Requirements Engineering

Chapter 4 – Ian Sommerville

Lecture Notes Prepared by Asst. Prof. Melody Angelique C. Rivera
Faculty, College of Computer Studies, Silliman University

Requirements

- The descriptions of what the system should do —**the services that it provides and the constraints on its operation**
- These requirements 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 (RE)

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

User Requirements

- **high-level abstract requirements**
- 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
- Examples (Functional User Requirements):

  FUR1: The user shall be able to enter his or her credentials to log into the system.

  FUR2: …

  …

  FUR*n*: The user shall be able to generate monthly summary and detailed sales reports.

System Requirements

- **more detailed descriptions of the software system's functions, services, and operational constraints** (what the system should do)
- 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
- Examples:

  FSR1: The system shall accept a user's credentials and check their validity before logging into the system.

  FSR2:…

  …

  FSR*n*: The system shall allow the user to generate monthly summary and detailed sales reports.

User and System Requirements

User requirements definition

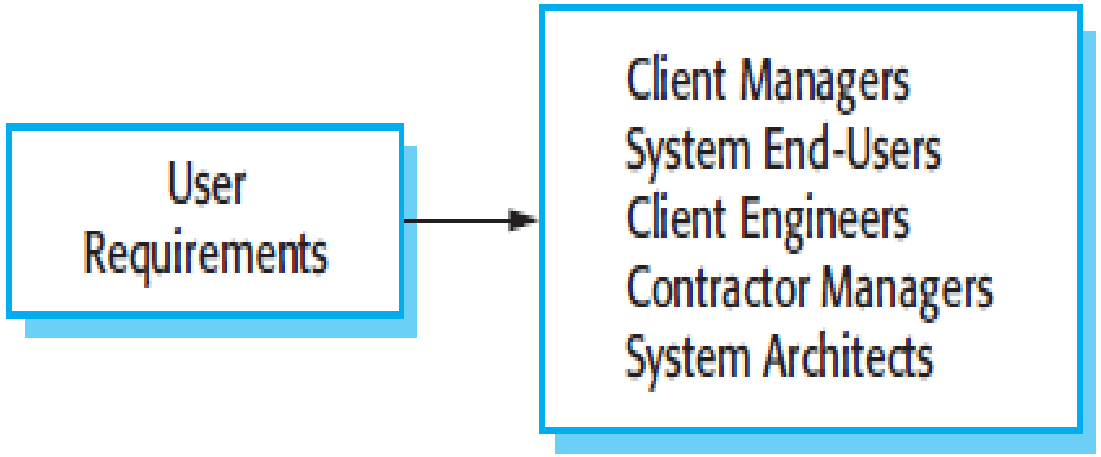
1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

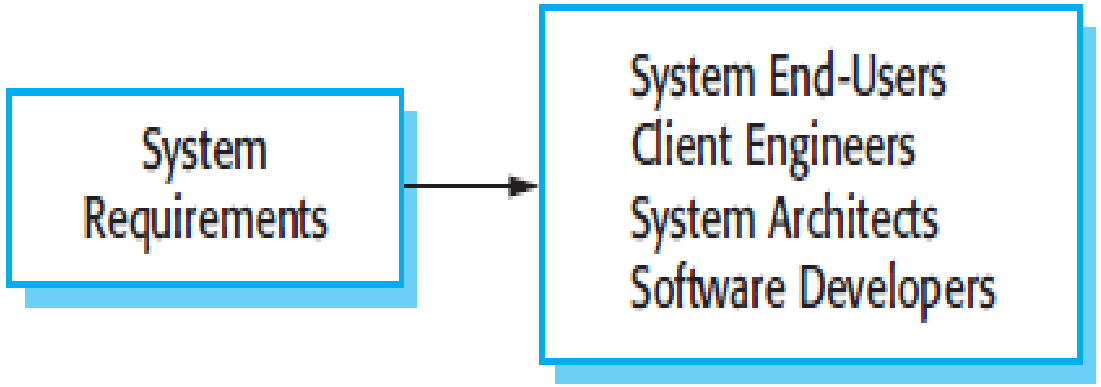
1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list 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.

Readers of Different Types of Requirements Specification

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



System Stakeholders

- Any person or organization who is affected by the system in some way and so who has a legitimate interest
- Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders

Stakeholders in the Mentcare System

- **Patients** whose information is recorded in the system.
- **Doctors** who are responsible for assessing and treating patients.
- **Nurses** who coordinate the consultations with doctors and administer some treatments.
- **Medical receptionists** who manage patients' appointments.
- **IT staff** who are responsible for installing and maintaining the system.

Functional Requirements

- statements of services that the system should provide
  - how the system should react to particular inputs
  - how the system should behave in particular situations
  - descriptions of how some computations must be carried out
- In some cases, the functional requirements may also explicitly state what the system should *not* do
- The functional requirements for a system describe what the system should do

Functional Requirements (cont.)

- These requirements depend on the type of software being developed, the expected users of the software, and the general approach taken by the organization when writing requirements
- Functional **user** requirements are usually described in an abstract way that can be understood by system users
  - More specific functional system requirements describe the system functions, its inputs and outputs, exceptions, etc., in detail
- Functional **system** requirements vary from general requirements covering what the system should do to very specific requirements reflecting local ways of working or an organization's existing systems

Examples

- User Functional Requirements

  UR1: A user shall be able to search the appointments lists for all clinics.

- System Functional Requirements

  SR1: The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.

Functional Requirements (cont.)

- In principle, the functional requirements specification of a system should be both **complete** and **consistent**
- ***Completeness*** means that all services required by the user should be defined
- ***Consistency*** means that requirements should not have contradictory definitions

Non-functional Requirements

- Requirements that are not directly concerned with the specific services delivered by the system to its users
- 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**

Non-functional Requirements (cont.)

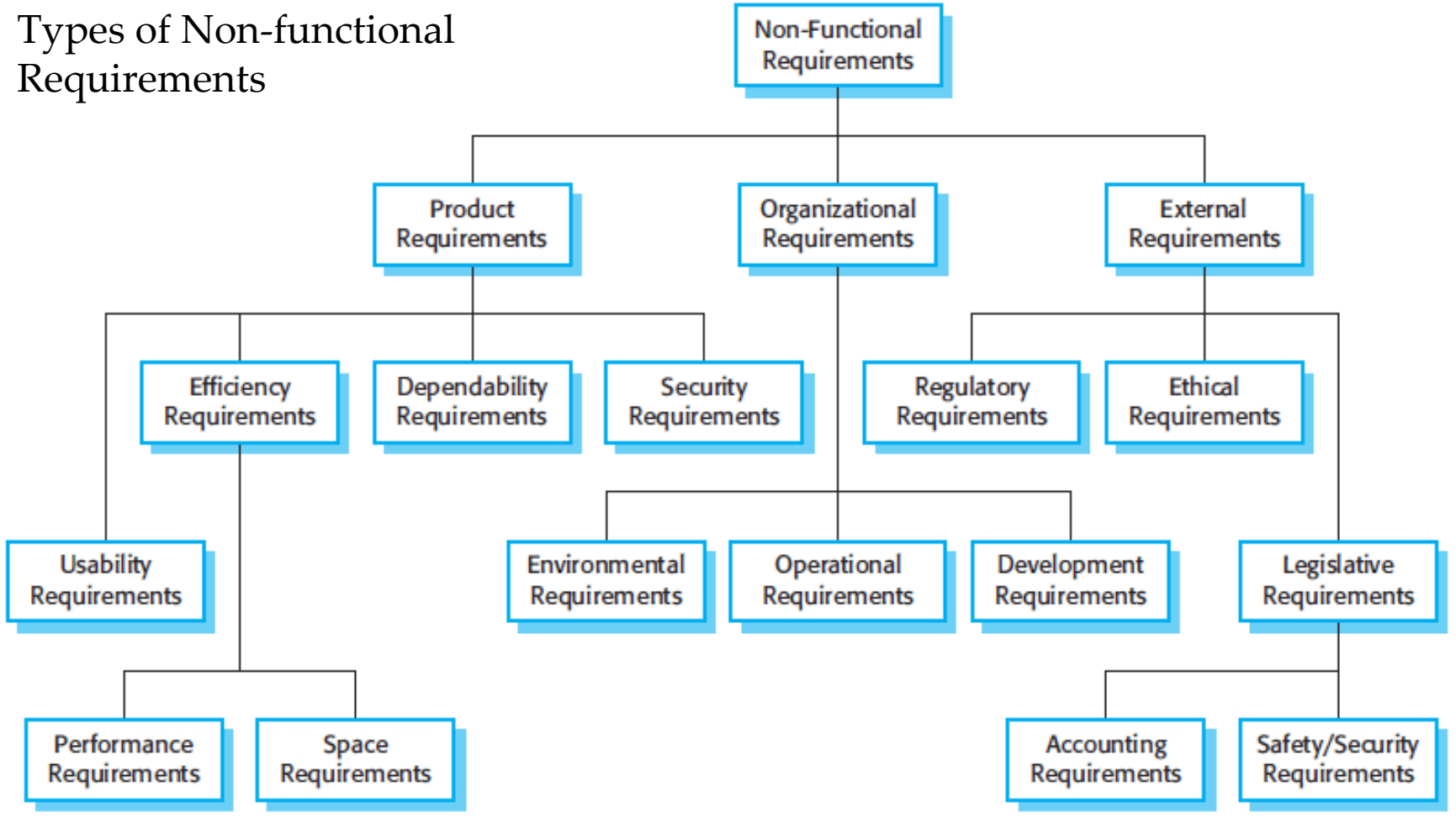
- They may relate to emergent system properties such as reliability, response time, and store occupancy
- Alternatively, they may define constraints on the system implementation such as the capabilities of I/O devices or the data representations used in interfaces with other systems
- Non-functional requirements, such as performance, security, or availability, usually specify or constrain characteristics of the system as a whole

Non-functional Requirements (cont.)

- Non-functional requirements are often more critical than individual functional requirements
- System users can usually find ways to work around a system function that doesn't really meet their needs
- **Failing to meet a non-functional requirement can mean that the whole system is unusable**
  - if an aircraft system does not meet its reliability requirements, it will not be certified as safe for operation;
  - if an embedded control system fails to meet its performance requirements, the control functions will not operate correctly

Non-functional Requirements (cont.)

Types of Non-functional Requirements



Non-functional Requirements (cont.)

Product requirements

- These requirements specify or constrain the behavior of the software.
- Examples include
  - performance requirements on how fast the system must execute and how much memory it requires
  - reliability requirements that set out the acceptable failure rate
  - security requirements
  - usability requirements

Non-functional Requirements (cont.)

Organizational requirements

- These requirements are broad system requirements derived from policies and procedures in the customer's and developer's organization
- Examples include
  - operational process requirements that define how the system will be used
  - development process requirements that specify the programming language
  - the development environment or process standards to be used
  - environmental requirements that specify the operating environment of the system

Non-functional Requirements (cont.)

External requirements

- This broad heading covers all requirements that are derived from factors external to the system and its development process
- These may include
  - regulatory requirements that set out what must be done for the system to be approved for use by a regulator, such as a central bank
  - legislative requirements that must be followed to ensure that the system operates within the law
  - ethical requirements that ensure that the system will be acceptable to its users and the general public

Non-functional Requirements (cont.)

Metrics for specifying non-functional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Whenever possible, you should write non-functional requirements quantitatively so that they can be objectively tested

Examples

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.

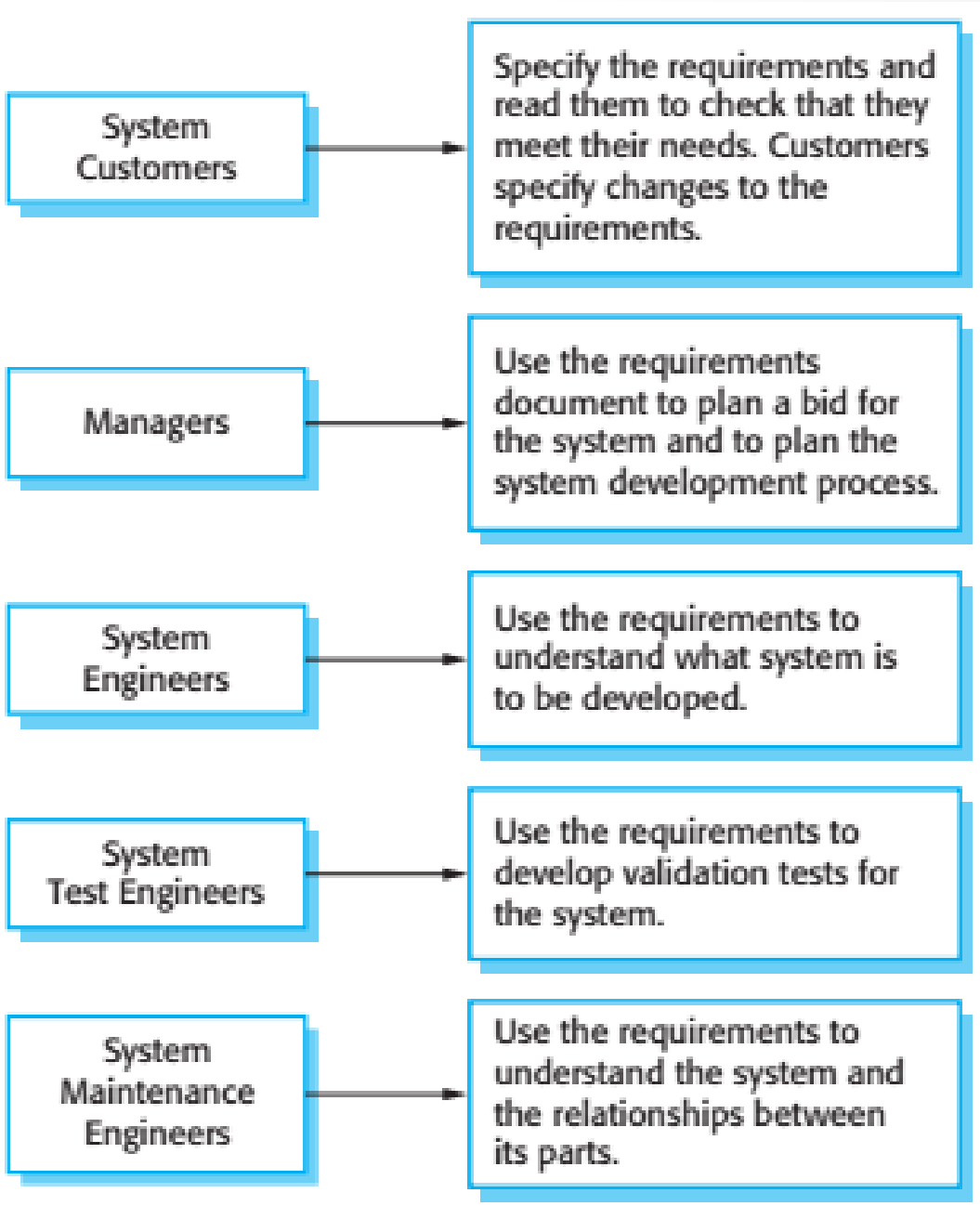
Usability Requirement from a manager:
The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.

Non-functional Requirement:
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

The Software Requirements Specification Document (SRS)

- **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
- Sometimes, the user and system requirements are integrated into a single description
- In other cases, the user requirements are defined in an introduction to the system requirements specification
- If there are a large number of requirements, the detailed system requirements may be presented in a separate document

Users of a requirements document



Requirements Specification

- **The process of writing down the user and system requirements in a requirements document**
- The user and system requirements should be **clear**, **unambiguous**, **easy to understand**, **complete**, and **consistent**
- When writing user requirements, you **should not use** software jargon, structured notations, or formal notations
- When writing system requirements, you should simply describe the external behavior of the system and its operational constraints
  - They should not be concerned with how the system should be designed or implemented

Ways of writing a System Requirements Specification

Notation	Description
Natural language sentences	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.
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.

Natural Language Requirements Specification Guidelines

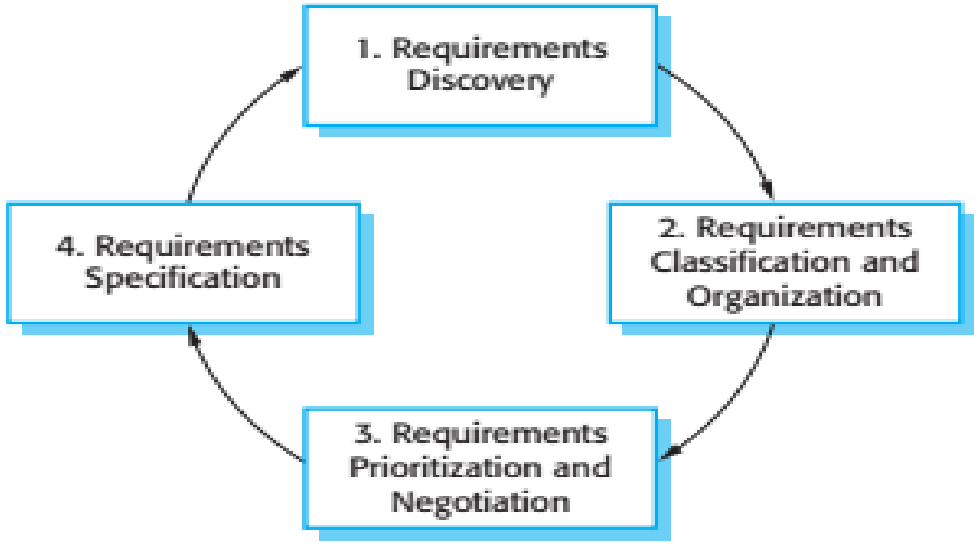
- Invent a standard format and ensure that all requirement definitions adhere to that format
- Use language consistently to distinguish between mandatory and desirable requirements
  - Mandatory requirements are requirements that the system must support and are usually written using '**shall**'
  - Desirable requirements are not essential and are written using '**should**'
- Use text highlighting (bold, italic, or color) to pick out key parts of the requirement
- Do not assume that readers understand technical software engineering language
  - avoid the use of jargon, abbreviations, and acronyms

The Requirements Engineering Process

- Feasibility study
- Requirements elicitation and analysis
- Requirements specification
- Requirements validation
- Requirements management

Requirements elicitation and analysis

- an iterative process that can be represented as a spiral of activities— *requirements discovery*, *requirements classification and organization*, *requirements negotiation*, and *requirements documentation*



Requirements elicitation and analysis (cont.)

Requirements discovery

- This is the process of interacting with stakeholders of the system to discover their requirements
- Domain requirements from stakeholders and documentation are also discovered during this activity

Requirements elicitation and analysis (cont.)

Requirements classification and organization

- This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters
- The most common way of grouping requirements is to use a model of the system architecture to identify sub-systems and to associate requirements with each sub-system
- In practice, requirements engineering and architectural design cannot be completely separate activities

Requirements elicitation and analysis (cont.)

Requirements prioritization and negotiation

- When multiple stakeholders are involved, requirements will conflict
- This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation
- Usually, stakeholders have to meet to resolve differences and agree on compromise requirements

Requirements elicitation and analysis (cont.)

Requirements specification

- The requirements are documented and input into the next round of the spiral
- Formal or informal requirements documents may be produced

Requirements validation

- The process of checking the requirements for validity, consistency, completeness, realism, and verifiability
- It is the process of checking that requirements actually define the system that the customer really wants
- It overlaps with analysis as 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

Requirements validation (cont.)

- different types of checks should be carried out on the requirements

1. *Validity checks*
   - A user may think that a system is needed to perform certain functions. However, further thought and analysis may identify additional or different functions that are required. Systems have diverse stakeholders with different needs and any set of requirements is inevitably a compromise across the stakeholder community
2. *Consistency checks*
   - Requirements in the document should not conflict
   - There should not be contradictory constraints or different descriptions of the same system function

Requirements validation (cont.)

3. *Completeness checks*
   - The requirements document should include requirements that define all functions and the constraints intended by the system user
4. *Realism checks*
   - Using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented
   - These checks should also take account of the budget and schedule for the system development
5. *Verifiability*
   - To reduce the potential for dispute between customer and contractor, system requirements should always be written so that they are verifiable
   - This means that you should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement

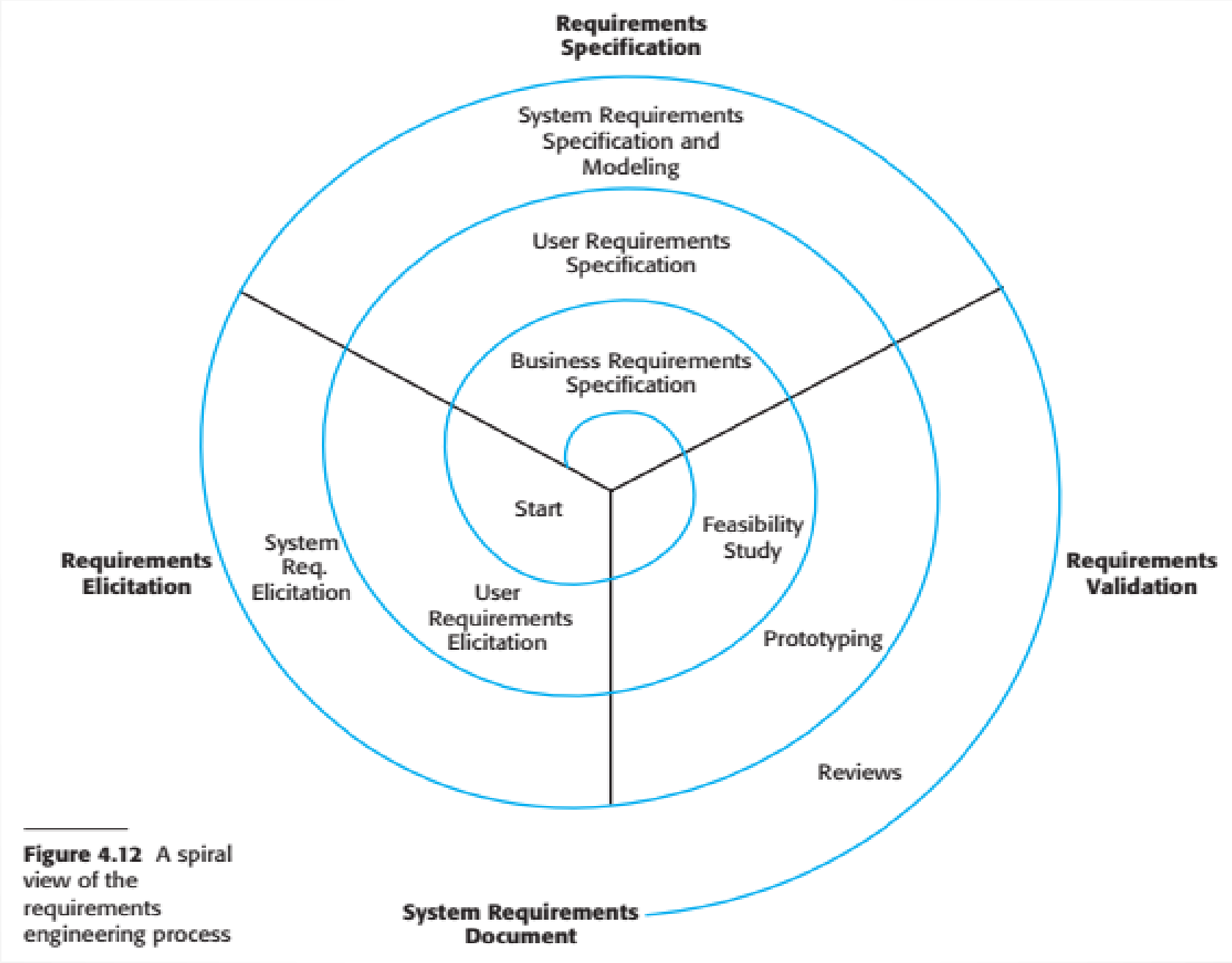


Figure 4.12 A spiral view of the requirements engineering process

Requirements management

- Business, organizational, and technical changes inevitably lead to changes to the requirements for a software system
- Requirements management is the process of understanding, managing and controlling changes to system requirements
- There are several reasons why change is inevitable:

1. The business and technical environment of the system always changes after installation
   - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by

Requirements management (cont.)

2. The people who pay for a system and the users of that system are rarely the same people
   - System customers impose requirements because of organizational and budgetary constraints
   - These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals
3. Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory
   - The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed

Requirements Change Management

Principal stages of Requirements change management

- Requirements change management should be applied to all proposed changes to a system's requirements after the requirements document has been approved
- Change management is essential because you need to decide if the benefits of implementing new requirements are justified by the costs of implementation

Requirements Change Management (cont.)

Problem analysis and change specification

- The process starts with an identified requirements problem or, sometimes, with a specific change proposal
- 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

Requirements Change Management (cont.)

Change analysis and costing

- The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements
- The cost of making the change is estimated both in terms of modifications to the requirements document and, if appropriate, to the system design and implementation
- Once this analysis is completed, a decision is made whether or not to proceed with the requirements change

Requirements Change Management (cont.)

Change implementation

- The requirements document and, where necessary, the system design and implementation, are modified
- You should organize the requirements document so that you can make changes to it without extensive rewriting or reorganization
- As with programs, changeability in documents is achieved by minimizing external references and making the document sections as modular as possible
  - individual sections can be changed and replaced without affecting other parts of the document

End of Presentation