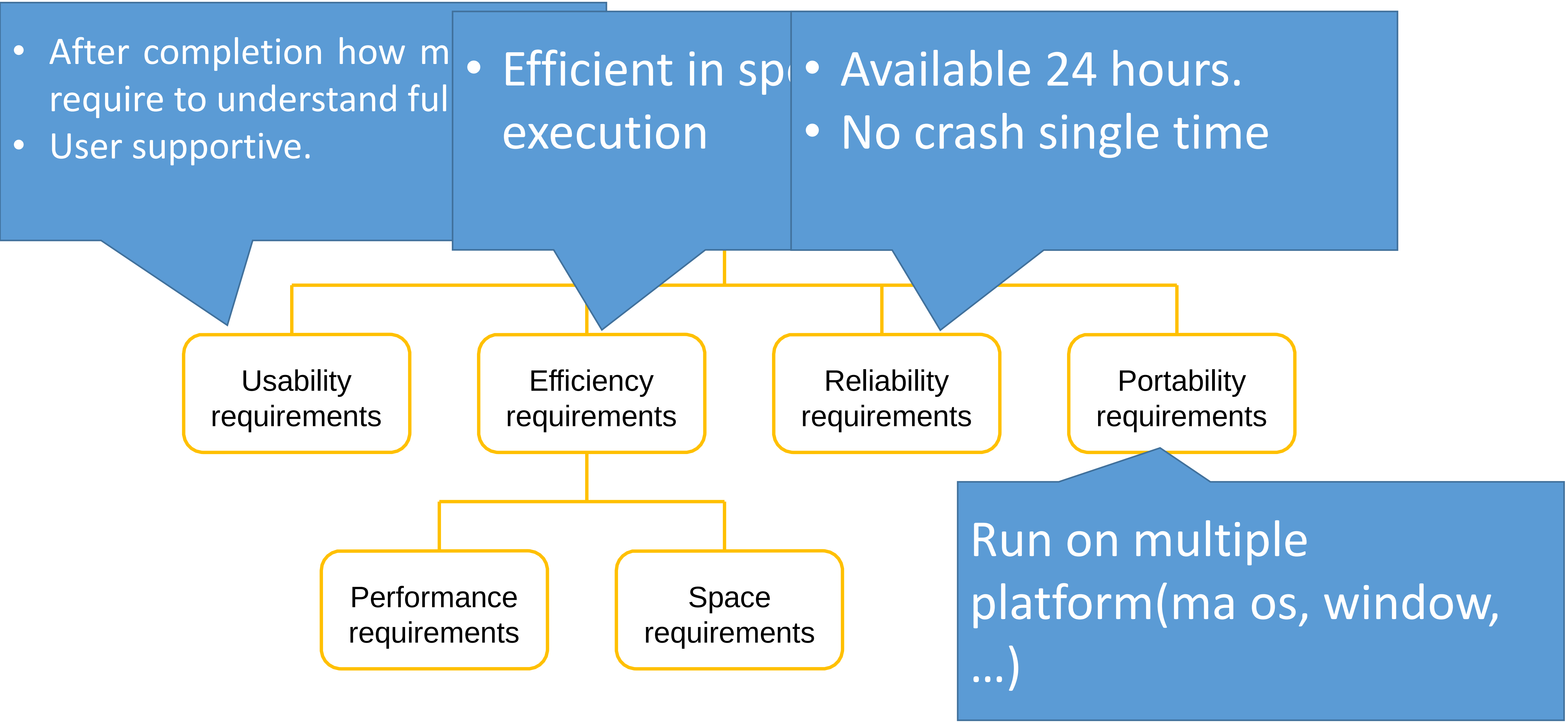
Difference

Non-Functional Requirements

A **user requirement** concerned with **security**, such as a statement **limiting access to authorized users**, may appear to be a nonfunctional requirement.

However, when developed in more detail, this requirement may generate other requirements that are clearly functional, such as the need to include **user authentication facilities** in the system.

Product Requirements



- After completion how m require to understand ful
- User supportive.

- Efficient in sp execution

- Available 24 hours.
- No crash single time

Usability requirements

Efficiency requirements

Reliability requirements

Portability requirements

Performance requirements

Space requirements

Run on multiple platform(ma os, window, …)

Product Requirements

Reliability and performance

- The system shall allow one hundred thousand hits per minute on the website

Mean time to failure

- The system shall not have down time of more than one second for continuous execution of one thousand hours

Usability requirements:

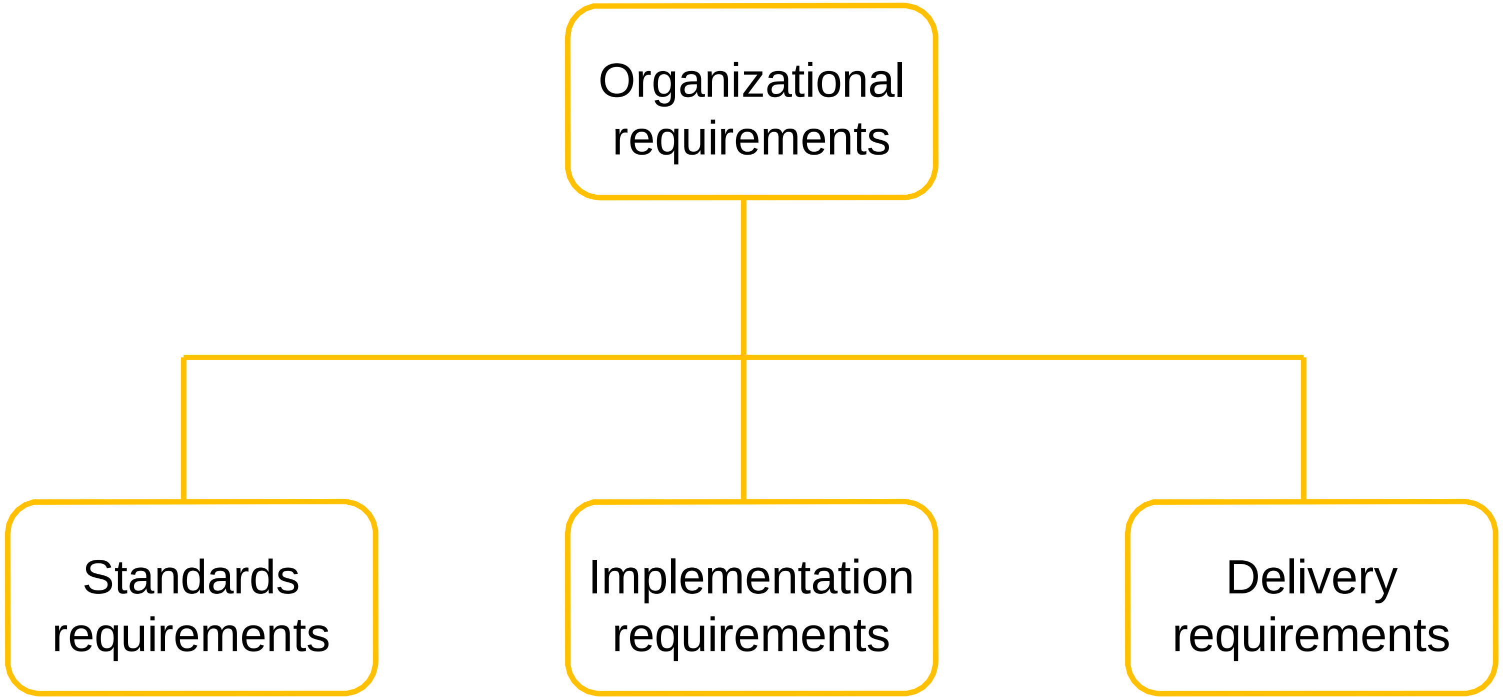
The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.

Usability requirements:
Previously written usability req was not testable.
Here it is rectified as:

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.

See some metrics discussed by Ian Somerville to quantify NFRs.

Organizational Requirements



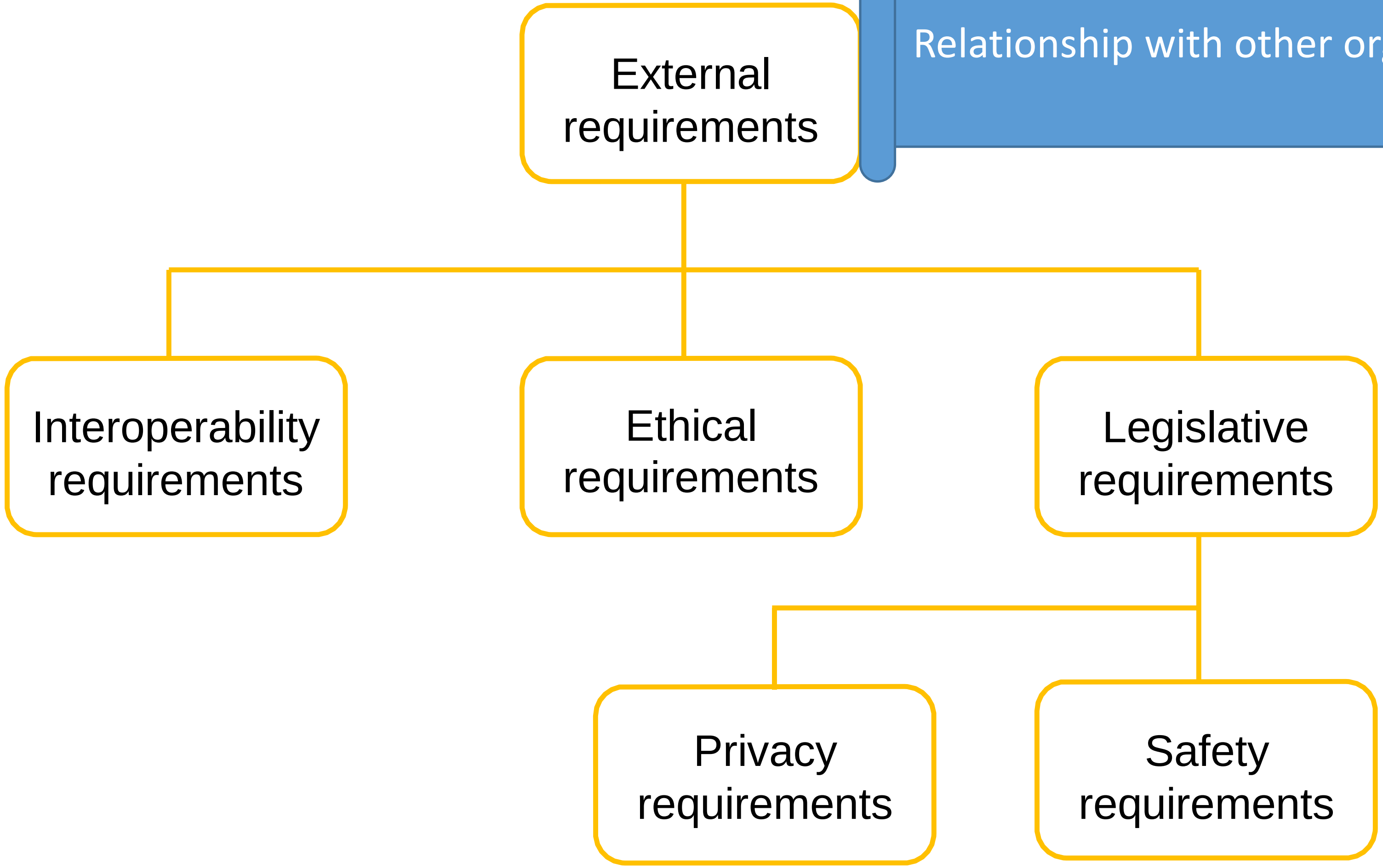
Organizational requirements

Standards requirements

Implementation requirements

Delivery requirements

External Requirements



Relationship with other organizations.

External requirements

Interoperability requirements

Ethical requirements

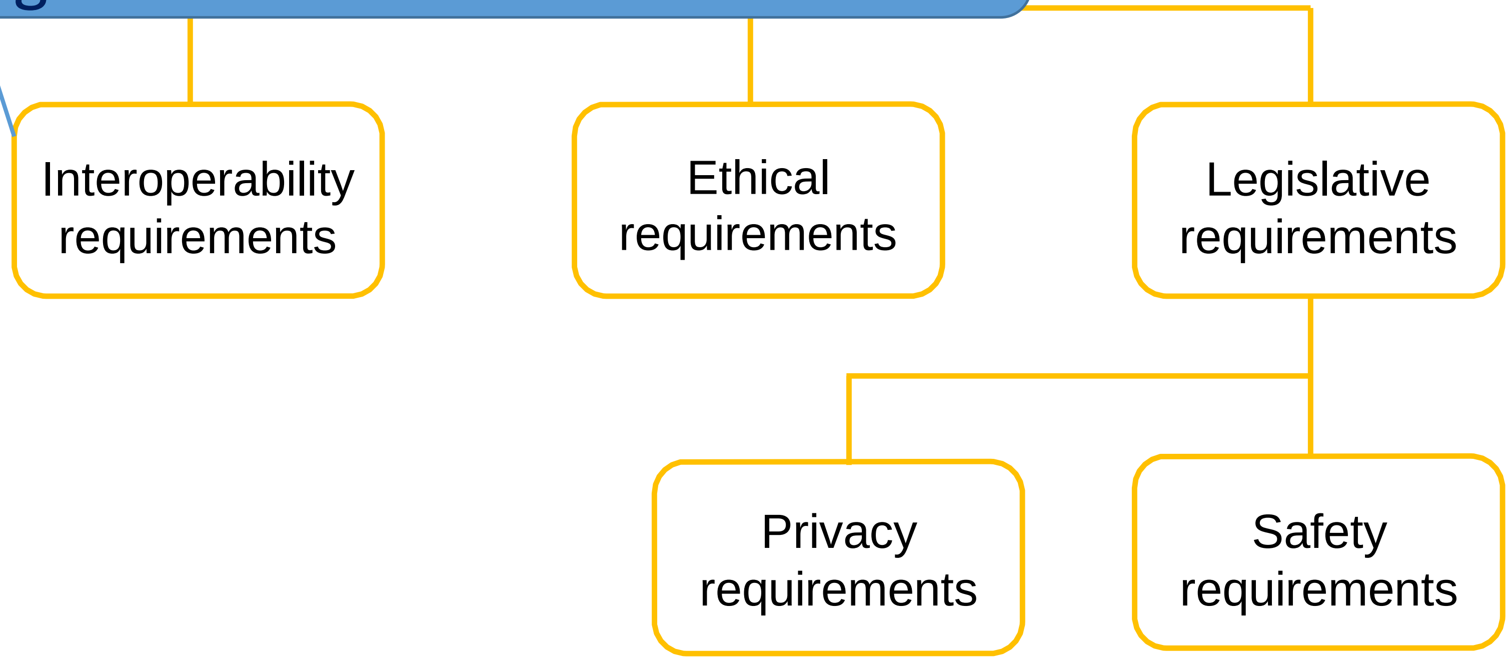
Legislative requirements

Privacy requirements

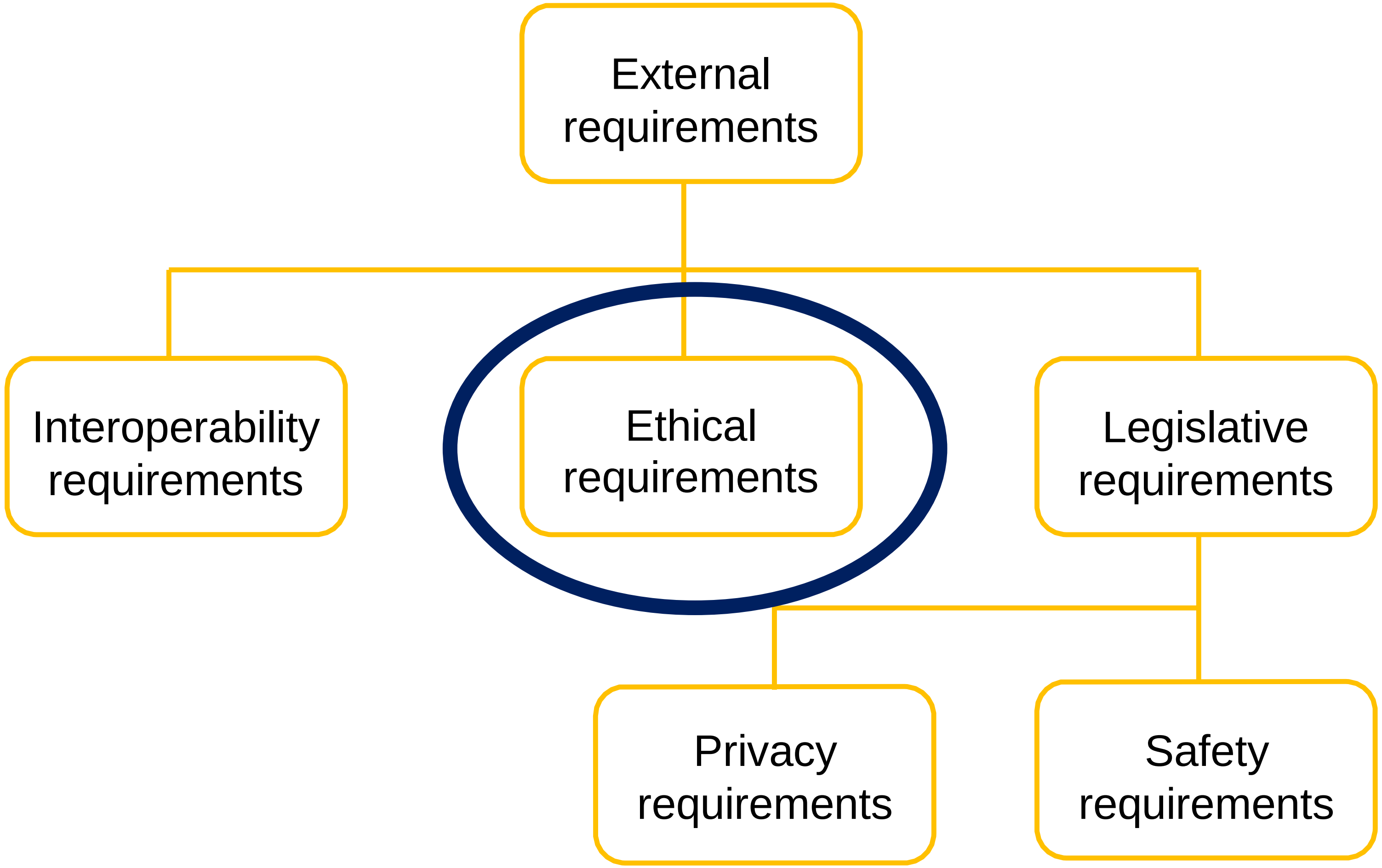
Safety requirements

External Requirements

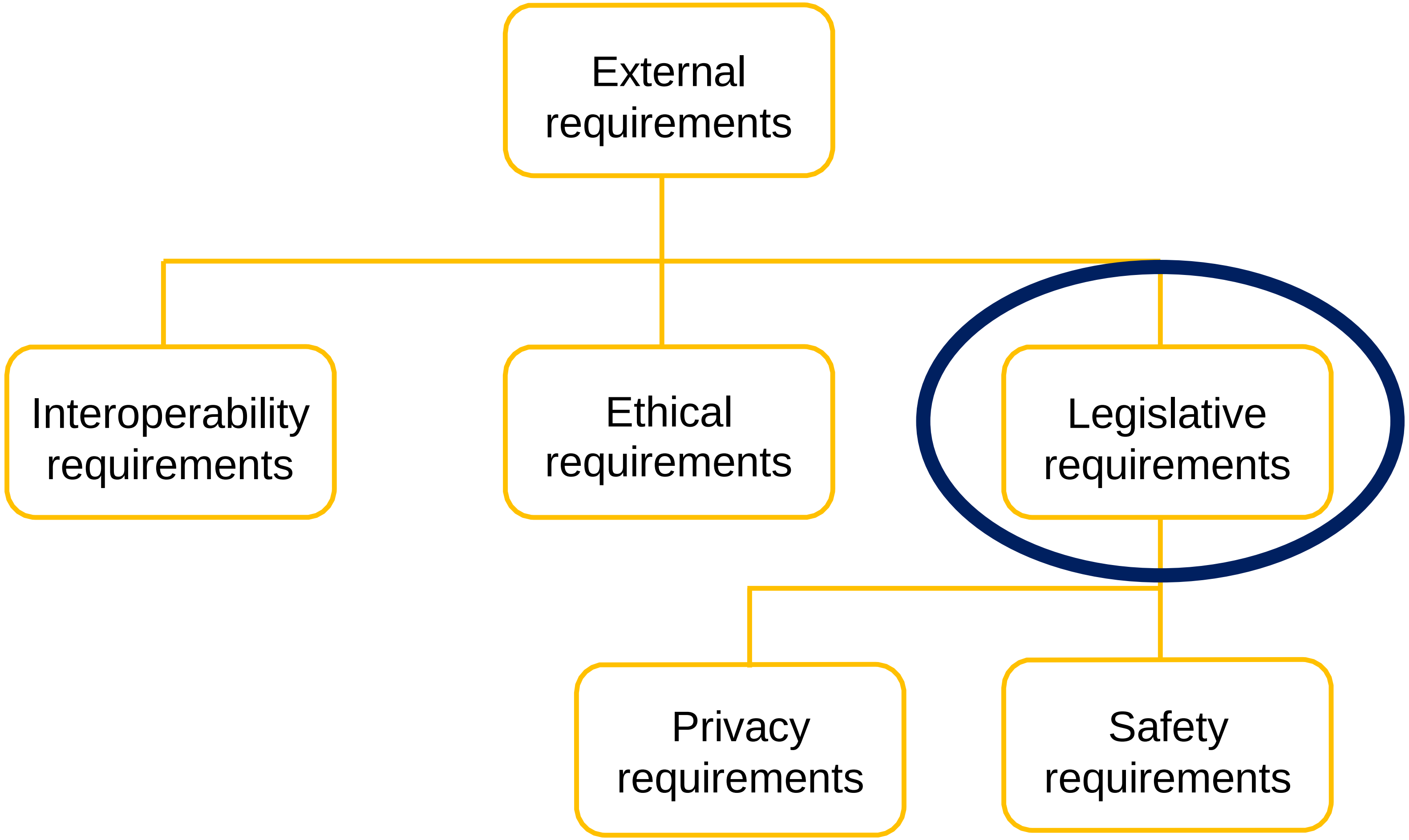
Multiple organization are going to use your same product.
Therefore software companies does not give code and documentation



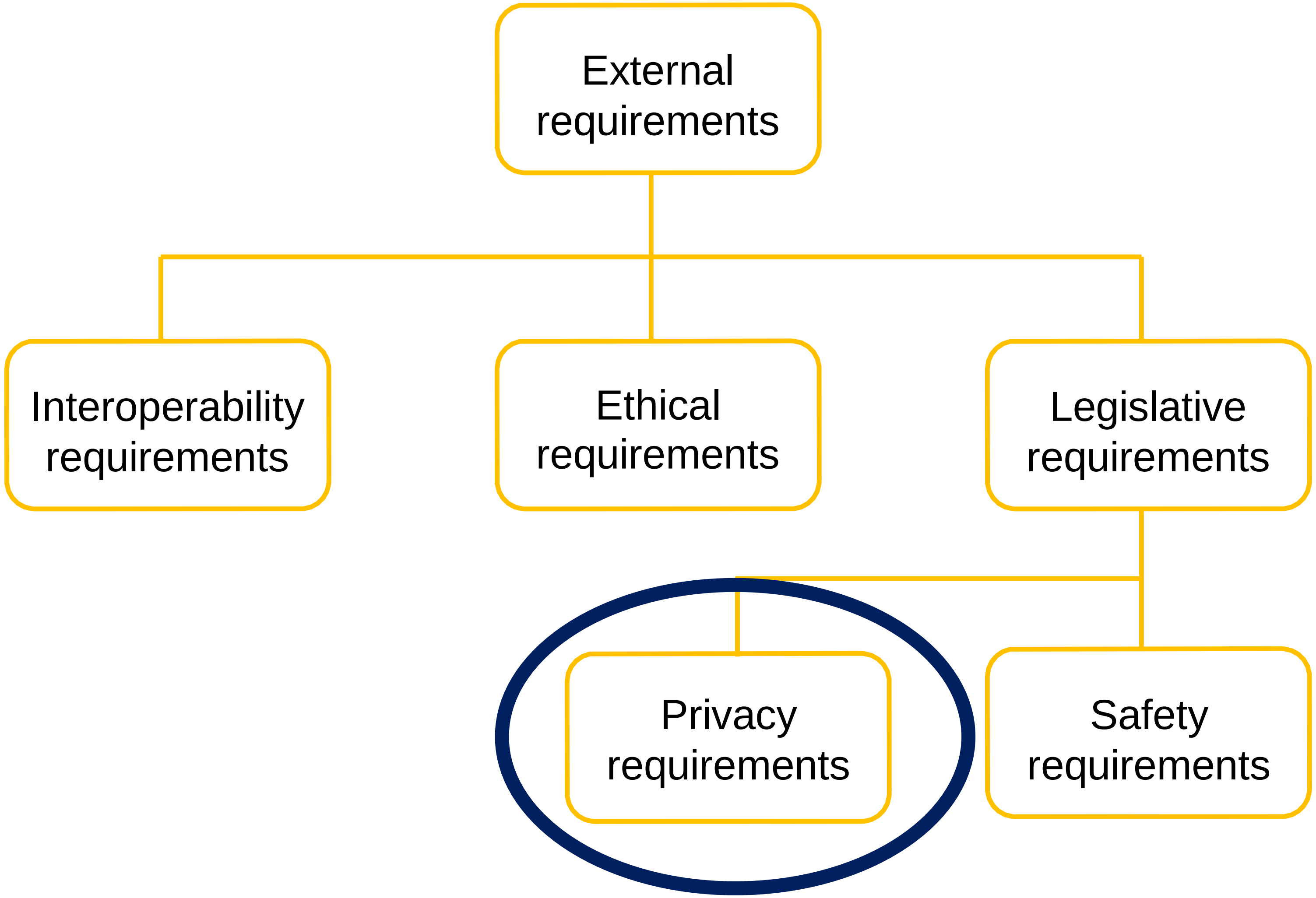
External Requirements



External Requirements



External Requirements



External requirements

Interoperability requirements

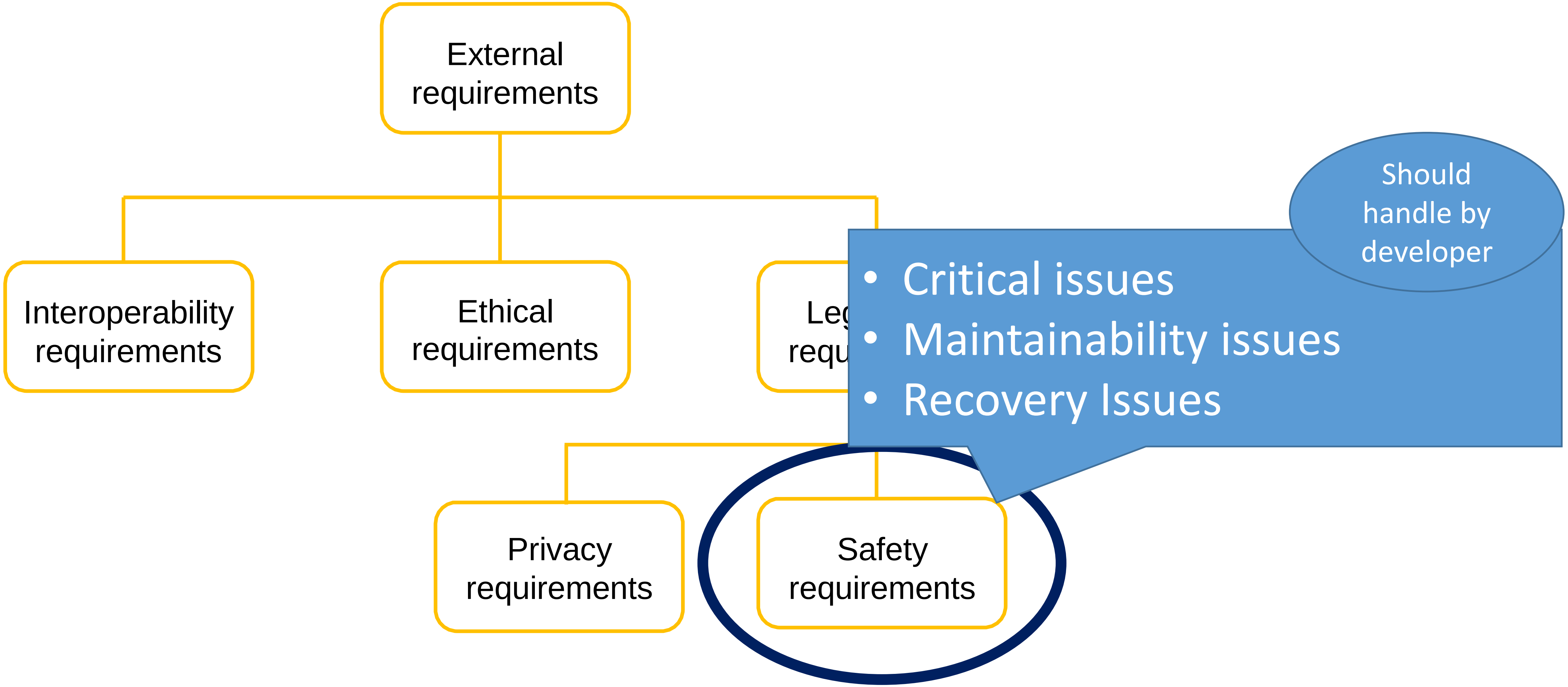
Ethical requirements

Legislative requirements

Privacy requirements

Safety requirements

External Requirements



- Critical issues
- Maintainability issues
- Recovery Issues

External Requirements

- The system **shall not** disclose any personal **information** about members of the library system to other members except system administrators

- The system **shall** comply with the local and national laws regarding the use of software tools

License

Domain Requirements

- Requirements that come from the application domain and reflect fundamental characteristics of that application domain
- Can be functional or non-functional
- These requirements, sometimes, are not explicitly mentioned, as domain experts find it difficult to convey them. However, their absence can cause significant dissatisfaction

Domain Requirements

- Domain requirements can impose strict constraints on **solutions**

- This is particularly true for scientific and engineering domains

- Domain-specific terminology can also cause confusion

Example: In a commission-based sales businesses, there is no concept of negative commission. However if care is not taken novice developers can be tempted into developing systems, which calculate negative commission

Inverse Requirements

- They explain what the system **shall not do**.

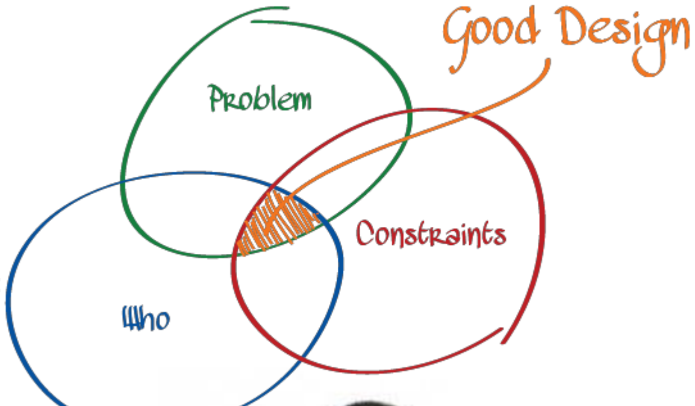
- Many people find it convenient to describe their needs in this manner
- These requirements indicate the **indecisive** nature of customers about certain aspects of a new software product
- Example: The system shall not use red color in UI, whenever it asking for inputs from the user.
- Safety and security requirements are usually stated in this manner

Design and Implementation Constraints

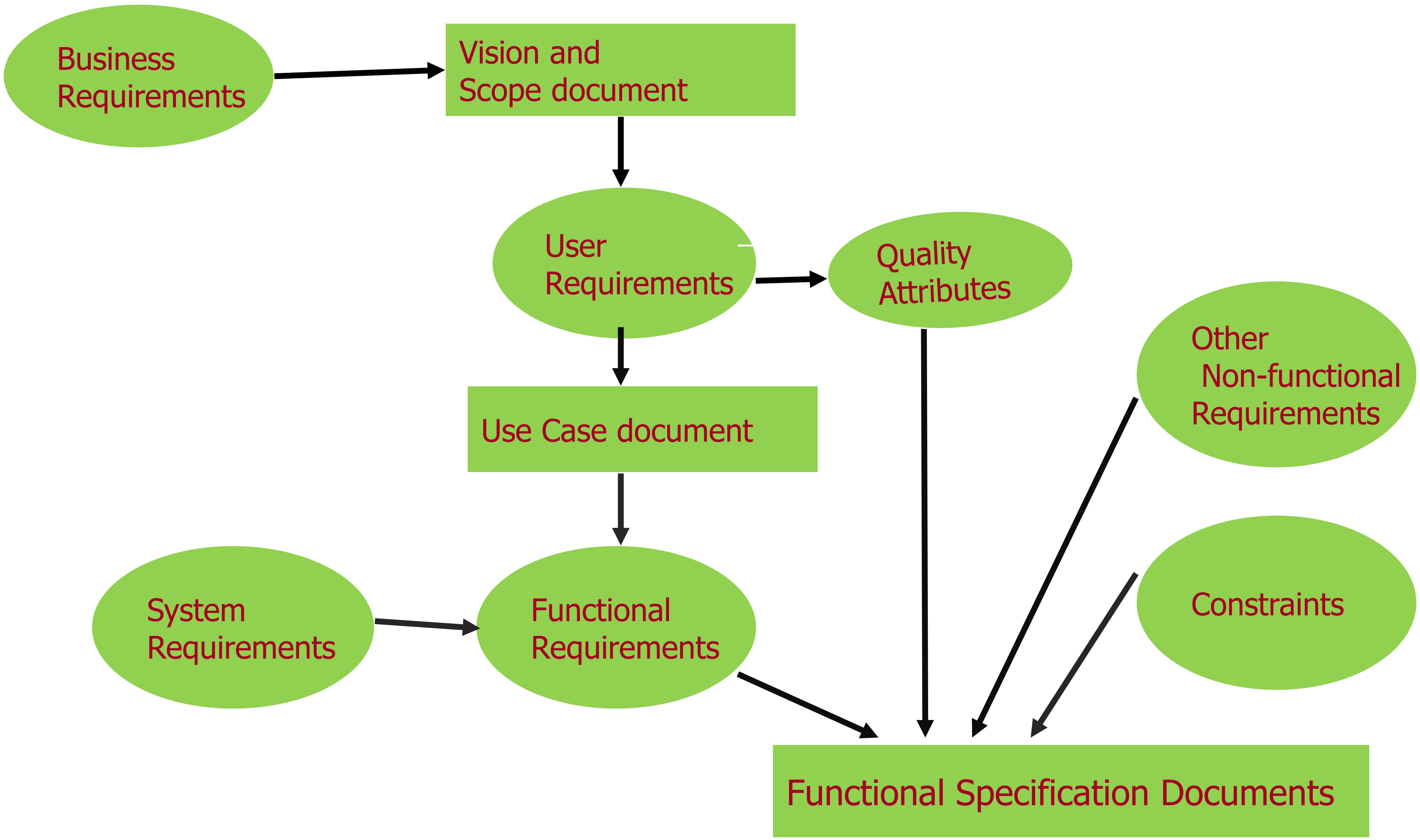
- They are development guidelines within which the designer must work, which can seriously limit design and implementation options
- Can also have impact on human resources

Design and Implementation Constraints

- Example:
  - ✓ The system shall be developed using the **Microsoft .Net platform**
  - ✓ The system shall be developed using **open source tools** and shall run on
    Linux operating system
  - ✓ The System should be implemented using **C# language**
  - ✓ The software must fit into the memory of a 512Kbyte machine.



Relationship of Several components of Software Requirements



Business Requirements

Vision and Scope document

User Requirements

Quality Attributes

Other Non-functional Requirements

Use Case document

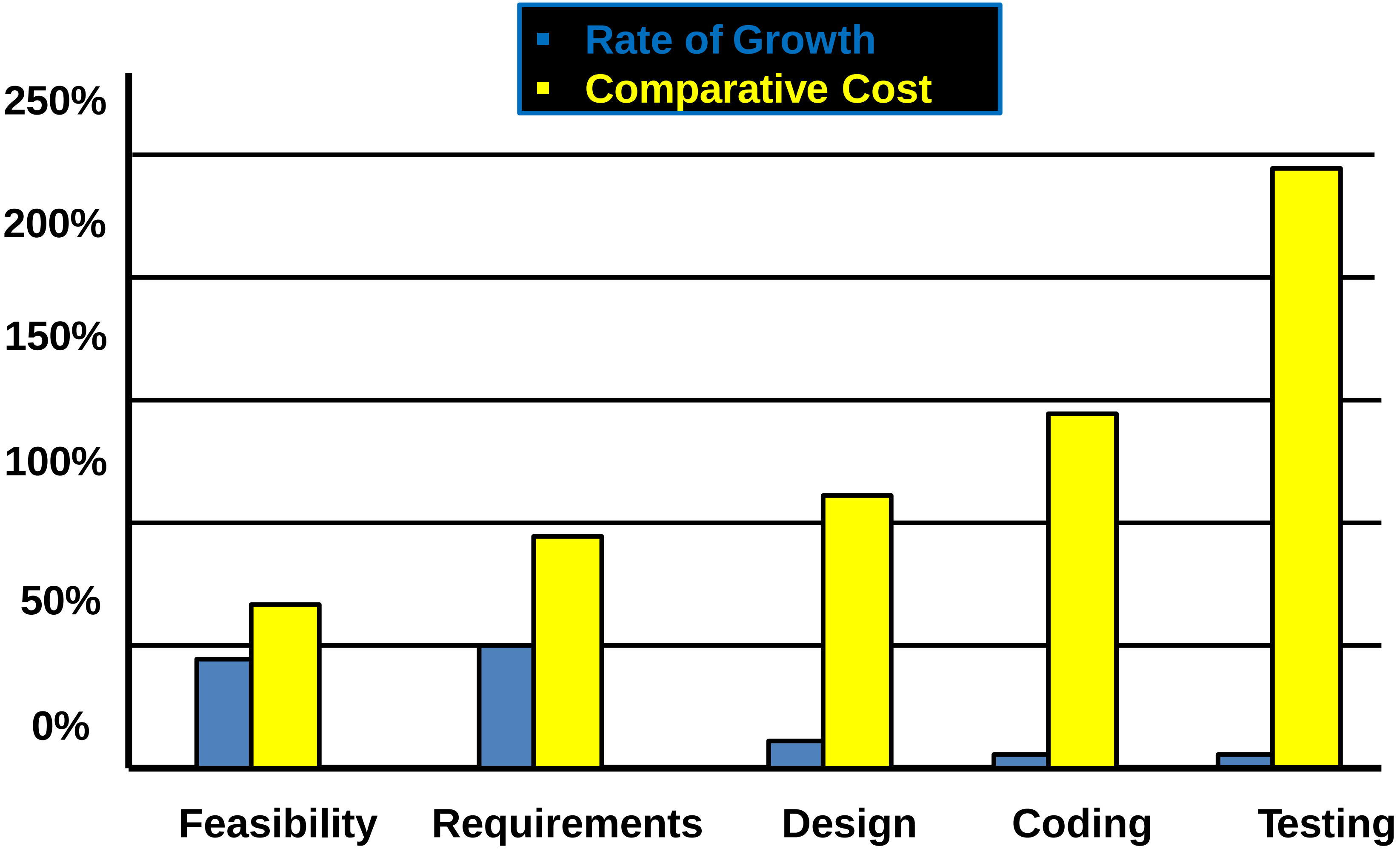
Constraints

System Requirements

Functional Requirements

Functional Specification Documents

Requirement Origin v/s Costs



Some Risks From Inadequate Requirement Process

- Insufficient user involvement leads to unacceptable products
- Creeping user requirements contribute to overruns and degrade product quality
- Ambiguous requirements lead to ill-spent time and rework
- Gold-plating by developers and users adds unnecessary features

Some Risks From Inadequate Requirement Process

- Minimal specifications lead to missing key requirements

- Overlooking the needs of certain stake holders leads to dissatisfied customers

- Incompletely defined requirements make accurate **project planning and tracking impossible**

Ambiguous Requirements

- The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.

- Allows user to cancel sales orders.

- The system provides searching of data quickly.

- “The system is user friendly.”

Ambiguous Requirements

- “Software produces a print normally in ten seconds.”
- The staff shall be able to use all the system functionalities. The average number of errors made by users shall not exceed two per hour of system use.

Ambiguous Requirements

- “The operator identity consists of the operator name and password; the password consists of six digits. It should be displayed on the security screen and deposited in the login file when an operator logs into the system.”

Characteristics of Good Requirements

- Numbered
- Inspected
- Unambiguous
- Testable
- Complete
- Consistent
- Understandable
- Traceable
- Feasible
- Modifiable

Task - 1

Write down five functional requirements and five non-functional requirements of a mobile telephone.

Task - 2

Write down five functional requirements and five non-functional requirements of an online examination system.

Quality Factors

- Portability
- Reusability
- Interoperability
- Survivability
- Safety
- Manageability
- Supportability
- Replaceability
- Functionality

- Correctness
- Reliability
- Efficiency
- Usability
- Integrity
- Maintainability
- Flexibility
- Testability
- Security

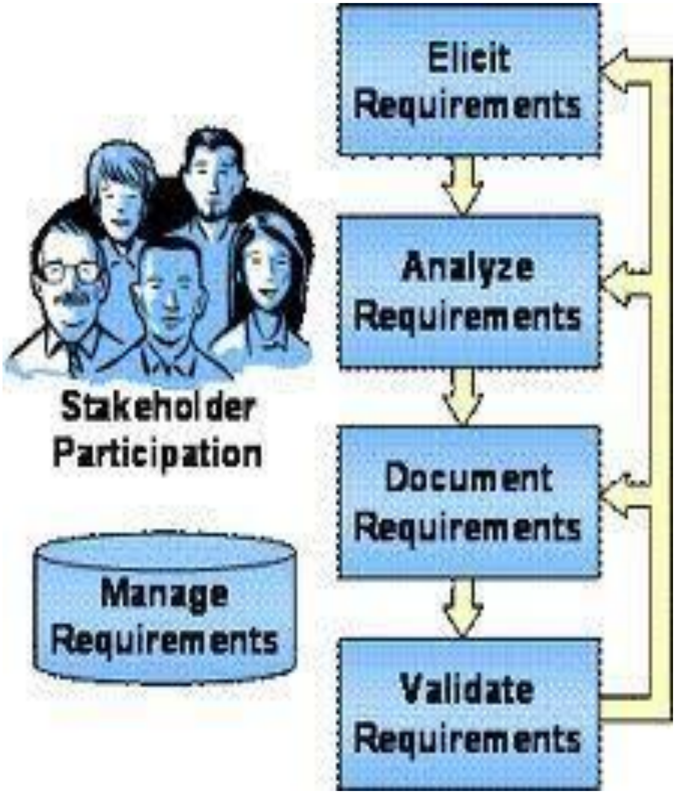
The Problems with our Requirements Practices

- We have trouble **understanding the requirements** that we do acquire from the customer
- We often record requirements in a **disorganized manner**
- We spend far too **little time verifying** what we do record
- We **allow change to control us**, rather than establishing mechanisms to control change
- Most importantly, we **fail to establish a solid foundation for the system or software** that the user wants built

(more on next slide)

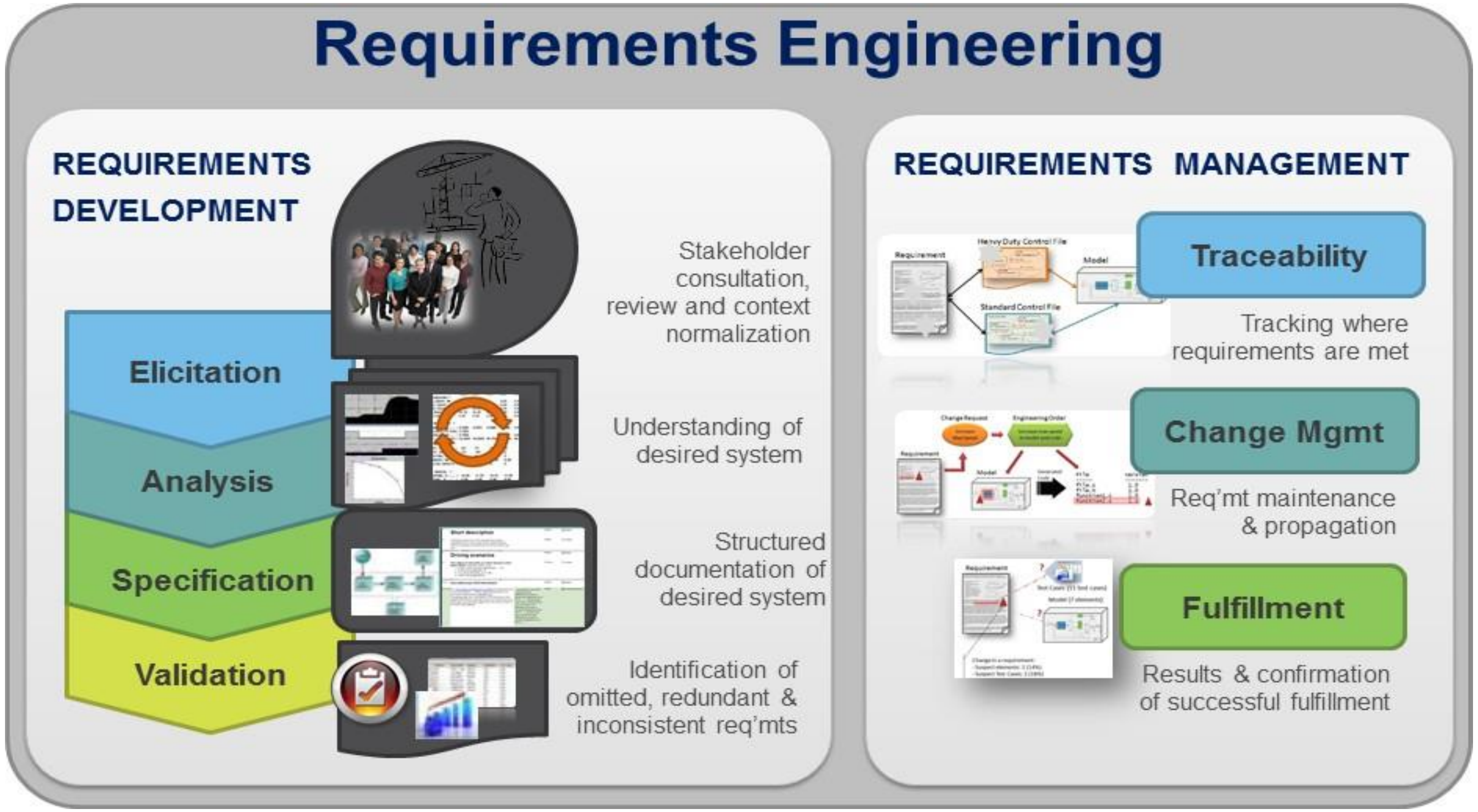
Requirements Engineering Process

- **Elicitation:** work with the customer on gathering requirements
- **Analysis:** process this information to understand it, classify in various categories, and relate the customer needs to possible software requirements
- **Specification:** Structure the customer input and derived requirements as written documents and diagrams
- **Validation:** you'll ask your customer to confirm that what you've written is accurate and complete and to correct errors.



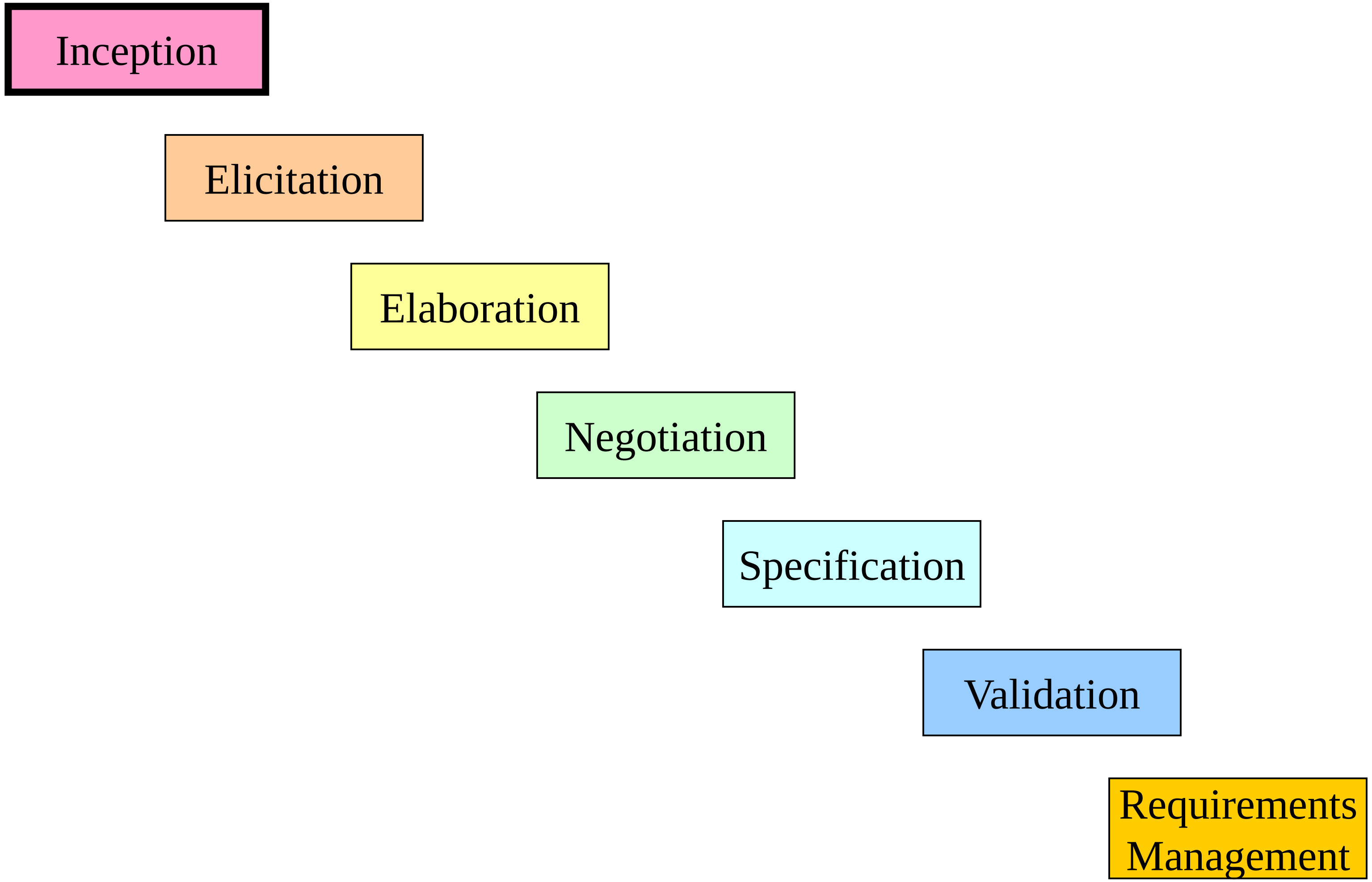
Requirements Management

- **Requirements management** is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders.


Requirements Engineering
REQUIREMENTS DEVELOPMENT
Elicitation
Analysis
Specification
Validation
Stakeholder consultation, review and context normalization
Understanding of desired system
Structured documentation of desired system
Identification of omitted, redundant & inconsistent req'mts
REQUIREMENTS MANAGEMENT
Traceability
Tracking where requirements are met
Change Mgmt
Req'mt maintenance & propagation
Fulfillment
Results & confirmation of successful fulfillment

Requirements Engineering Tasks

- Seven distinct tasks
  - Inception
  - Elicitation
  - Elaboration
  - Negotiation
  - Specification
  - Validation
  - Requirements Management
- Some of these tasks **may occur in parallel** and all are adapted to the needs of the project
- All strive to define what the customer wants
- All serve to establish a solid foundation for the design and construction of the software



Inception

Elicitation

Elaboration

Negotiation

Specification

Validation

Requirements Management

Inception Task

- During inception, the requirements engineer asks a set of questions to establish…
  - A basic understanding of the problem
  - The people who want a solution
  - The nature of the solution that is desired
  - The effectiveness of preliminary communication and collaboration between the customer and the developer
- Through these questions, the requirements engineer needs to…
  - Identify the stakeholders
  - Recognize multiple viewpoints
  - Work toward collaboration
  - Break the ice and initiate the communication

The First Set of Questions

These questions focus on the customer, other stakeholders, the overall goals, and the 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?

The Next Set of Questions

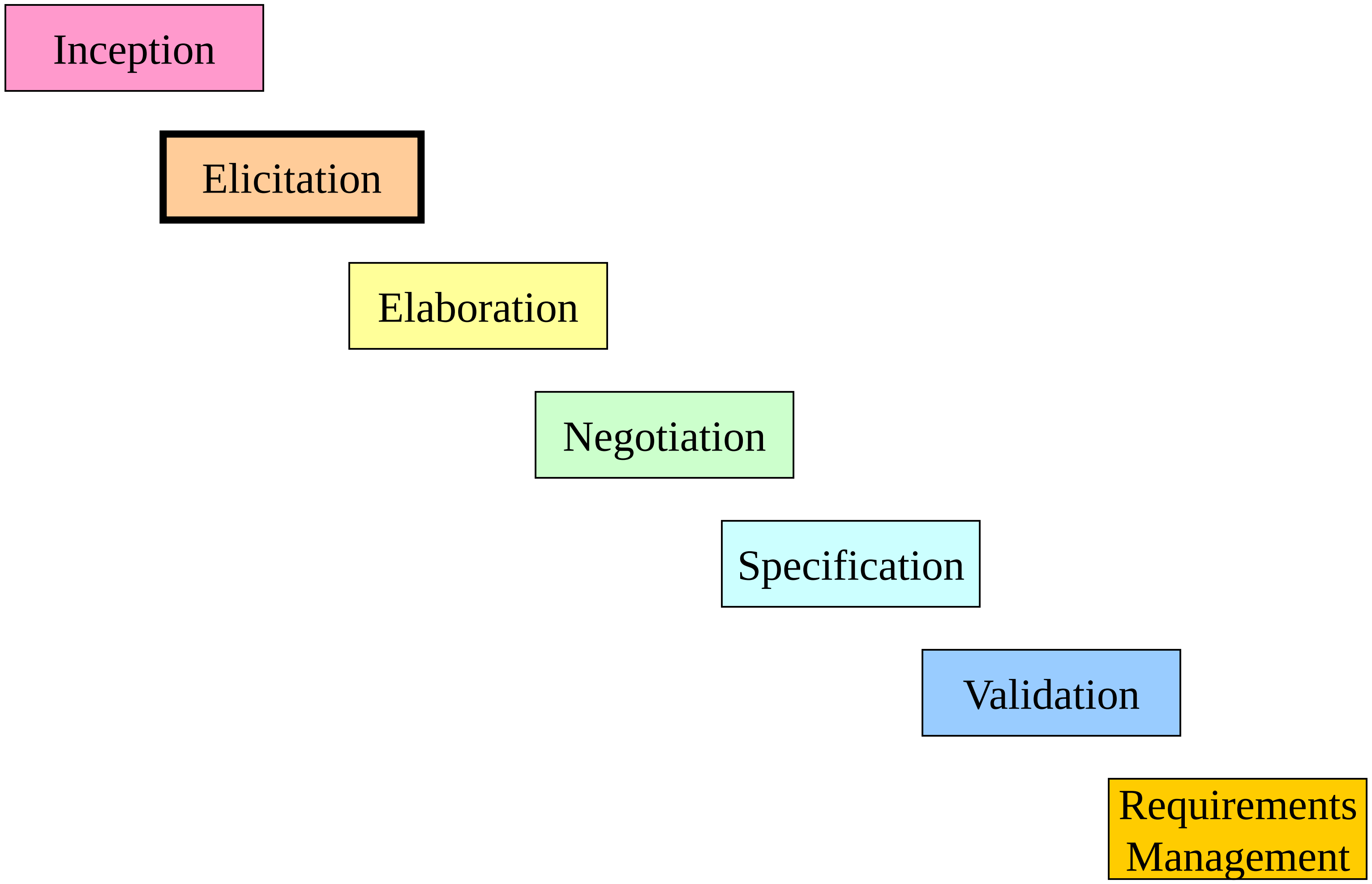
These questions enable the requirements engineer to gain a better understanding of the problem and allow the customer to voice his or her perceptions about a solution

- 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 (or describe) the business environment in which the solution will be used?
- Will special performance issues or constraints affect the way the solution is approached?

The Final Set of Questions

These questions focus on the effectiveness of the communication activity itself

- 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?



Inception

Elicitation

Elaboration

Negotiation

Specification

Validation

Requirements Management

Elicitation Task

- Eliciting requirements is difficult because of
  - **Problems of scope** in identifying the boundaries of the system or specifying too much technical detail rather than overall system objectives
  - **Problems of understanding** what is wanted, what the problem domain is, and what the computing environment can handle (Information that is believed to be "obvious" is often omitted)
  - **Problems of volatility** because the requirements change over time
- **Elicitation may be accomplished through several activities**
  - **Collaborative requirements gathering (Req workshops, Brainstorming)**
  - **Quality function deployment**

Basic Guidelines of Collaborative Requirements Gathering

- **Meetings** are conducted and attended by both **software engineers, customers, and other interested stakeholders**

- 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**" (customer, developer, or outsider) controls the meeting

- A "**definition mechanism**" is used such as work sheets, flip charts, wall stickers, electronic bulletin board, chat room, or some other virtual forum

- The **goal** is to identify the **problem**, **propose** elements of the solution, **negotiate** different approaches, and specify a preliminary set of solution requirements

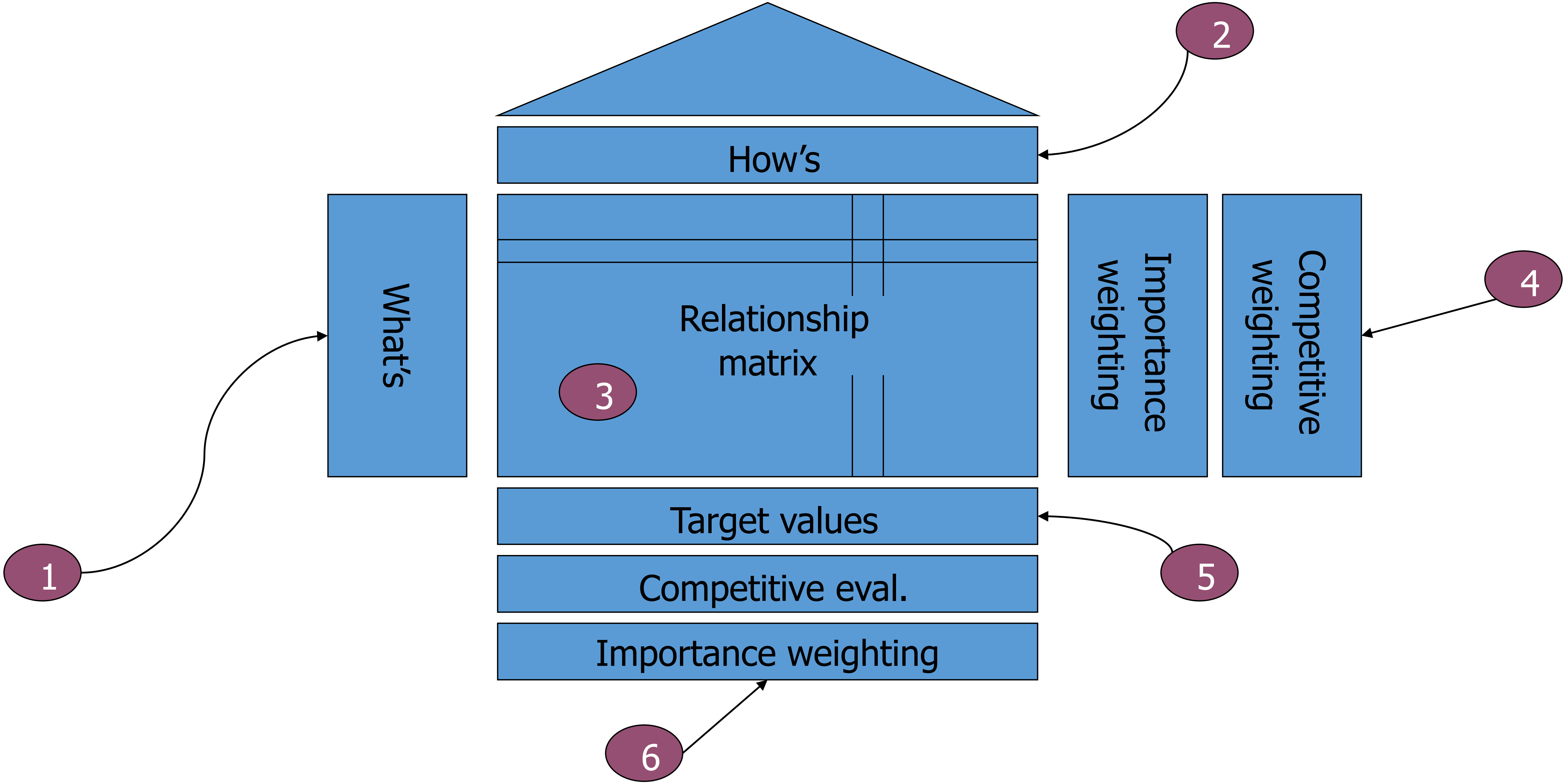
Quality Function Deployment

- 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

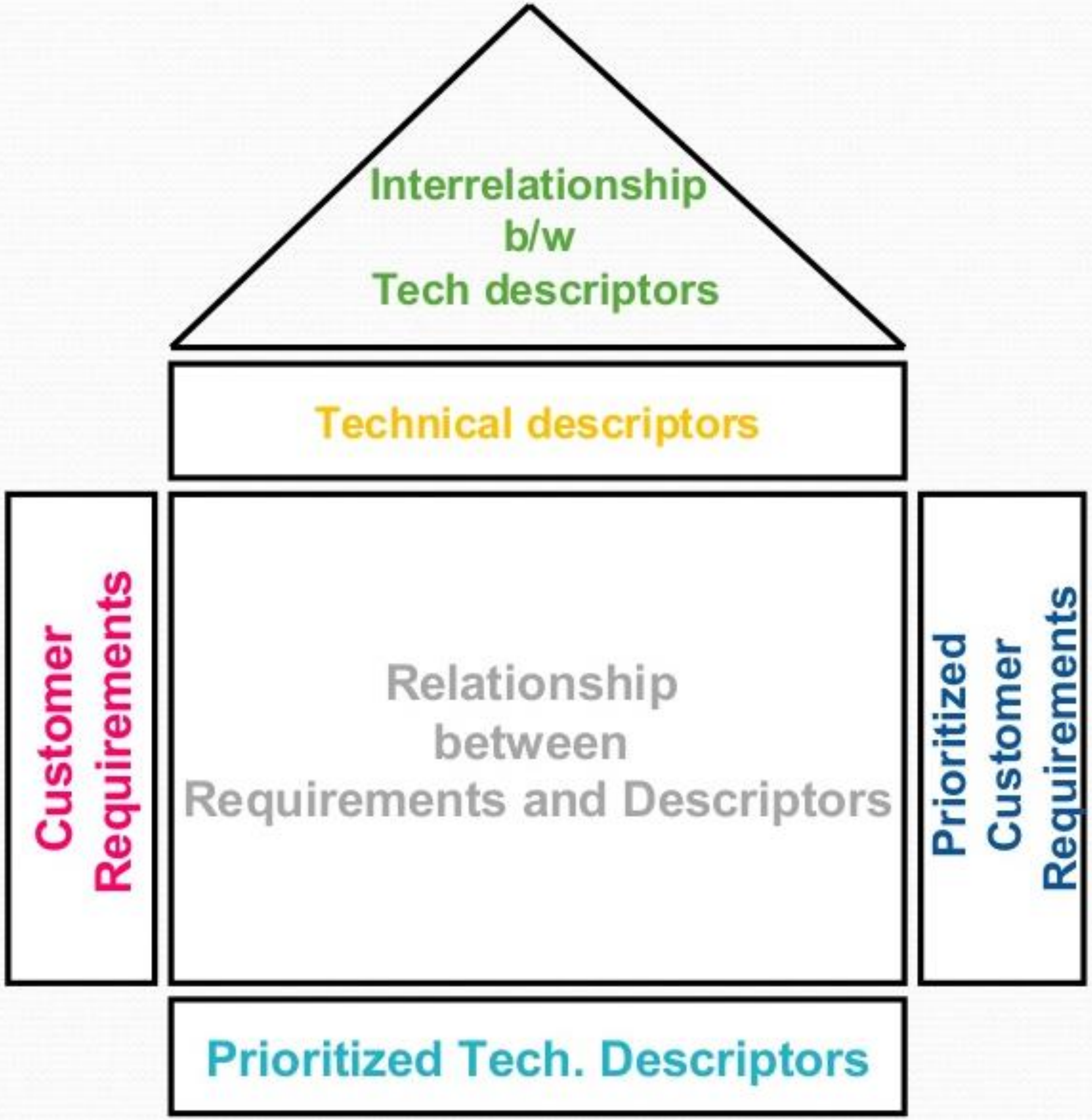
QFD process (1)

- The basic idea of QFD is to construct relationship matrices between customer needs, technical requirements, priorities and (if needed) competitor assessment.
- To achieve this the following process is prescribed:
  1. Identify stakeholder’s attributes or requirements
  2. Identify technical features of the requirements
  3. Relate the requirements to the technical features
  4. Conduct an evaluation of competing products
  5. Evaluate technical features and specify a target value for each feature
  6. Prioritize technical features for development effort.

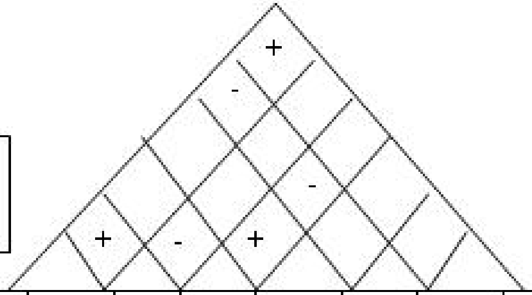
QFD process (2)



House of Quality


Interrelationship b/w Tech descriptors
Technical descriptors
Customer Requirements
Relationship between Requirements and Descriptors
Prioritized Customer Requirements
Prioritized Tech. Descriptors

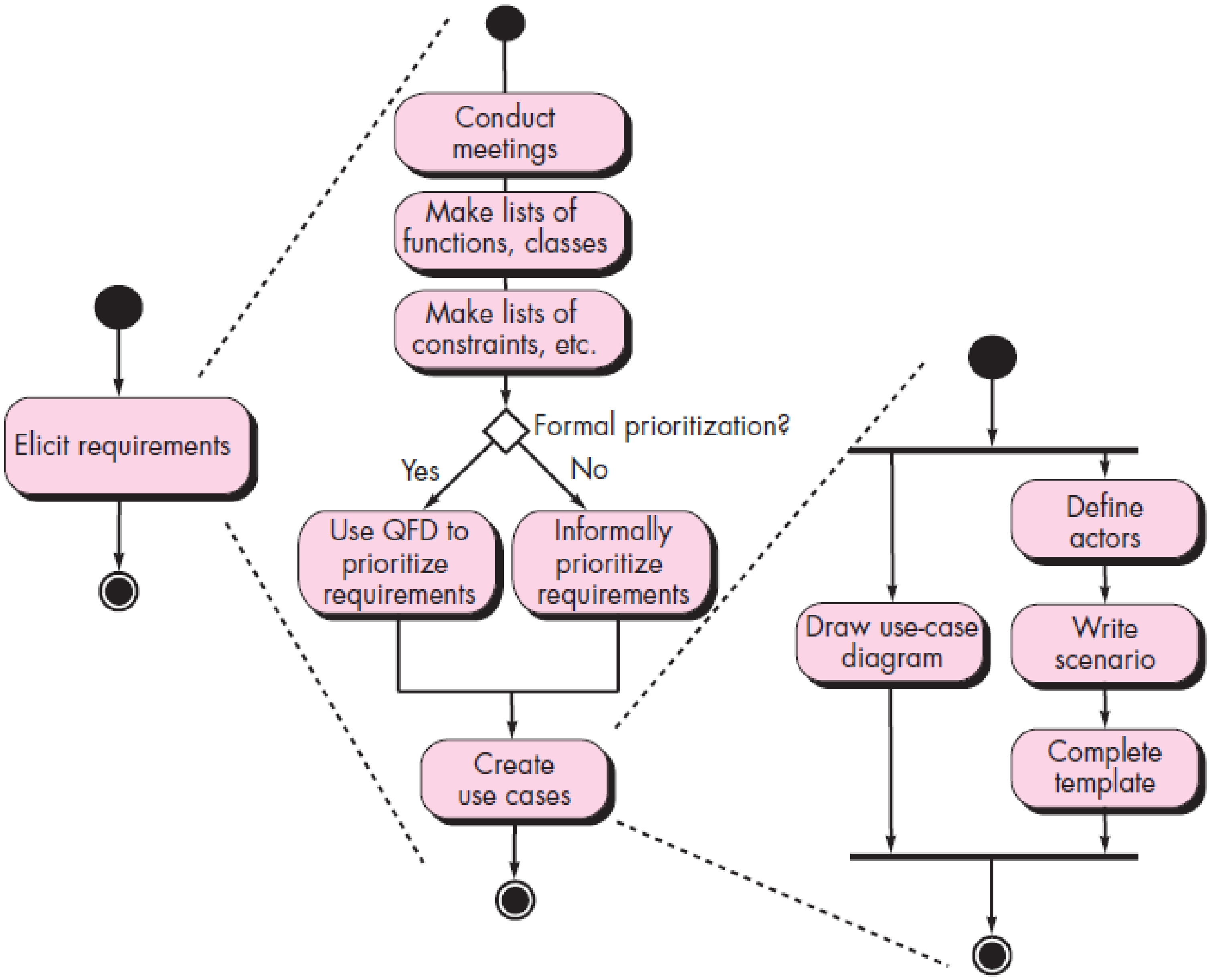
INTERACTIONS:
+ Positive Relationship
- Negative Relationship

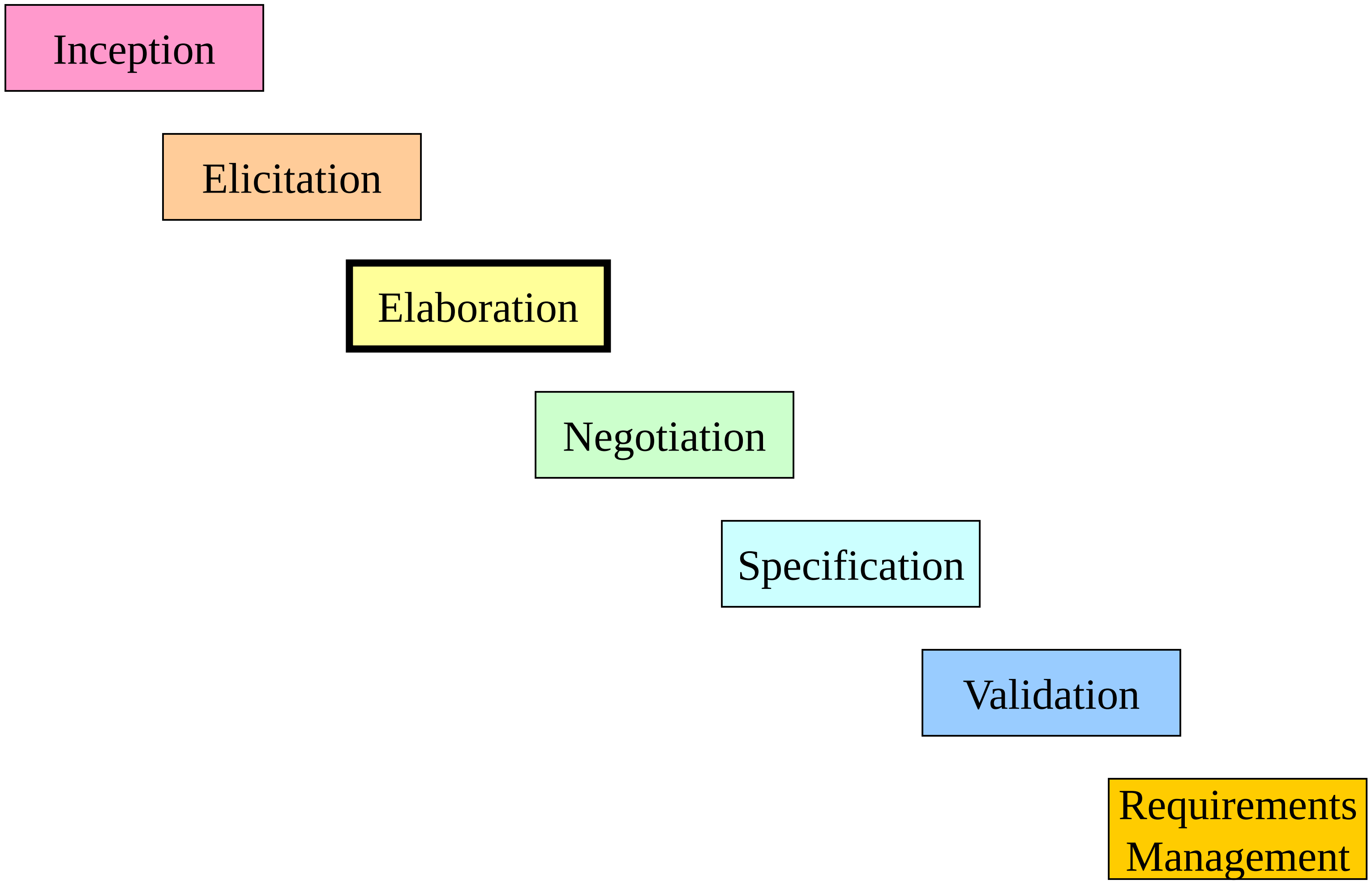


	PRIORITY	Product Design Requirements	Smaller Casing (length reduction)	Less Gears	Long Battery Life	Manual Override	LED installed	Finger Grooves	Competitive Evaluation
Customer Requirements									
Safety	5		-			+	+	+	Δ Ⓜ B
Durability	4		-						Δ B Ⓜ
Ease of Use	4		+		+	+	+	+	ⓂB Δ
Cost +=less cost	3		+	+	+	-	-	-	Δ B Ⓜ
Weight +=less weight	3		+	+		-	-		Ⓜ B Δ
Battery Life	4			-	+		-		BΔⓂ
Technical Evaluation	5 1		Δ B Ⓜ	Δ ⓂB	ΔⓂB	B ΔⓂ			RELATIONSHIPS: Δ Black & Decker Ⓜ Handy Gourmet B One-Touch
Target Value			1.25in.	2	100 cans	2 lb in	5,000 hrs	4 in.	
Technical Difficulty			3	2	3	5	4	3	
Importance Rating			2	3	4	4	4	3	

Figure 5.3

UML activity diagrams for eliciting requirements





Inception

Elicitation

Elaboration

Negotiation

Specification

Validation

Requirements Management

Goals of Analysis Modeling

- Provides the first technical representation of a system
- Is easy to understand and maintain
- Deals with the problem of size by partitioning the system
- Uses graphics whenever possible
- Differentiates between essential information versus implementation information
- Helps in the tracking and evaluation of interfaces
- Provides tools other than narrative text to describe software logic and policy

Analysis Rules of Thumb

- The analysis model should **focus on requirements that are visible within the problem** or business domain
  - The level of abstraction should be relatively high

Our concern about what we will implement

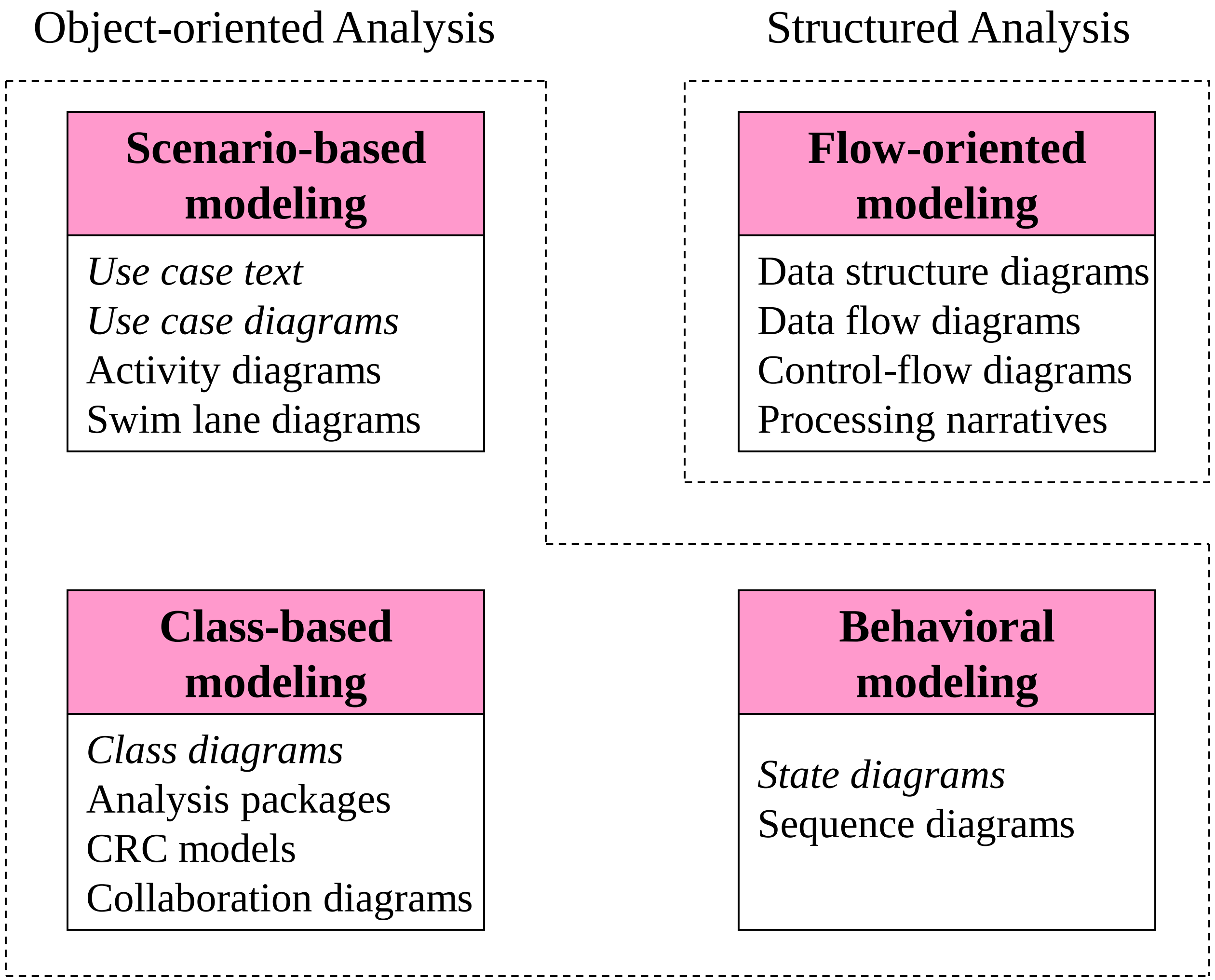
Not worried about that how we will implement

- Each element of the analysis model **should add to an overall understanding of software requirements** and provide insight into the following
  - Information domain, function, and behavior of the system
- The model should **minimize coupling** throughout the system
  - Reduce the level of interconnectedness among functions and classes
- The model should provide value to all stakeholders
- The model should be kept as simple as can be

Analysis Modeling Approaches

- Structured analysis
  - Considers **data** and the **processes** that transform the data as separate entities
  - Data is modeled in terms of only attributes and relationships (but no operations)
  - Processes are modeled to show the 1) input data, 2) the transformation that occurs on that data, and 3) the resulting output data
- Object-oriented analysis
  - Focuses on the definition of **classes** and the manner in which they **collaborate** with one another to fulfill customer requirements

Elements of the Analysis Model



Object-oriented Analysis

Structured Analysis

Scenario-based modeling

Use case text
Use case diagrams
Activity diagrams
Swim lane diagrams

Flow-oriented modeling

Data structure diagrams
Data flow diagrams
Control-flow diagrams
Processing narratives

Class-based modeling

Class diagrams
Analysis packages
CRC models
Collaboration diagrams

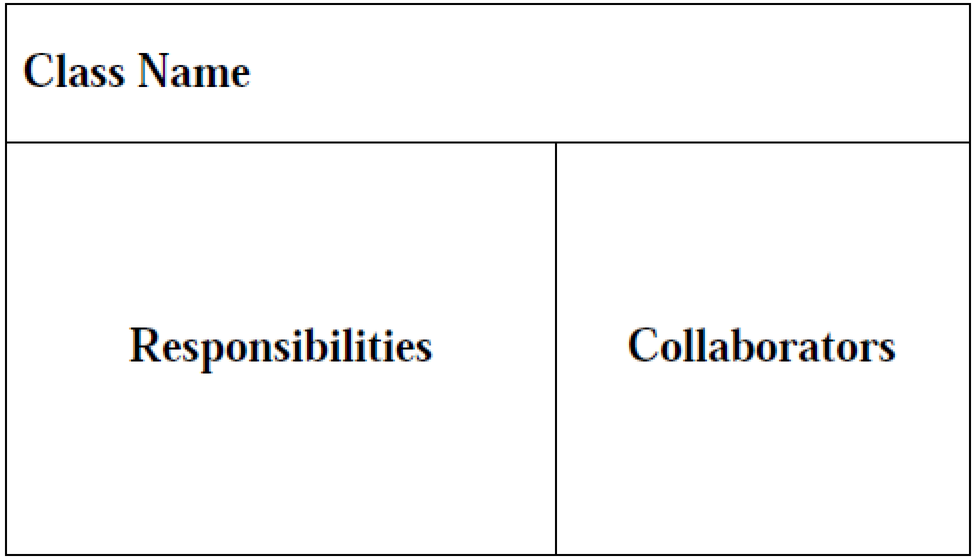
Behavioral modeling

State diagrams
Sequence diagrams

CRC Card

CRC Cards

- A CRC card is divided into three sections (Beck & Cunningham, 1989; Ambler, 1995)



Class Name	
Responsibilities	Collaborators

Responsibilities and Collaborations

- A responsibility is anything that a class knows or does.
- Sometimes a class will have a responsibility to fulfill, but will not have enough information to do it. When this happens it has to collaborate with other classes to get the job done.

CRC Model

- A CRC model is a collection of CRC cards that represent whole or part of an application or problem domain.
- The most common use for CRC models is to gather and define the user requirements for an object-oriented application.
- Collaborating cards are placed close to each other

Inventory Item	
Item number	
Name	
Description	
Unit Price	
Give price	

Order	
Order number	Order Item
Date ordered	Customer
Date shipped	
Order items	
Calculate order total	
Print invoice	
Cancel	

Order Item	
Quantity	Inventory item
Inventory item	
Calculate total	

Customer	
Name	Order
Phone number	Surface Address
Customer number	
Make order	
Cancel order	
Make payment	

Surface Address	
Street	
City	
State	
Zip	
Print label	

Creating CRC Model

1. Put together the CRC modeling team.
2. Organize the modeling room.
3. Do some brainstorming.
4. Explain the CRC modeling technique.
5. Iteratively perform the steps of CRC modeling.
6. Perform use-case scenario testing.

CRC Team

- **Business Domain Experts (BDEs).**
- **Facilitator.**
- **Scribe(s).**
- **Observers.**

Room preparation

- **Reserve a meeting room that has something write on.**
- **Bring CRC modeling supplies.**
- **Have a modeling table**
- **Have chairs and desks for the scribe(s).**
- **Have chairs for the BDEs.**
- **Have chairs for the observers**

Brainstorming

Set of questions:

- Who is this system for?
- What will they do with the system?
- Why do we do this?
- Why do we do this the way that we do?
- What business needs does this system support?
- What do/will our customers want/demand from us?
- How is the business changing?
- What is our competition doing? Why? How can we do it better?
- Do we even need to do this?
- If we were starting from scratch, how would we do this?
- Just because we were successful in the past doing this, will we be successful in the future?

Brainstorming (cont…)

- Can we combine several jobs into one? Do we want to?
- How will people’s jobs be affected? Are we empowering or disempowering them?
- What information will people need to do their jobs?
- Is work being performed where it makes the most sense?
- Are there any trivial tasks that we can automate?
- Are people performing only the complex tasks that the system can’t handle?
- Will the system pay for itself?
- Does the system support teamwork, or does it hinder it?
- Do our users have the skills/education necessary to use this system? What training will they need?
- What are our organization’s strategic goals and objectives? Does this system support them?
- How can we do this faster?
- How can we do this cheaper?
- How can we do this better?

Explain the CRC Modeling Technique

- Facilitator should describe the CRC modeling process. This usually takes between ten and fifteen minutes and will often include the creation of several example CRC cards.
- Lead BDEs through creation of example CRC cards

Perform The Iterative Steps of CRC Modeling

- **Find classes**
  - Look for anything that interacts with the system, or is part of the system
  - Ask yourself “Is there a customer?”
  - Follow the money
  - Look for reports generated by the system
  - Look for any screens used in the system
  - Immediately prototype interface and report classes
  - Look for the three to five main classes right away
  - Create a new card for a class immediately
  - Use one or two words to describe the class
  - Class names are singular

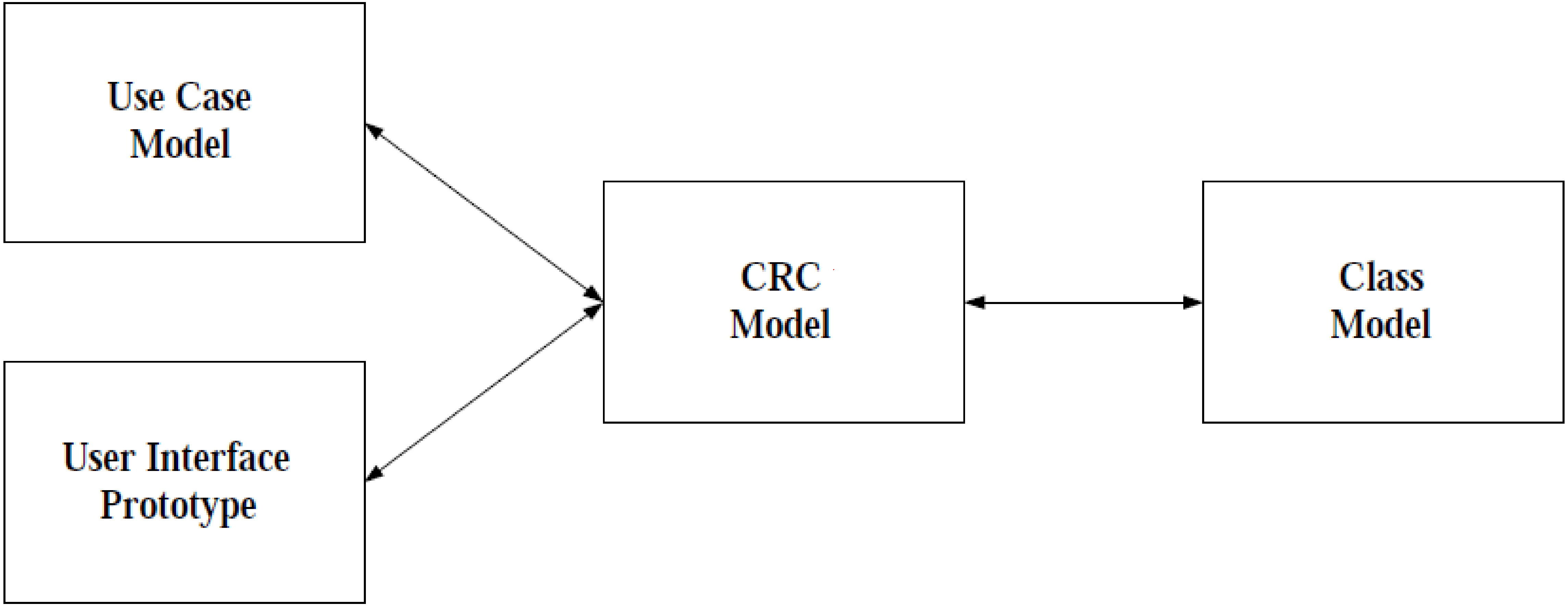
- Find responsibilities
    - Ask yourself what the class knows
    - Ask yourself what the class does
    - If you’ve identified a responsibility, ask yourself what class it "belongs" to
    - Sometimes get responsibilities that we won’t implement, and that’s OK
    - Classes will collaborate to fulfill many of their responsibilities
- Define collaborators
    - Collaboration occurs when a class needs information that it doesn’t have
    - Collaboration occurs when a class needs to modify information that it doesn’t have
    - There will always be at least one initiator of any given collaboration
    - Sometimes the collaborator does the bulk of the work
    - Don’t pass the buck
    - New responsibilities may be created to fulfill the collaboration

- Define use-cases
  - The BDEs will identify them as responsibilities of actor classes
  - Do some brainstorming
  - Transcribe the scenarios onto cards
- Arrange the cards on the table
  - Cards that collaborate with each other should be close to one another on the desk
  - The more that two cards collaborate, the closer that they should be on the desk
  - Expect to be moving the cards around a lot at the beginning
  - Put "**busy**" cards towards the center of the table
  - Actually move them around
  - People will identify relationships/associations between classes as they move them around

Perform **Use-Case** Scenario Testing

- Call out a new scenario (Select a new use case or new scenario)
- Determine which card should handle the responsibility
- Update the cards whenever necessary
- Describe the processing logic
- Collaborate if necessary
- Pass the ball back when done

How CRC Modeling Fits in?



CRC Modeling Tips and Techniques

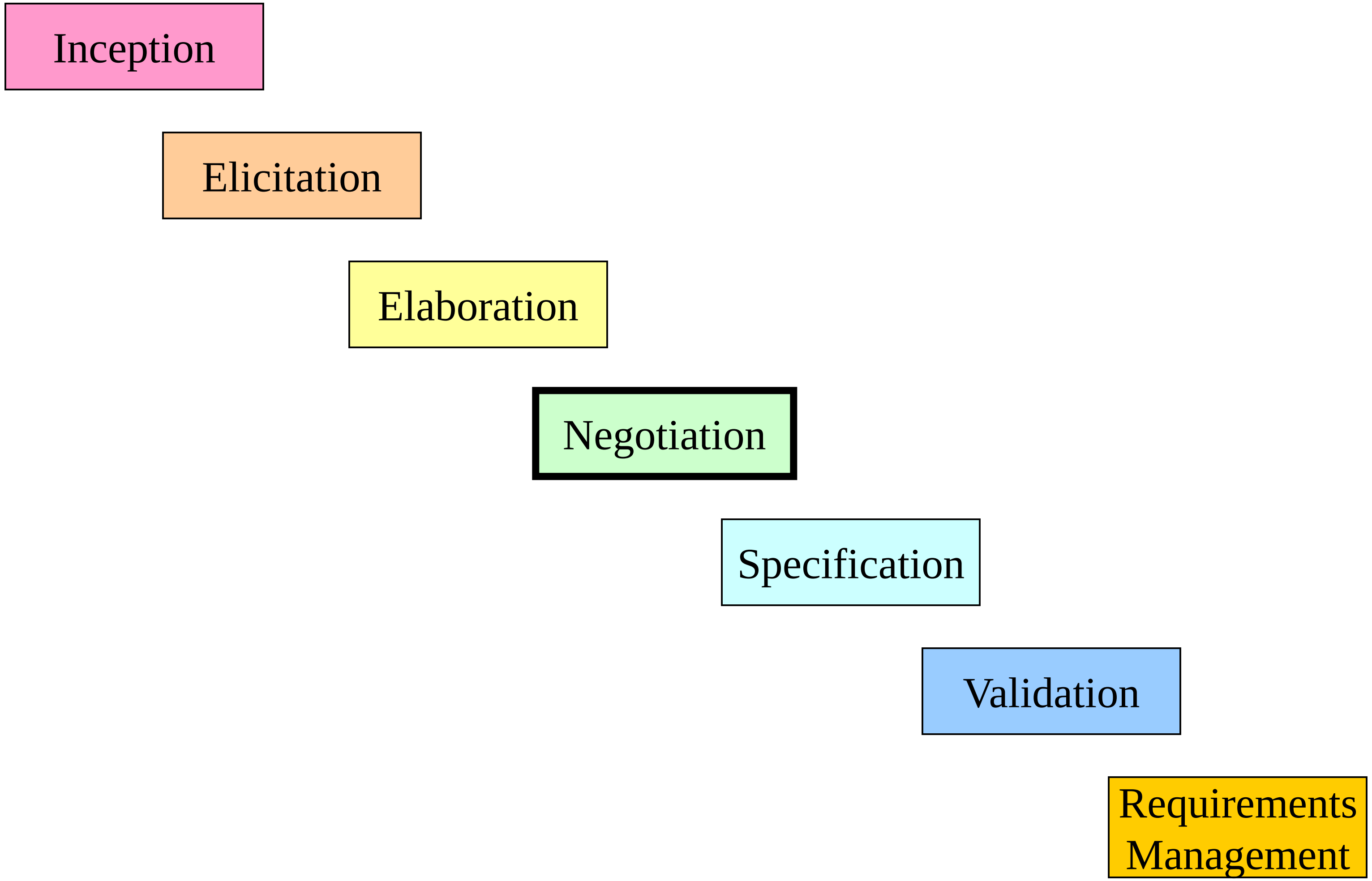
- Send ahead an agenda a few days before the modeling session
- Prominently display the CRC definitions(etc. responsibility, collaboration)
- Use their terminology
- Keep it low tech
- Expect to prototype
- Expect to take a few days
- Get management support
- Include CRC modeling in your system development life cycle
- Do CRC modeling with front-line staff only

Advantages of CRC Modeling

- The experts do the analysis
- User participation increased
- Breaks down communication barriers
- It’s simple and straightforward
- It’s non-threatening to users
- It’s inexpensive and portable
- It goes hand-in-hand with prototyping
- It leads directly into class diagramming

Disadvantages of CRC Modeling

- It’s threatening to some developers
- It’s hard to get users together
- CRC cards are limited



Inception

Elicitation

Elaboration

Negotiation

Specification

Validation

Requirements Management

Negotiation Task

- During negotiation, the software engineer **reconciles the conflicts** between what the customer wants and what can be achieved given **limited business resources**

- **Requirements are ranked** (i.e., prioritized) by the customers, users, and other stakeholders

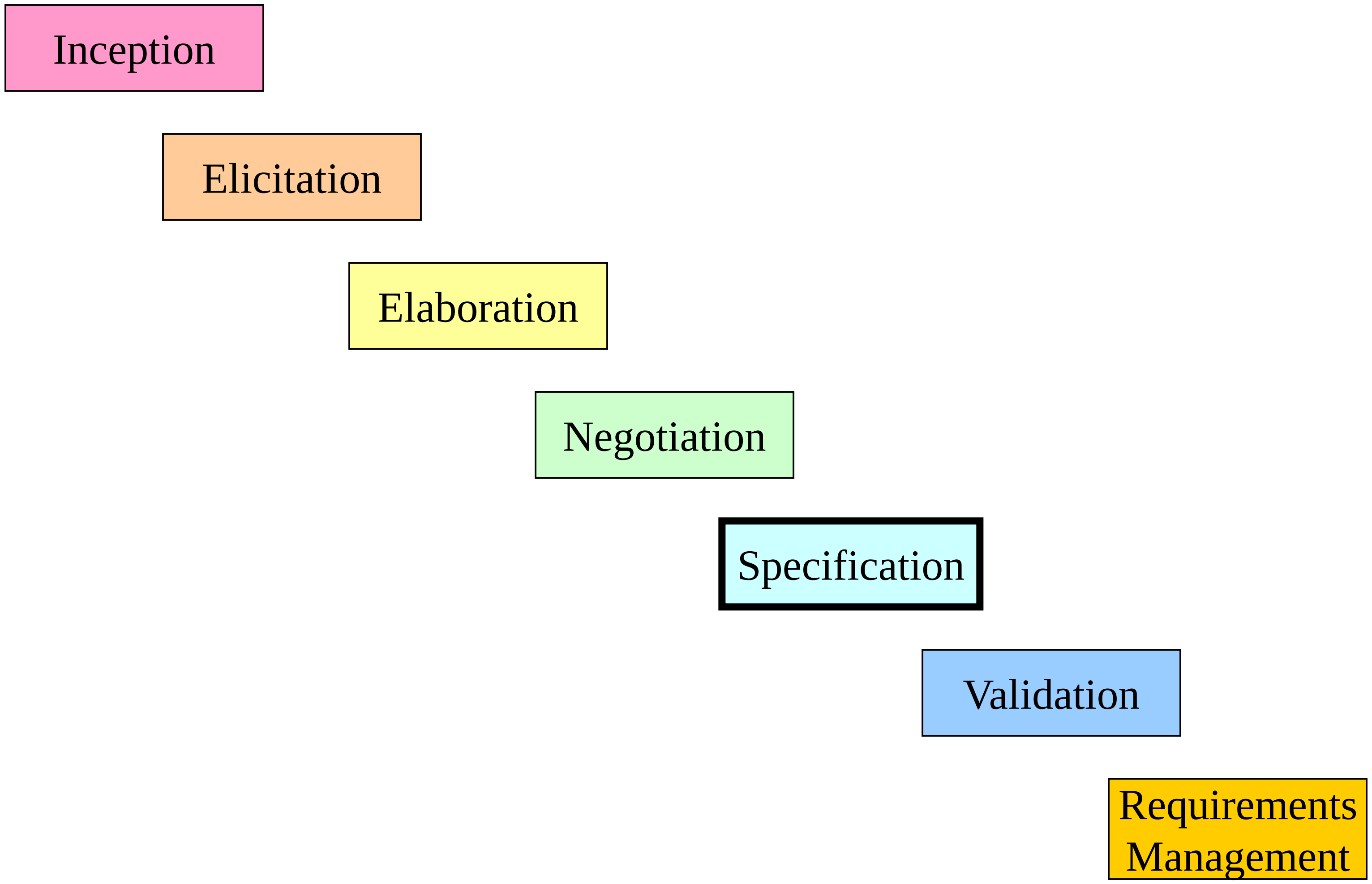
- **Risks** associated with each requirement are identified and analyzed

- **Rough guesses of development effort** are made and used to assess the impact of each requirement on **project cost and delivery time**

- Using an **iterative approach**, **requirements are eliminated**, **combined** and/or modified so that each party achieves some measure of satisfaction

Art of Negotiation

- Recognize that it is **not competition**
- Map out a **strategy**
- Listen **actively**
- **Focus** on the other party’s interests
- **D**on’t let it get personal
- Be creative
- Be ready to commit



Inception

Elicitation

Elaboration

Negotiation

Specification

Validation

Requirements Management

Specification Task

- A specification is the final work product produced by the requirements engineer

- It is normally in the form of a software requirements specification

- It serves as the **foundation** for **subsequent software engineering activities**

- It describes the function and performance of a computer-based system and the constraints that will govern its development

- It **formalizes** the informational, functional, and behavioral requirements of the proposed software in both a graphical and textual format

Typical Contents of a Software Requirements Specification

- Requirements
  - Required states and modes
  - Software requirements grouped by capabilities (i.e., functions, objects)
  - Software external interface requirements
  - Software internal interface requirements
  - Software internal data requirements
  - Other software requirements (safety, security, privacy, environment, hardware, software, communications, quality, etc.)
  - Design and implementation constraints
- Qualification provisions to ensure each requirement has been met
  - Demonstration, test, analysis, inspection, etc.
- Requirements traceability
  - Trace back to the system or subsystem where each requirement applies

Software Requirements Specification (SRS) template

TABLE OF CONTENTS

- 1.0 Introduction
  - 1.1 Purpose
  - 1.2 Scope
  - 1.3 Definitions, Acronyms, and Abbreviations
  - 1.4 References
  - 1.5 Overview
- 2.0 General Description
  - 2.1 Product Perspective
  - 2.2 Product Functions
  - 2.3 User Characteristics
  - 2.4 General Constraints
  - 2.5 Assumptions and Dependencies

SRS template

- 3.0 Specific Requirements
    - 3.1 Functional Requirements
        - 3.1.1 Unit Registration
          3.1.2 Retrieving and Displaying Unit Information
          3.1.3 Report Generation
          3.1.4 Data Entry
    - 3.2 Design Constraints
    - 3.3 Non-Functional Requirements
        - 3.3.1 Security
- Appendix A

SRS explained

- **1.0 INTRODUCTION**

  This document specifies all the requirements for

  - **1.1 Purpose**

    The purpose of the …is to ….

    The system should assist ….

    The intended audience for this document is …

    This specification describes …..

  - **1.2 Scope**

    This document applies only to …...

    This specification is not concerned with …..

SRS explained

- **1.3 Definitions, Acronyms, and Abbreviations**

  SRS - Software Requirements Specifications

  IEEE - Institute of Electrical and Electronic Engineering
- **1.4 Reference**

  [1] IEEE 830-1993: IEEE Recommended Practice for Software Requirements Specifications" IEEE Standards Collection, IEEE, 1997.
- **1.5 Overview**

  In the following sections of this specification……will be presented.

  In Section 2, the general product and its functions will be introduced.

SRS explained

In Section 3, all detailed requirements will be specified and grouped.

In Appendix …….

2.0 GENERAL DESCRIPTION

2.1 Product Perspective

This system allows stakeholders to…..

The system will display…..

The system will help ……

The system provides information about ….

2.2 Product Functions

The system provides the following functions:

SRS explained

- **2.3 User Characteristics**

The users of the system are:

❿**Level of Users’ Computer Knowledge**

❿**Level of Users’ Business Knowledge**

❿**Frequency of Use**

- **2.4 General Constraints**

The system will support ….

The system will not allow ……

- **2.5 Assumption and Dependencies**

This system relies on ……

The system must have a satisfactory interface and ……

Section 3 of SRS

- **3 SPECIFIC REQUIREMENTS**
  - **3.1 Functional Requirements**
    - **3.1.1 Unit Registration**
      - The unit registration requirements are concerned with functions regarding unit registration which includes students selecting, adding, dropping, and changing a unit.
      - **SRS-001 (3.1.1.1):**
      - The system shall allow the user to register a unit.
      - **SRS-002 (3.1.1.2):**
      - STS shall allow the user to delete a unit if the user has chosen to drop that unit.
      - **SRS-003 (3.1.1.3):**
      - STS shall check if a unit has been filled by enough registered students.

SRS functional egs

- **SRS-004 (3.1.1.4):**
- STS shall allow the user to add his/her name to the unit waiting list if the user wants to register in a unit which has been filled already with enough registered students.
- **SRS-005 (3.1.1.5):**
- STS shall automatically register the unit for the user who is the first one on the waiting list if a vacancy appears for that unit.
- **SRS-006 (3.1.1.6):**
- STS shall allow the user to change practical session(s) within a unit.
- **SRS-007 (3.1.1.7):**
- STS shall allow the user to change tutorial session(s) within a unit.

Functional parent reqs broken into many child-reqs.

- **3.1.2 Retrieving and Displaying Unit Information**
- The retrieving and displaying requirements are concerned with how information is retrieved and presented to the user.
  - **SRS-014 (3.1.2.1):**
  - The system shall allow users to enter the following selection criteria to retrieve unit information: by unit code, by unit number, by title of unit, by weight of unit (credit points).
  - **OR** by unit code **(3.1.2.1.1)** , by unit number **(3.1.2.1.2)** , by title of unit **(3.1.2.1.3)** , by weight of unit (credit points) **(3.1.2.1.4).**

Design Constraints (3.2)

- **3.2 Design Constraints**
  - **SRS-031 (3.2.1):**
  - STS shall store and retrieve persistent data.
  - **SRS-032 (3.2.2):**
  - STS shall support PC and/or UNIX platforms.
  - **SRS-033 (3.2.3):**
  - STS shall be developed using the JAVA programming language

Non-functional requirements

- **3.3 Non-Functional Requirements**
  - **SRS-034 (3.3.1):**
  - STS shall respond to any retrieval in less than 5 seconds.
  - **SRS-035 (3.3.2):**
  - STS shall generate a report within 1 minute.
  - **SRS-036 (3.3.3):**
  - STS shall allow the user to remotely connect to the system.
  - **SRS-041 (3.3.8):**
  - The system will be accompanied by a comprehensive user manual.

Safety and security issues

- **3.5.3 Security**
- The security requirements are concerned with security and privacy issues.

SRS-029:

- VSS shall provide staff ID and password verification protection to protect from unauthorised use of the system.

SRS-030:

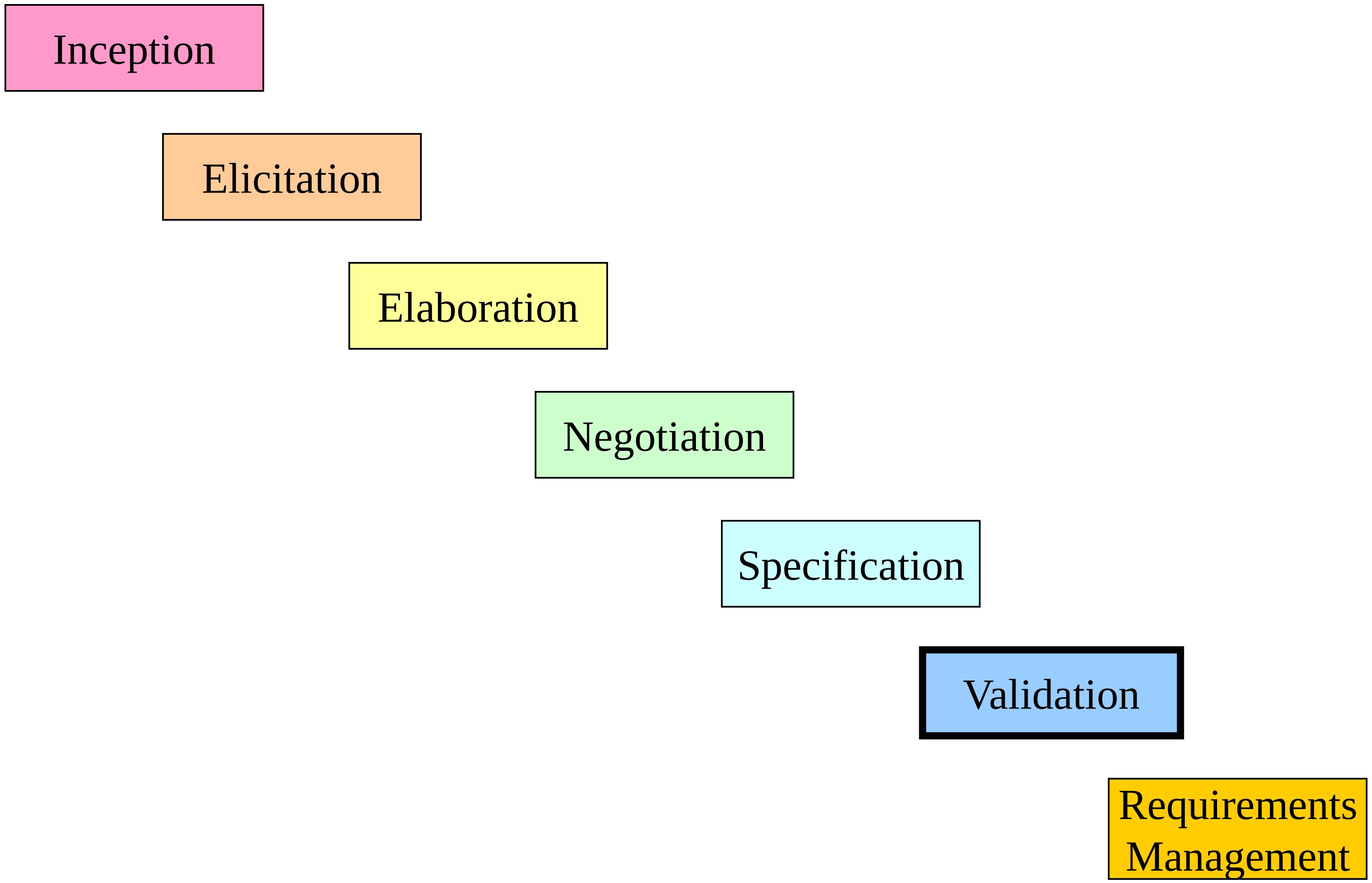
- VSS shall allow the store manager to add, remove and modify staff ID and passwords as required.

Other SRS template for section 3

- 3. Specific Requirements
    - 3.1 External Interface Requirements
        - 3.1.1 User Interfaces
        - 3.1.2 Hardware Interfaces
        - 3.1.3 Software Interfaces
        - 3.1.4 Communication Interfaces
    - 3.2 Functional Requirements
        - 3.2.1 Requirement 1
            - 3.2.1.1 Introduction
            - 3.2.1.2 Inputs
            - 3.2.1.3 Processing
            - 3.2.1.4 Outputs
        - 3.2.2 Requirement 2 …..

Other SRS template for section 3

- 3.3 Performance Requirements
- 3.4 Design Constraints
  - 3.4.1 Standards Compliance
  - 3.4.2 Hardware Limitations ……
- 3.5 Software System Attributes
  - 3.5.1 Reliability
  - 3.5.2 Availability
  - 3.5.3 Security
  - 3.5.4 Maintainability
  - 3.5.5 Portability
  - 3.5.6 Reusability
  - 3.5.7 Usability 3.5.8 Other Factors …..
- 3.6 Other Requirements
  - 3.6.1 Database
  - 3.6.2 Operations …..



Inception

Elicitation

Elaboration

Negotiation

Specification

Validation

Requirements Management

Validation Task

- During validation, the work products produced as a result of requirements engineering are assessed for quality

- The specification is examined to ensure that
  - all software requirements have been stated unambiguously
  - inconsistencies, omissions, and errors have been detected and corrected
  - the work products conform to the standards established for the process, the project, and the product

- The formal technical review serves as the primary requirements validation mechanism
  - Members include **software engineers**, **customers**, **users**, and **other stakeholders**

Questions to ask when Validating Requirements

- Is each requirement consistent with the overall objective for the system/product?

- Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?

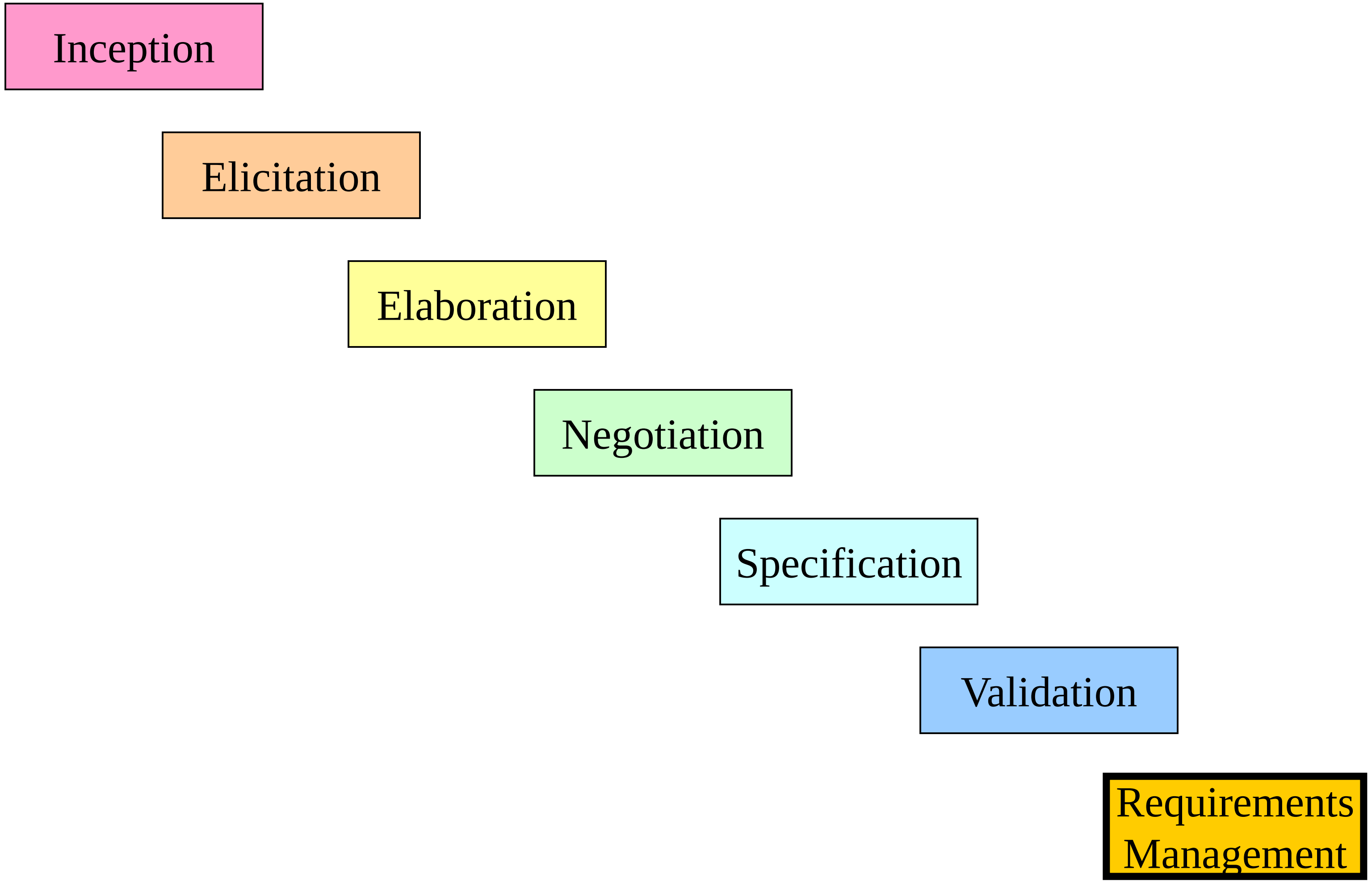
- Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?
- Is each requirement bounded and unambiguous?

- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?

(more on next slide)

Questions to ask when Validating Requirements (continued)

- Do any requirements **conflict** with other requirements?
- Is each requirement **achievable** in the **technical environment** that will house the system or product?
- Is each requirement **testable**, once implemented?
  - Approaches: Demonstration, actual test, analysis, or inspection
- Does the requirements **model properly reflect the information**, **function**, and **behavior** of the **system to be built?**
- Has the requirements model been "**partitioned**" in a way that exposes progressively more ***detailed information*** about the system?



Inception

Elicitation

Elaboration

Negotiation

Specification

Validation

Requirements Management

Requirements Management Task

- During requirements management, the project team performs a set of activities to identify, control, and track requirements and changes to the requirements at any time as the project proceeds

- Each requirement is assigned a unique identifier

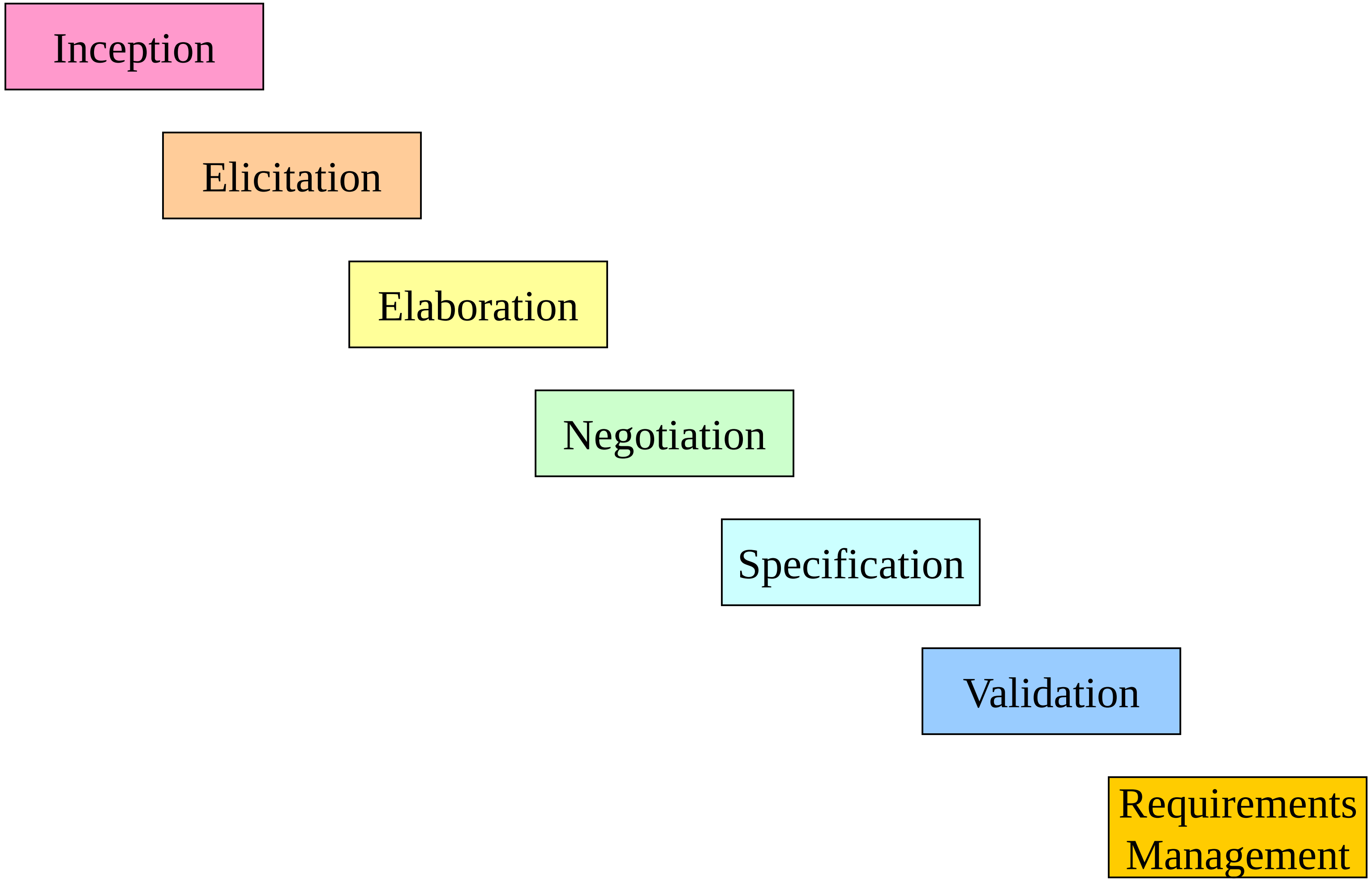
- The requirements are then placed into one or more traceability tables

Requirements Management Task

- These tables may be stored in a database that relate features, sources, dependencies, subsystems, and interfaces to the requirements

- A requirements traceability table is also placed at the end of the software requirements specification

Summary



Inception

Elicitation

Elaboration

Negotiation

Specification

Validation

Requirements Management

☺

Any Question?