Introduction of Software Engineering SOF107

CHAPTER 3: REQUIREMENT ENGINEERING

Dr. Hejab M. Al-Fawareh

alfawarehhejab.Khaled@xmu.edu.my

0199184370

University mission and vision

- **Vision**
- Xiamen University Malaysia aspires to become a university with a distinct global outlook, featuring first-class teaching and research, and embracing cultural diversity.
- **Mission**
- To nurture young talents with dignity and wisdom, turning them into fine citizens of the region who will contribute to the prosperity of the people and social progress of Malaysia, China and Southeast Asia.

Learning Outcome

- **Apply and demonstrate the technique and methodology for software requirement.**
  -

At the end of these lectures students are able to:

1. Introduce the concepts of user and system requirements

2. Described functional / non-functional requirements

3. Explain techniques for describing system requirements

Content

- REQUIREMENT ENGINEERING
- User Requirement & System Requirement
- Requirement Functional Requirement& Non- Functional Requirement
- Domain Requirement
- REQUIREMENTS ENGINEERING PROCESS SOFTWARE
- SOFTWARE REQUIREMENTS DOCUMENTATION/SPECI FICATION.

REQUIREMENT ENGINEERING

- **Requirements engineering** is the process of establishing
  - the services that the customer requires from a system
  - the constraints under which it operates and is developed

- 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

Requirements engineering

- How we should specify "exactly" what is expected before we start designing something
- Requirements describe the necessary functions and features of the system we are to conceive, design, implement and operate.
  - ❑ Performance
  - ❑ Schedule
  - ❑ Cost
  - ❑ Other Characteristics (e.g. lifecycle properties)

Requirements engineering

- Requirements are often organized hierarchically
  - ❑At a high level requirements focus on what should be achieved, not how to achieve it
  - ❑Requirements are specified at every level, from the overall system to each hardware and software component.

❖ *The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as: A condition or capability needed by a user to solve a problem or achieve an objective.*

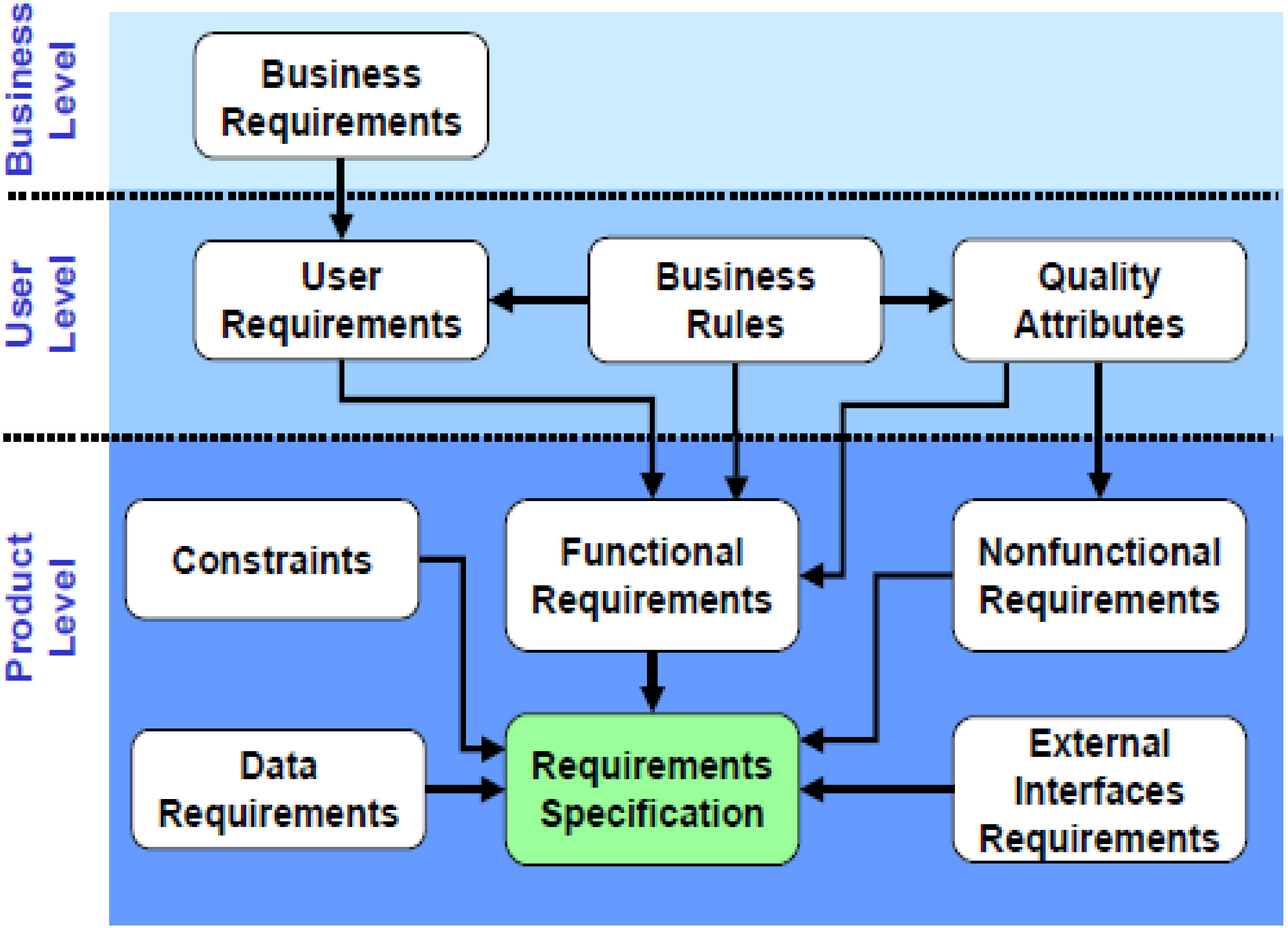
WHAT IS A REQUIREMENT?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is inevitable as 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.

Types of requirements

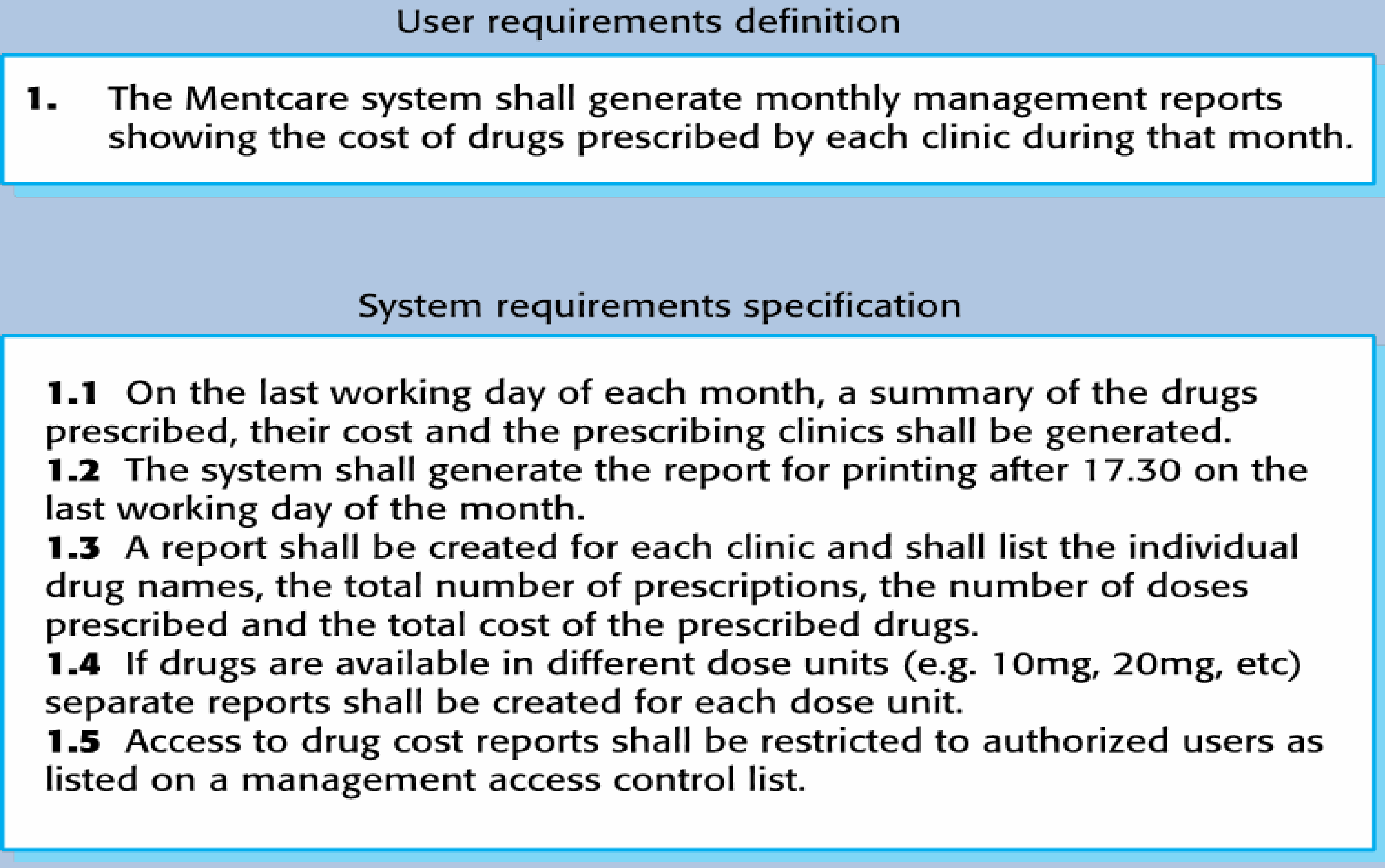
- Business requirements
- User requirements
- Software requirements

Business requirements	User requirements	Software requirements
Outline measurable goals for the business.	Reflect specific user needs or expectations.	Identify features, functions, non-functional requirements and use cases.
Define the *why* behind a software project.	Describe the *who* of a software project.	Delve into the *how* of a software project.
Match project goals to stakeholder goals.	Highlight how users interact with it.	Describe software as functional modules and non-functional attributes.
Maintain a BRD with requirements, updates or changes.	Create a URS, or make them part of the BRD.	Compose an SRS, and, optionally, an FRS.


Business Level
User Level
Product Level
Business Requirements
User Requirements
Business Rules
Quality Attributes
Constraints
Functional Requirements
Nonfunctional Requirements
Data Requirements
Requirements Specification
External Interfaces Requirements

- User requirement
  - We need to be able to spell check documents
- System requirement
  - The system needs to be able to spell check documents and provide autocorrect facilities. Their will be support for the following languages, English, French and German will plug in support for other languages
- Software specification
  - CheckResult spellCheck(String word, Dictionary dictionary)
    - Word is defined in UNICODE formatted string
    - The Dictionary structure is defined in S.1.2
    - The CheckResult is defined in S.1.3 and contains a flag if the word has been found or not, plus a Vector object containing a list of possible other word suggestions depending if the word has been found or not
    - spellCheck will ideally use Hashing tables to improve code efficiency
    - ......

types of requirements



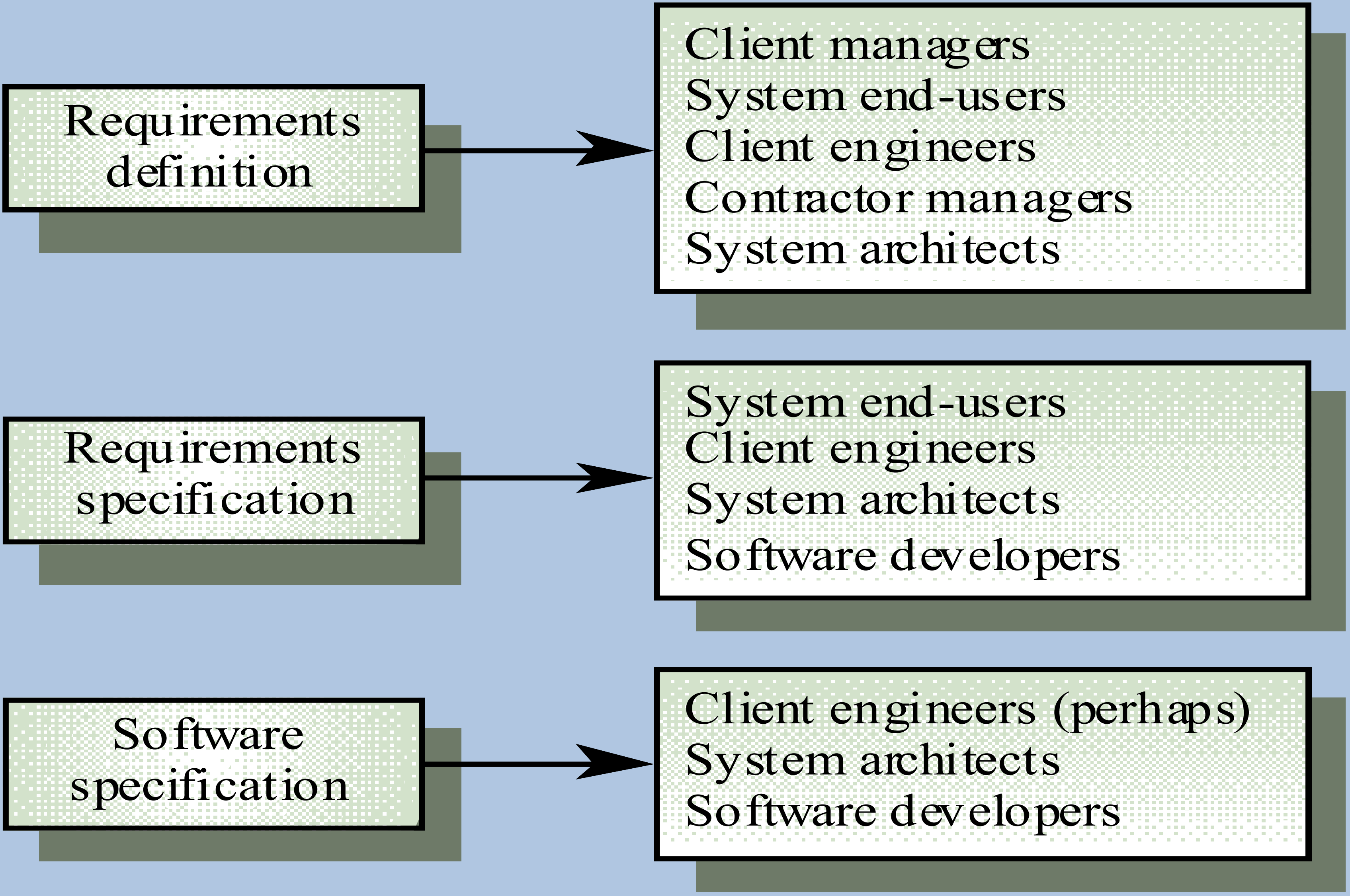
User requirements definition

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.

Types of Readers



Functional and non-functional requirements

- Software system requirements are often classified as functional requirements or nonfunctional requirements :
- **Functional requirement:** 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.

Functional and non-functional requirements

- **Non-functional requirements:** These are constraints on the services or functions offered by the system.

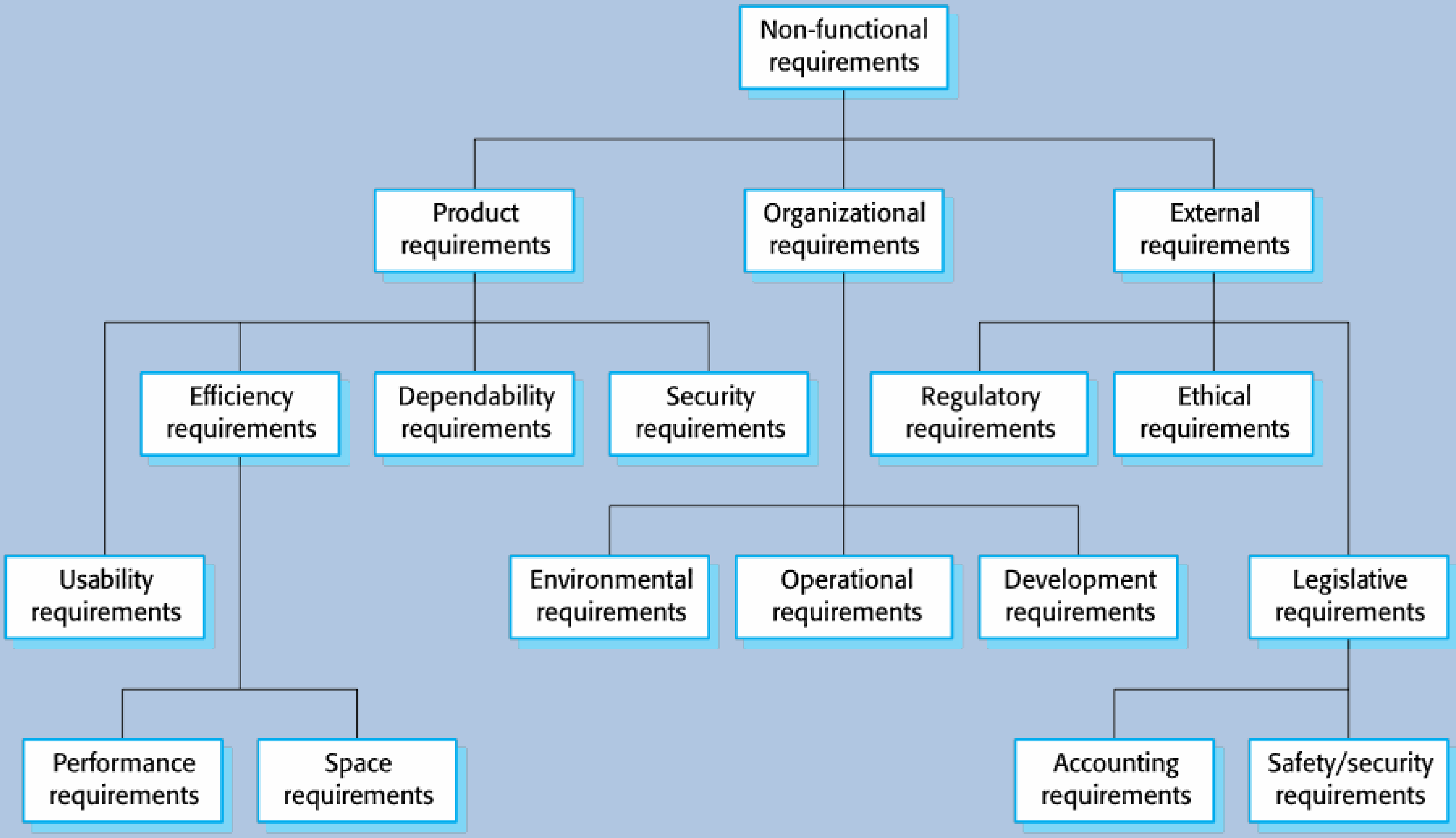
Implementation of non-functional requirements

- Non-functional requirements may affect the overall architecture of a system rather than the individual component

Classification of Non-functional requirements

- 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, legislative requirements, etc.

Classification of Non-functional requirements



Domain Requirement

- The system’s operational domain imposes requirements on the system.
- Domain requirements can be new functional requirements, constraints on existing
  requirements, or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable

Domain Requirements Problems

- Understandability :

➢ Requirements are expressed in the language of the application domain This is often not understood by software engineers developing the system.

- Implicitness :

➢ Domain specialists understand the area so well that they do not think of making the domain requirements explicit which leads to problems later if software developer implements the requirements in the wrong way

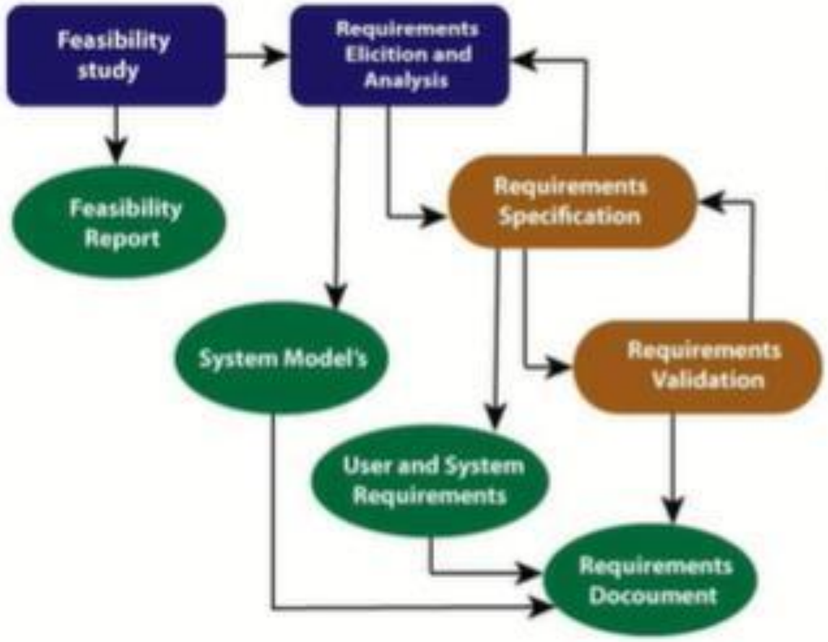
Requirements engineering processes

Requirements engineering processes

- The processes used for requirements engineering vary widely depending on the application domain, the people involved and the organization developing the requirements.
- However, there are a number of generic activities common to all processes which we will look at it .
- The goal of this stage of the software engineering process is to help create and maintain a system/software requirements document.

Requirements Engineering Processes

The goal - create and maintain a system requirements document.



Requirement Engineering Process

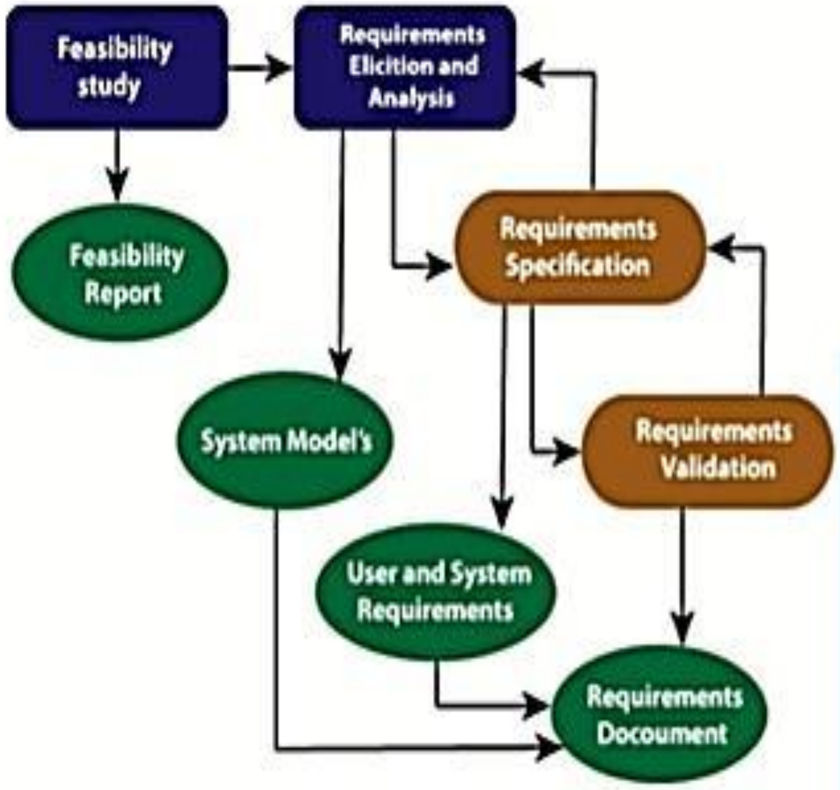
1. Requirements elicitation;
2. Requirements analysis;
3. Requirements validation;
4. Requirements management.

Requirements Engineering Processes

1. Requirements elicitation;
What services do the end-users require of the system?
2. Requirements analysis;
How do we classify, priorities and negotiate requirements?
3. Requirements validation;
Does the proposed system do what the users require?
4. Requirements management.
How do we manage the (sometimes inevitable) changes to the requirements document?

Requirements Engineering Processes

The goal - create and maintain a system requirements document.



Requirement Engineering Process

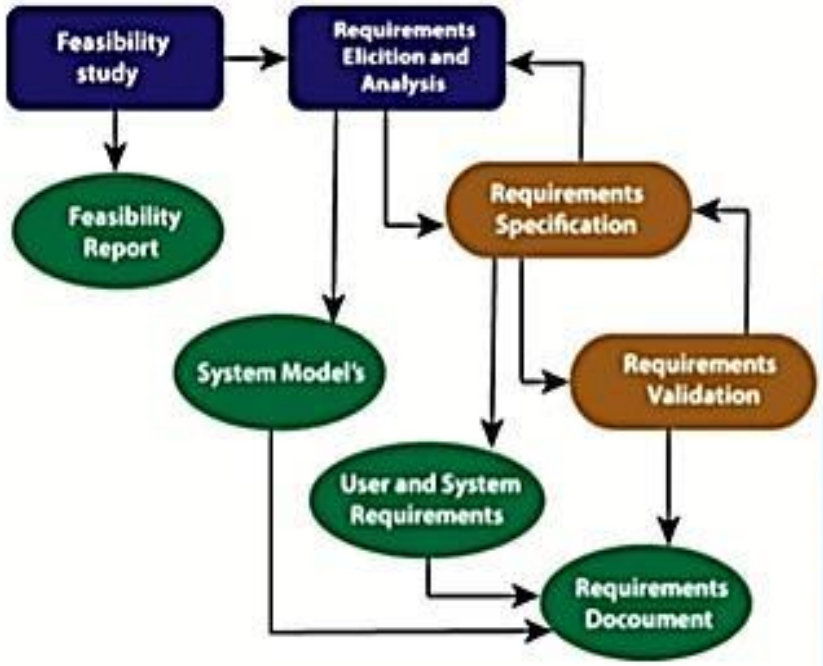
1. Requirements elicitation;
2. Requirements analysis;
3. Requirements validation;
4. Requirements management.

Example: Patient records system [Elicitation]

1. Talk to patients, doctors, nurses, receptionists, managers to find out current system practise, legal restrictions DPA, problems with current system, needs for improvement, security issues, costs.

Requirements Engineering Processes

The goal - create and maintain a system requirements document.



Requirement Engineering Process

1. Requirements elicitation;
2. Requirements analysis;
3. Requirements validation;
4. Requirements management.

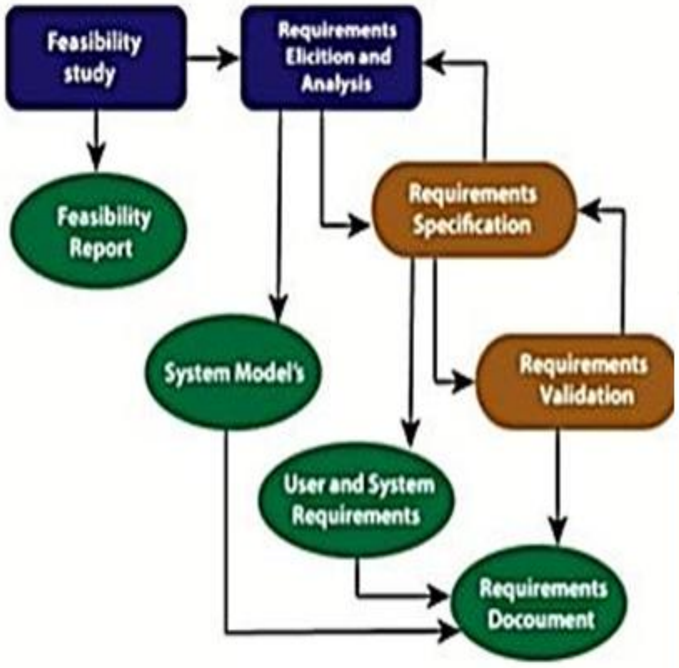
Example: Patient records system

[Analysis]

Develop draft documentation and review what is most important, what will it cost, what is the timescale, is new hardware required.

Requirements Engineering Processes

The goal - create and maintain a system requirements document.



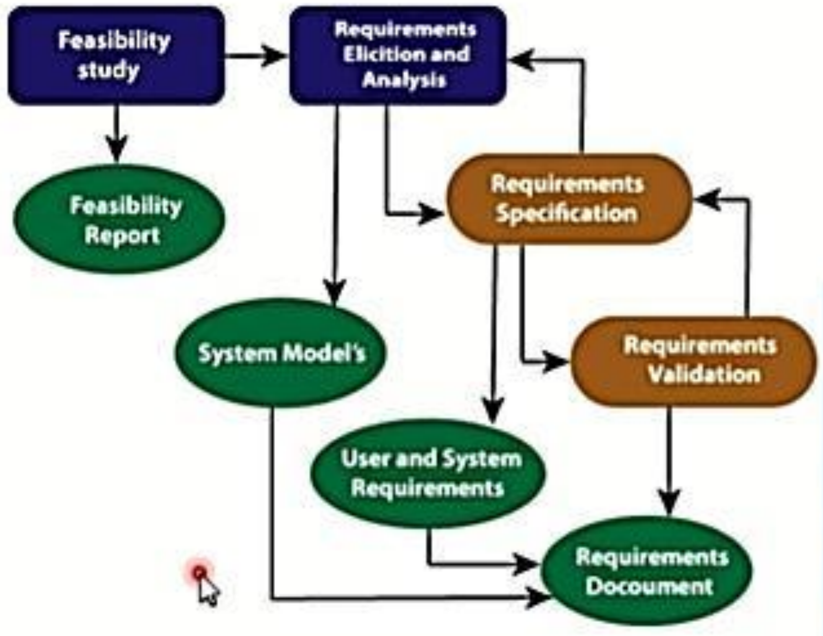
Requirement Engineering Process

1. Requirements elicitation;
2. Requirements analysis;
3. Requirements validation;
4. Requirements management.

Example: patient records system (Validation)
Send requirements to end users. Present them with Q&A. Go back to step 1, discuss requirements again

Requirements Engineering Processes

The goal - create and maintain a system requirements document.



Requirement Engineering Process

1. Requirements elicitation;
2. Requirements analysis;
3. Requirements validation;
4. Requirements management.

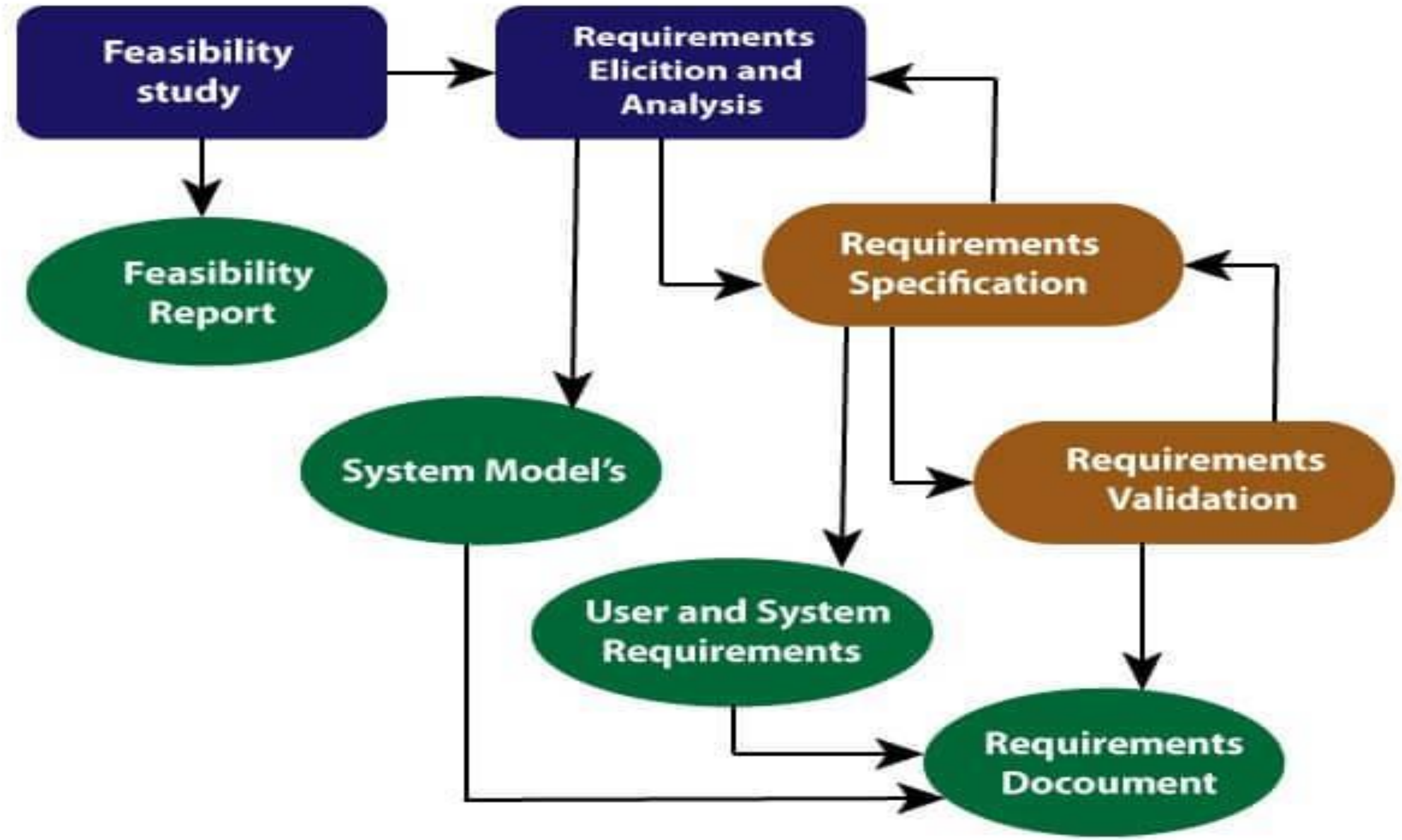
Example: Patient records system

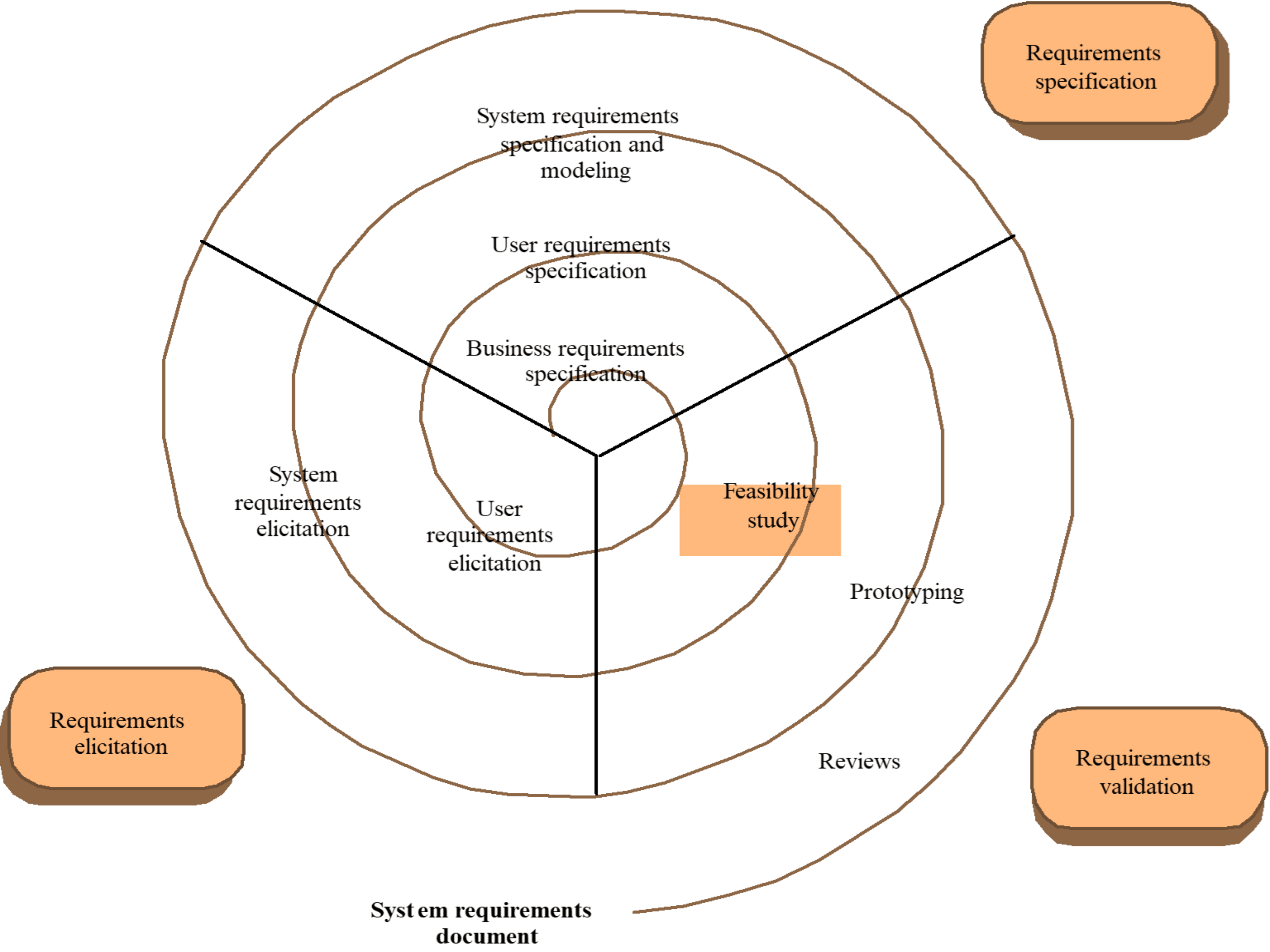
[Management]
Have a yearly review of requirements between all stakeholders.
Have a system of reviewing the cost and feasibility of change to system

Requirements engineering processes

- Requirements engineering processes may include **four high-level activities**.
- **feasibility study**
- **elicitation and analysis**
- **Specification**
- **validation**

Requirements Engineering Process




Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
System requirements elicitation
User requirements elicitation
Feasibility study
Prototyping
Reviews
Requirements elicitation
Requirements validation
Syst em requirements document

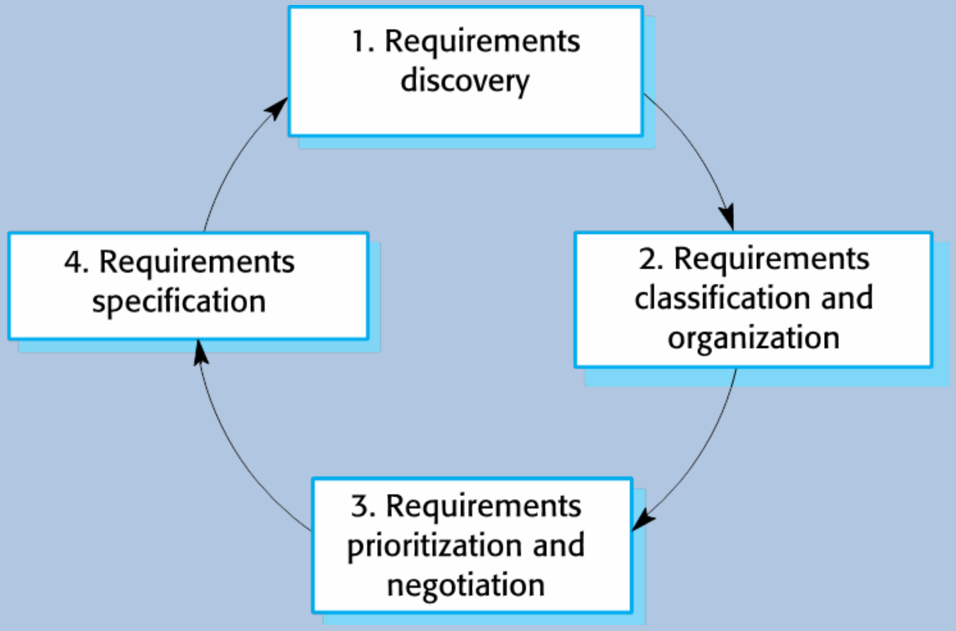
Requirements engineering processes

- **Feasibility Study**
- A feasibility study is a short, focused study that should take place early in the RE process. It should answer three key questions:
  - does the system contribute to the overall objectives of the organization?
  - can the system be implemented within schedule and budget using current technology?
  - can the system be integrated with other systems that are used?

Requirements elicitation and analysis

- After an initial feasibility study, the next stage of the requirements engineering process is requirements elicitation and analysis.
- In this activity, software engineers work with customers and system end-users to find out about the application domain, what services the system should provide, the required performance of the system, hardware constraints, and so on.
- Stakeholders (end-users, managers, engineers) also may involve in maintenance, domain experts, trade unions, etc.

Requirements engineering processes


1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirements specification

The process activities are

- **Requirements discovery.** This is the process of interacting with stakeholders of the system to discover their requirements.
- **Requirements classification and organization.** This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters.
- **Requirements prioritization and negotiation.** This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation.
- **Requirements specification**. The requirements are documented and input into the next round of the spiral.

Requirements Validation

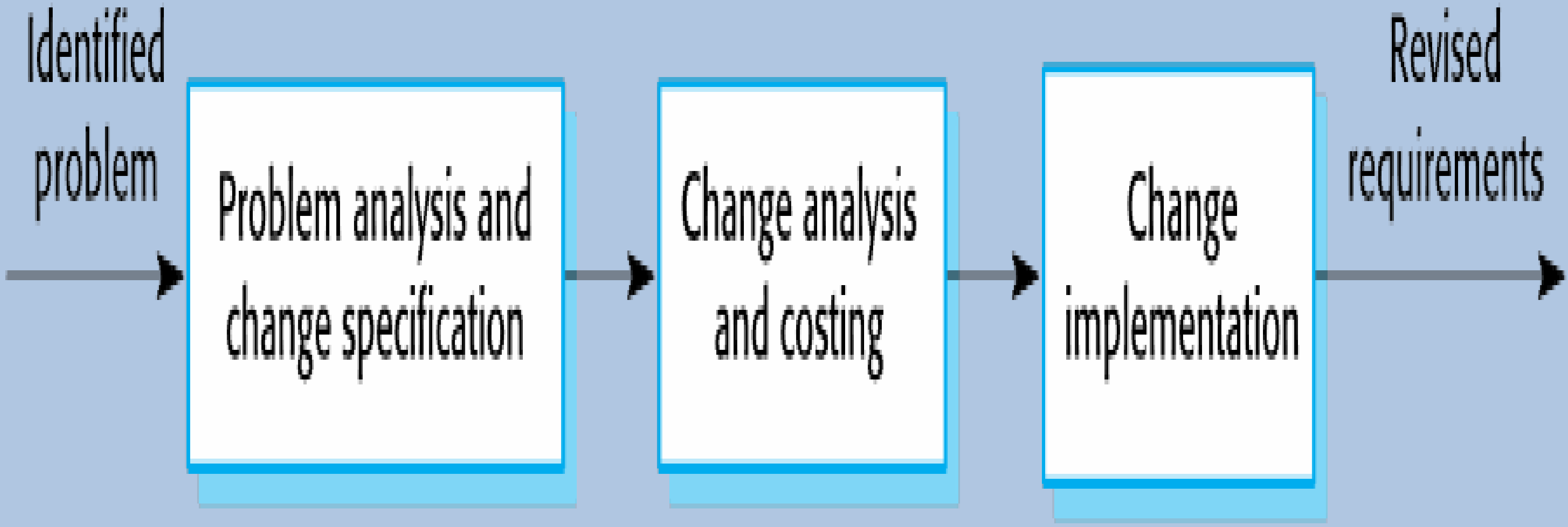
- Requirements validation is the process of checking a software system meets specification, what the customer needs.
- During the requirements validation process, different types of checks should be Carried out. These checks include:
- **Validity checks:** A user may think that a system is needed to perform certain functions.
- **Consistency checks**: Requirements in the document should not conflict.
- **Completeness checks**: The requirements document should include requirements that define all functions and the constraints intended by the system user.
- **Realism checks**: Using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented.
- **Verifiability:** To reduce the potent There are a number of requirements validation techniques that can be used individually or in conal for dispute between customer and contractor, system requirements should always be written so that they are verifiable.

Requirements validation techniques:

- **Requirements reviews**: The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies
- **Prototyping**: An executable model of the system in question is demonstrated to end-users and customers.
- **Test-case generation**: Requirements should be testable. If the tests for the requirements are devised as part of the validation process, this often reveals requirements problems.

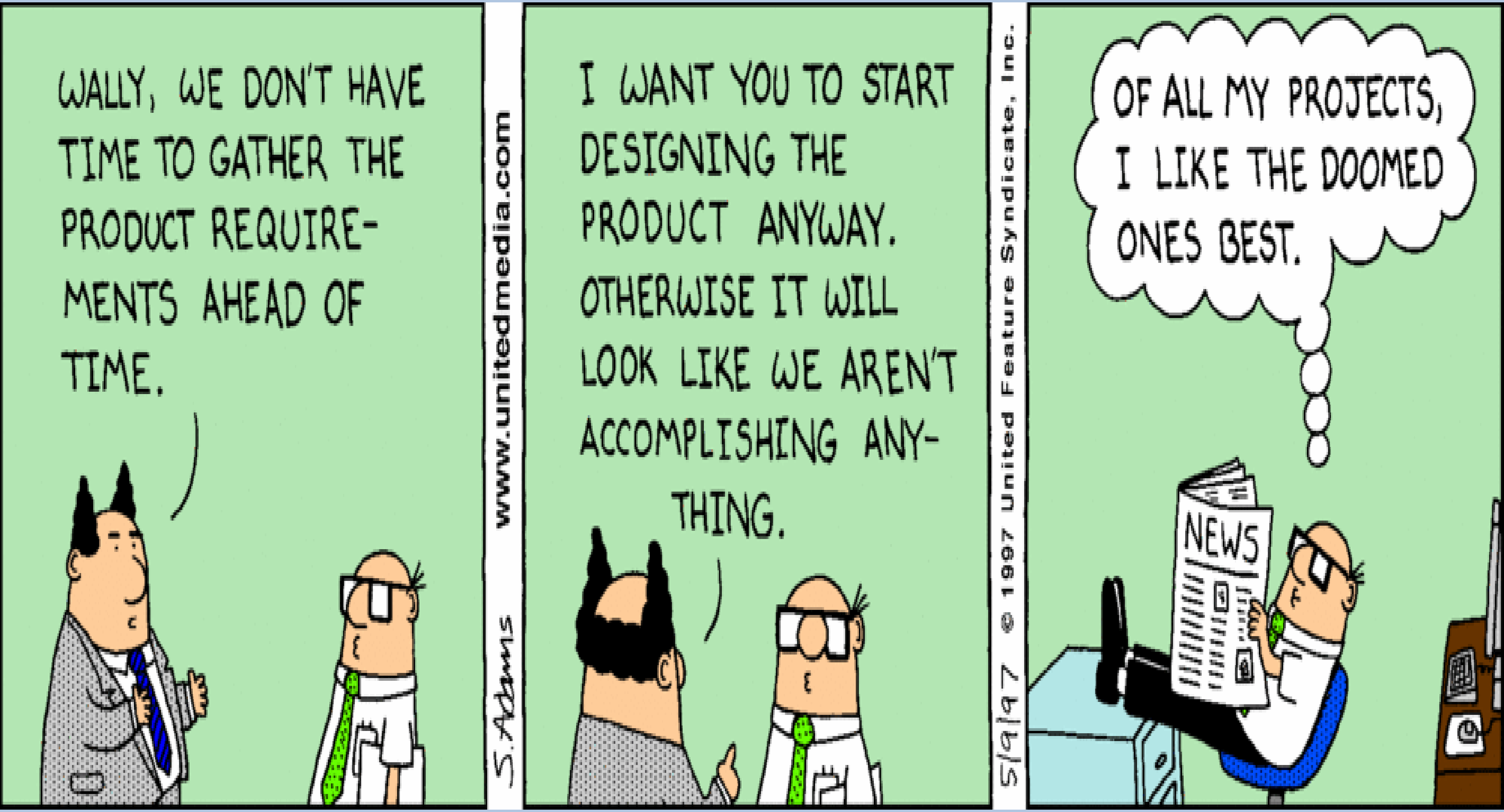
Requirements Management

- requirements management is the process of understanding and controlling changes to system requirements. Planning is an essential first stage in the requirements management process.
- planning stage establishes the level of requirements management detail that is required.


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements

SOFTWARE REQUIREMENTS DOCUMENTATION

SOFTWARE REQUIREMENTS DOCUMENT/SPECIFICATION


WALLY, WE DON'T HAVE TIME TO GATHER THE PRODUCT REQUIREMENTS AHEAD OF TIME.
I WANT YOU TO START DESIGNING THE PRODUCT ANYWAY. OTHERWISE IT WILL LOOK LIKE WE AREN'T ACCOMPLISHING ANYTHING.
OF ALL MY PROJECTS, I LIKE THE DOOMED ONES BEST.
NEWS
www.unitedmedia.com
S.Adams
5/9/97 © 1997 United Feature Syndicate, Inc.

Need for SRS

- Helps user understand his needs.
- SRS provides a reference for validation of the final product
- High quality SRS essential for high Quality SW
- Good SRS reduces the development cost

Structure of SRS

- Preface (including change history)
- Introduction
- Contents
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System models
- System evolution
- Appendices
- Index

Some key characteristics of a good SRS

- Some key ones are Complete
- Unambiguous
- Consistent
- Verifiable
- Ranked for importance and/or stability

Software Requirement Specification

- The process of writing down the user and system requirements in a requirements document.
- User requirements have to be understandable by end users and customers who do not have a technical background.
- System requirements are more detailed requirements and may include more technical information.
- The requirements may be part of a contract for the system development .
- It is therefore important that these are as complete as possible

Requirements Document/Specification (SRS)

- The software requirements document is the **official statement** of what is required of the system developers Should include both a definition and a specification of requirements It is NOT a design document.
  Set of WHAT the system should do rather than HOW it should do it

Requirements Document/Specification (SRS)

- **Specify external system behaviour** (what does it do?).
- **Specify implementation constraints** (what system it must run on, what programming language it must use).
- **Easy to change**

  Serve as reference tool for maintenance .
- **Record forethought about the life cycle of the system** i.e. predict changes (how can it be expanded for more users).
- **Characterize responses to unexpected events** (e.g. w hat should it do if power is lost!)

Requirements Document/Specification (SRS)

- The level of detail
  depends on both the type of system and the development process being used.
  In **evolutionary development** model the requirements may change many times.
  In the **waterfall model,** it should be more complete before design stage.
  If the (sub)-system developed by an external contractor or it is a critical system, more time needs to be taken on finalizing the requirements document

Software Requirement Specification

- A SRS must specify requirements on Functionality Performance Design constraints External interfaces.

Ways of writing a system requirements specification:

- Natural language: The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
- Structured natural language : The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
- Design description languages : This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications

Software Requirement Specification

- Graphical notations: Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
- Mathematical specifications : These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don’t understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract.

An SRS must specify requirements on Functionality Performance Design constraints External interfaces

- **Natural languages** mostly used, with some structure for the document Formal languages are precise and unambiguous but hard Formal languages used for special features or in highly critical systems
  Problem 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 in same document *Requirements amalgamation* Several different requirements may be expressed together Leads to problems with testing/debugging

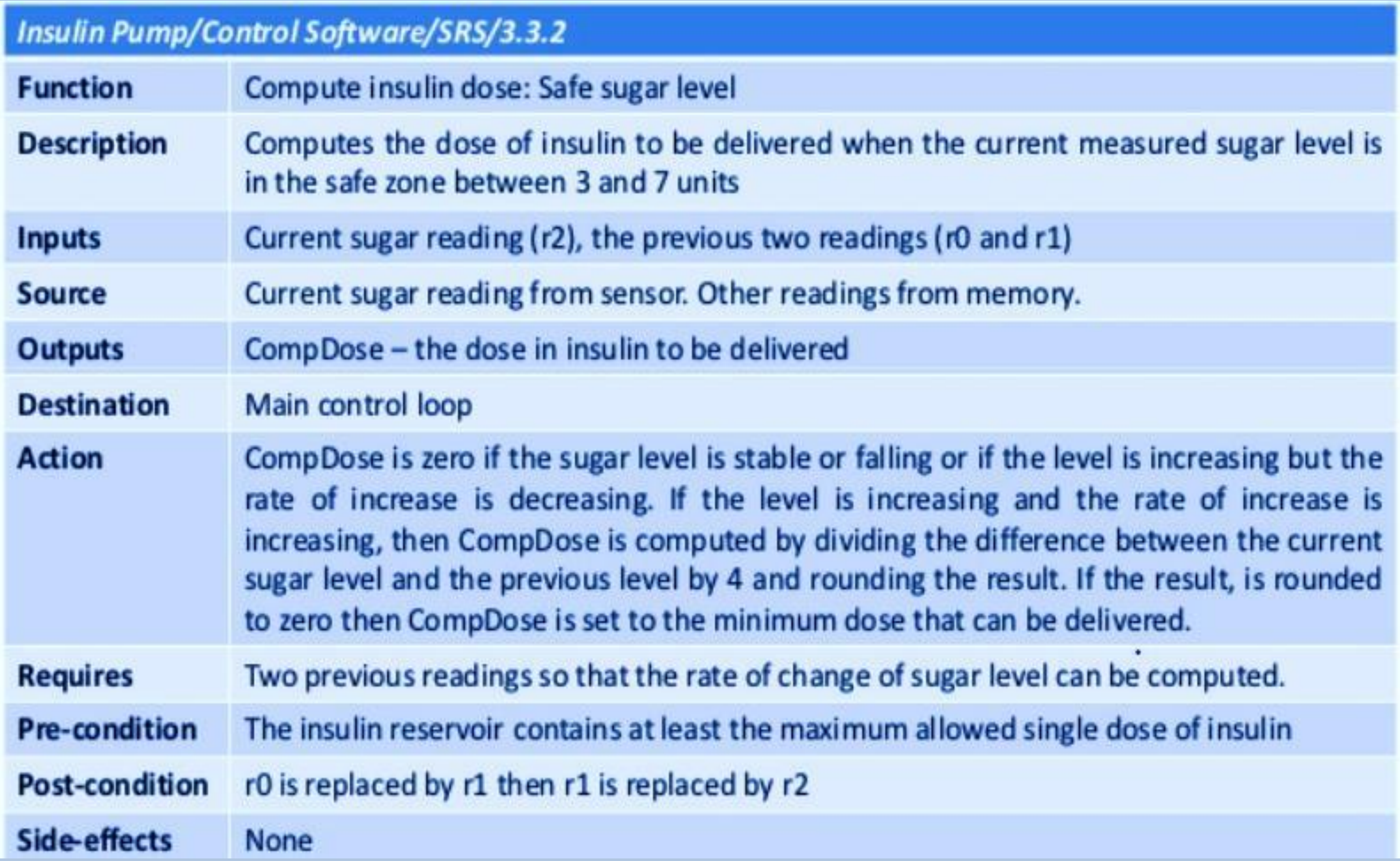
Structured specifications

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

Form-based specifications

- ✓ Definition of the function or entity.
- ✓ Description of inputs and where they come from.
- ✓ Description of outputs and where they go to.
- ✓ Information about the information needed for the computation and other entities used.
- ✓ Description of the action to be taken.
- ✓ Pre and post conditions (if appropriate).
- ✓ The side effects (if any) of the function.

Form-based specifications



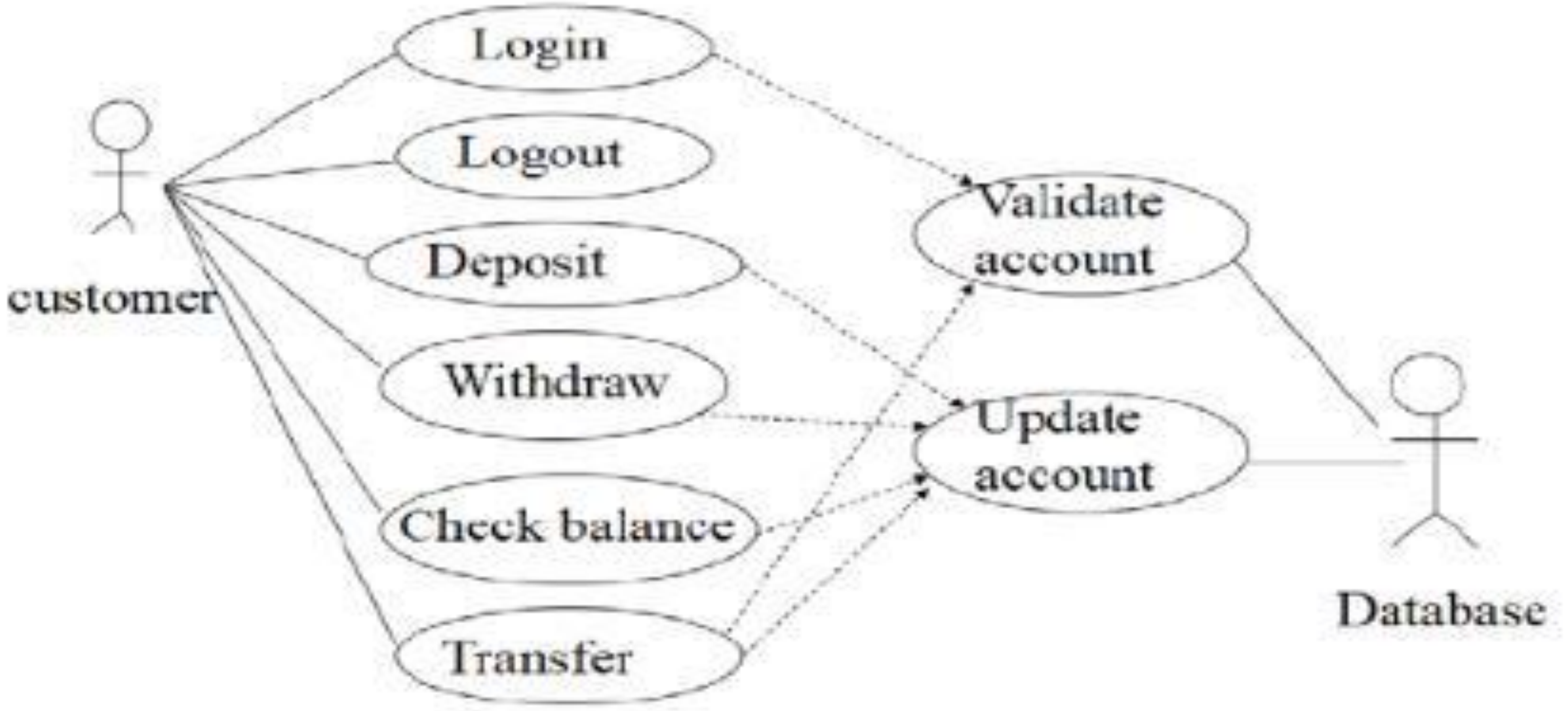
Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: Safe sugar level
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1)
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose – the dose in insulin to be delivered
Destination	Main control loop
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin
Post-condition	r0 is replaced by r1 then r1 is replaced by r2
Side-effects	None

Use case

- DEFINITION. Use case – A description of the interactions possible between actors and a system that, when executed, provide added value.
  Use cases specify a system from a user's (or other external actor's) perspective: every use case describes some functionality that the system must provide for the actors involved in the use case.
  - Use case diagrams provide an overview
  - Use case descriptions provide the details

[Jacobson et al. 1992 Glinz 2013]

Use Case Diagram



All dependency relationships are of type <<include>>

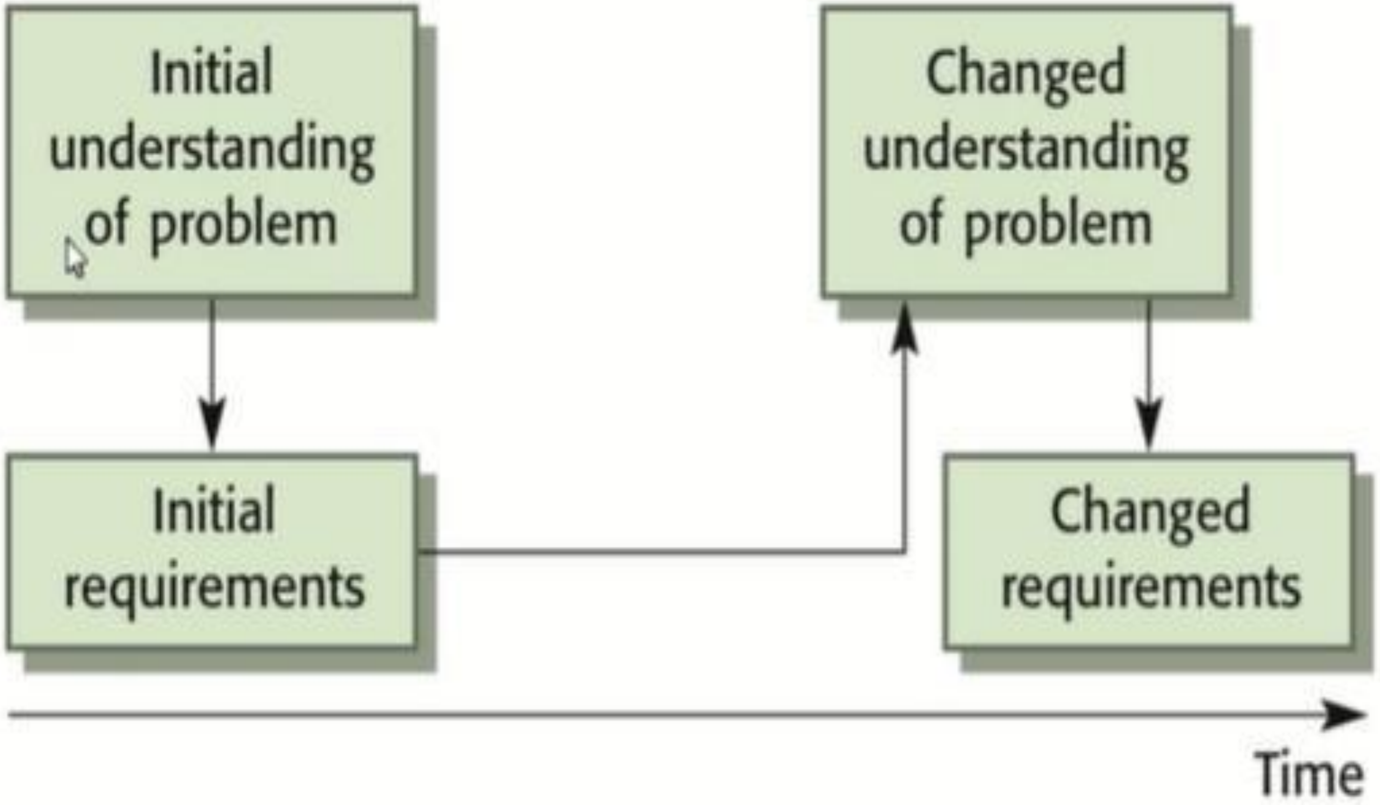
Requirements management

- Requirements management
  process of managing changing requirements during the requirements engineering process and system development.
  - ❑Emerging New requirements
  - ❑Keep track of individual requirements
  - ❑Maintain links between dependent requirements to assess the impact of requirements changes.
  - ❑A formal process needed for making change proposals and linking these to system requirements

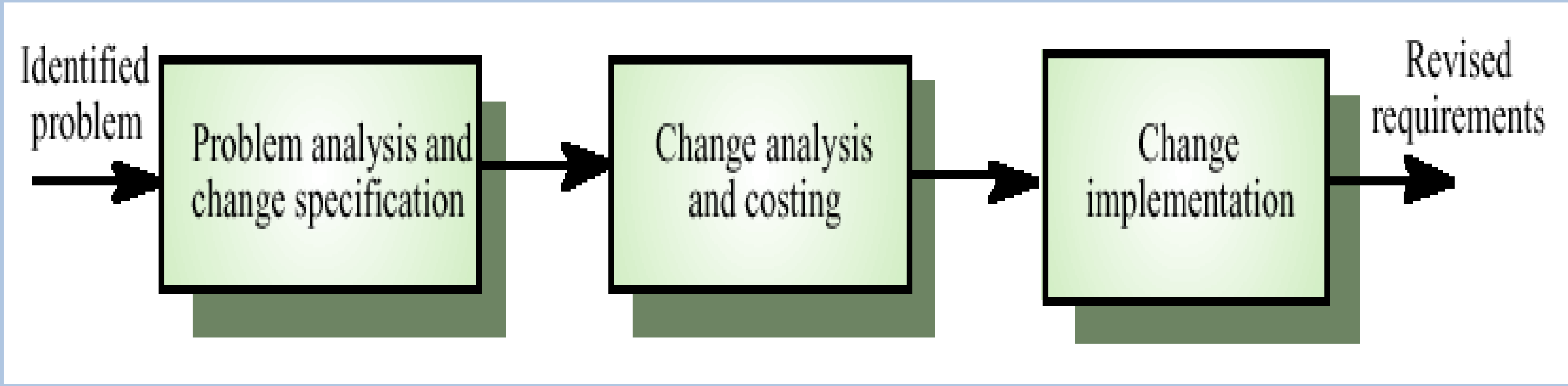
Requirements management

- Changing requirements
  - ❑ The business and technical environment of the system always changes after installation.
  - ❑ The people who pay for a system and the users of that system are rarely the same people
  - ❑ Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.

Requirements evolution


Initial understanding of problem
Initial requirements
Changed understanding of problem
Changed requirements
Time

Requirements change management


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements

Requirements change management

- During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request

The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.

The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented