

Requirements Engineering

Slide Credits: Software Engineering by Ian Sommerville

Topics covered



- ✧ Functional and non-functional requirements
- ✧ Requirements engineering processes
- ✧ Requirements elicitation
- ✧ Requirements specification
- ✧ Requirements validation
- ✧ Requirements change

Requirements engineering



- ✧ 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.

KEY TERMS: Services, Constraints

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;
 - Both these statements may be called requirements.

KEY TERMS: High-level abstract statement, mathematical function specification

Requirements abstraction (Davis)



“If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization’s needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system.”

Types of requirement



✧ User requirements Consists of -> Statements, Diagrams

- Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

✧ System requirements

- A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

generated during the requirement engineering process

Mentcare: A patient information system for mental health care



- ✧ A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received.
- ✧ Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.
- ✧ To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.

Mentcare



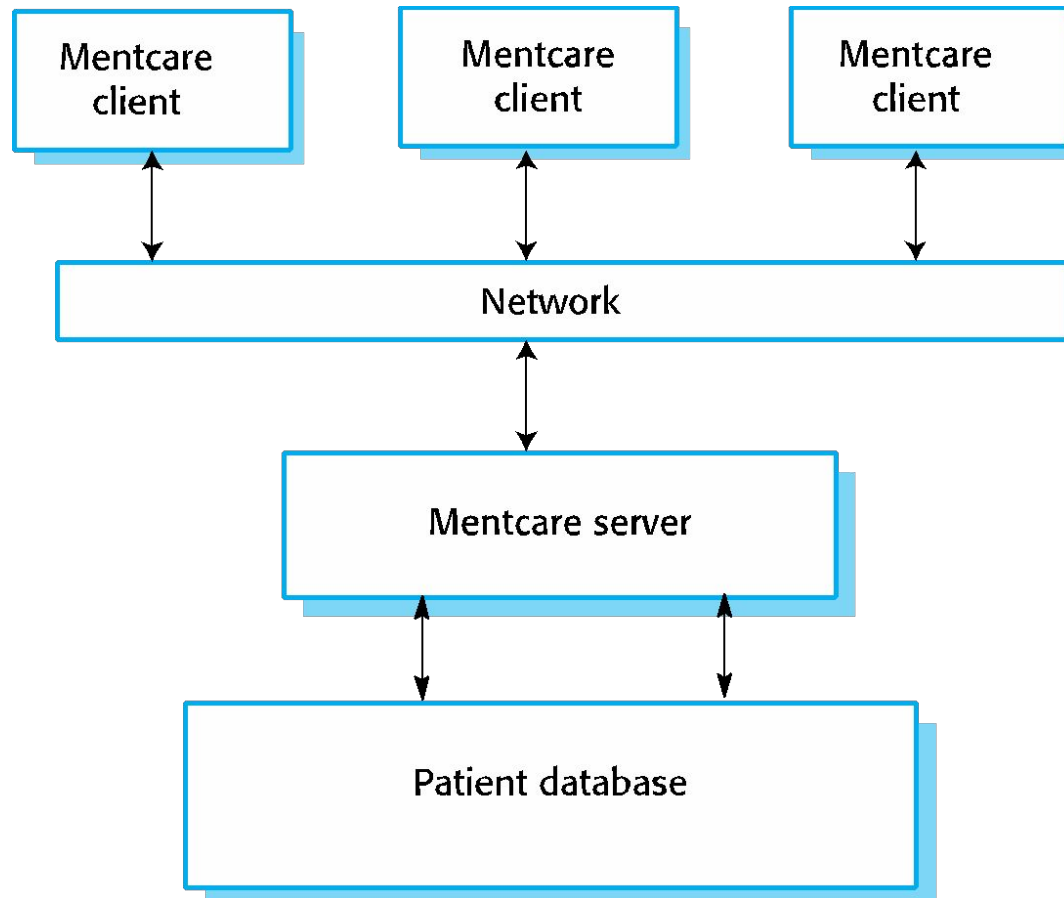
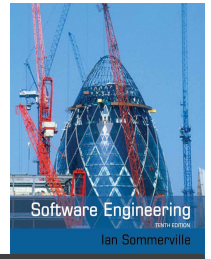
- ✧ **Mentcare is an information system** that is intended for use in clinics.
- ✧ It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- ✧ When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.

Mentcare goals



- ✧ To generate management information that allows health service managers to assess performance against local and government targets.
- ✧ To provide medical staff with timely information to support the treatment of patients.

The organization of the Mentcare system



Key features of the Mentcare system



✧ Individual care management

- Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.

✧ Patient monitoring

- The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.

✧ Administrative reporting

- The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

Mentcare system concerns



✧ Privacy

- It is essential that patient information **is confidential** and is never disclosed to anyone apart from authorised medical staff and the patient themselves.

✧ Safety

- Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, **the system should warn** medical staff about potentially suicidal or dangerous patients.
- The **system must be available** when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

User and system requirements



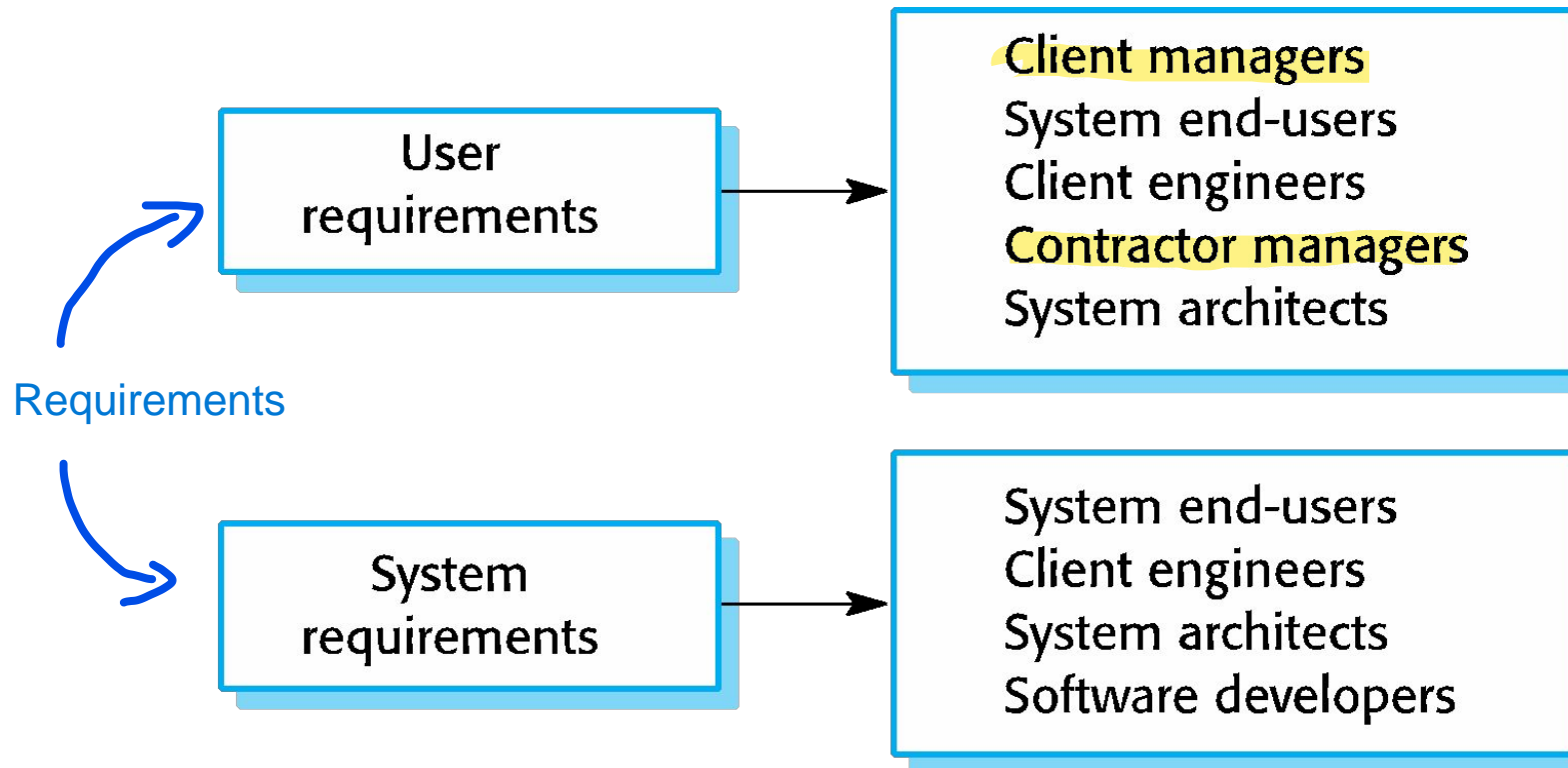
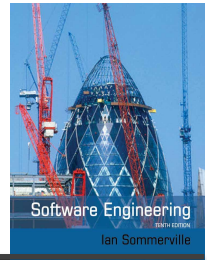
User requirements definition

- 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



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.

Stakeholders in the Mentcare system



- ✧ A **medical ethics manager** who must ensure that the system meets current ethical guidelines for patient care.
- ✧ **Health care managers** who obtain management information from the system.
- ✧ **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.



Agile methods and requirements

- ✧ Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly.
- ✧ The requirements document is therefore always out of date.
- ✧ Agile methods usually use incremental requirements engineering and may express requirements as 'user stories' (discussed in Chapter 3).
- ✧ This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

Functional and non-functional requirements

Functional and non-functional requirements



✧ Functional requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- May state what the system should not do.

✧ Non-functional requirements

- Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Often apply to the system as a whole rather than individual features or services.

✧ Domain requirements

- Constraints on the system from the domain of operation

Functional requirements



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

Mentcare system: functional requirements



- ✧ A user shall be able to **search** the appointments lists for all clinics.
- ✧ The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- ✧ Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements imprecision



- ✧ Problems arise when functional requirements are not precisely stated.
- ✧ Ambiguous requirements may be interpreted in different ways by developers and users.
- ✧ Consider the term 'search' in requirement 1
 - User intention – search for a patient name across all appointments in all clinics;
 - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

Requirements completeness and consistency



- ✧ In principle, requirements should be both complete and consistent.
- ✧ Complete
 - They should include descriptions of all facilities required.
- ✧ Consistent
 - There should be no conflicts or contradictions in the descriptions of the system facilities.
- ✧ In practice, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

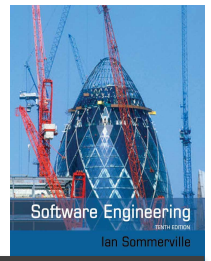
Non-functional requirements



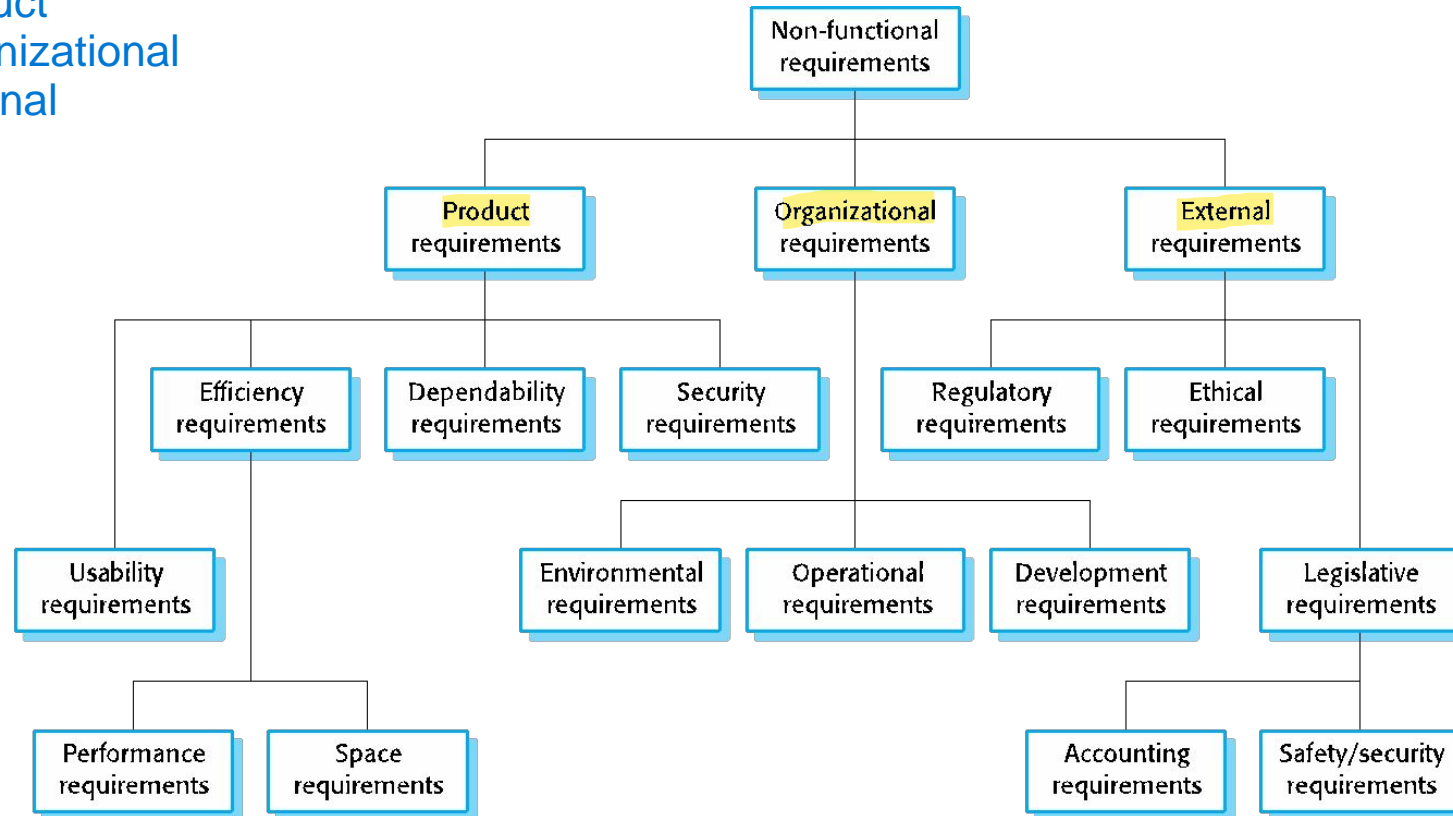
- ✧ These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- ✧ Process requirements may also be specified mandating a particular IDE, programming language or development method.
- ✧ Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Apply to whole system rather than a feature or service.

Types of nonfunctional requirement



Product
Organizational
External

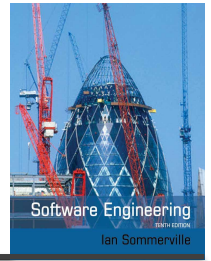


Non-functional requirements implementation



- ✧ Non-functional requirements may affect the overall architecture of a system rather than the individual components.
 - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- ✧ A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
 - It may also generate requirements that restrict existing requirements.

Non-functional classifications



✧ Product requirements

- Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

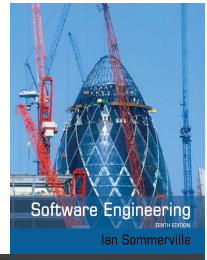
✧ Organisational requirements

- Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.

✧ External requirements

- Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Examples of nonfunctional requirements in the Mentcare system



Product requirement

The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). **Downtime** within normal working hours shall not exceed five seconds in any one day.

Organizational requirement

Users of the Mentcare 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.

Goals and requirements



- ✧ Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.

✓ ✧ Goal

- A general intention of the user such as ease of use.
- ✧ Verifiable non-functional requirement
 - A statement using some measure that can be objectively tested.
- ✧ Goals are helpful to developers as they convey the intentions of the system users.

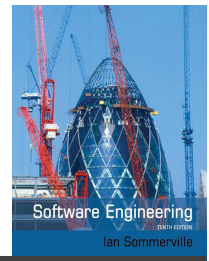
Usability requirements



- ✧ The system should be **easy to use** by medical staff and should be organized in such a way that **user errors are minimized.** (Goal)
- ✧ Testable 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.

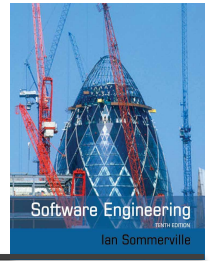
Metrics for specifying nonfunctional requirements

SSERRP



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

Another Example: The LIBSYS system



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

Examples of functional requirements of LIBSYS



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

Requirements imprecision in LIBSYS



✧ Ambiguity

✧ Consider the term 'appropriate viewers'

- User intention - special purpose viewer for each different document type;
- Developer interpretation - Provide a text viewer that shows the contents of the document.

Non-functional requirements LIBSYS



✧ Product requirement

- The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.

✧ Organisational requirement

- The system development process and deliverable documents shall conform to the process and deliverables defined in **XYZCo-SP-STAN-95**.

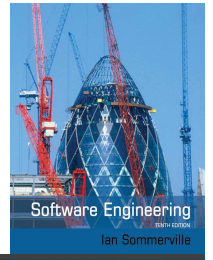
✧ External requirement

- The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

Domain Requirements of LIBSYS



- ✧ There shall be a **standard user interface** to all **databases** which shall be based on the Z39.50 standard.
- ✧ Because of **copyright restrictions**, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.



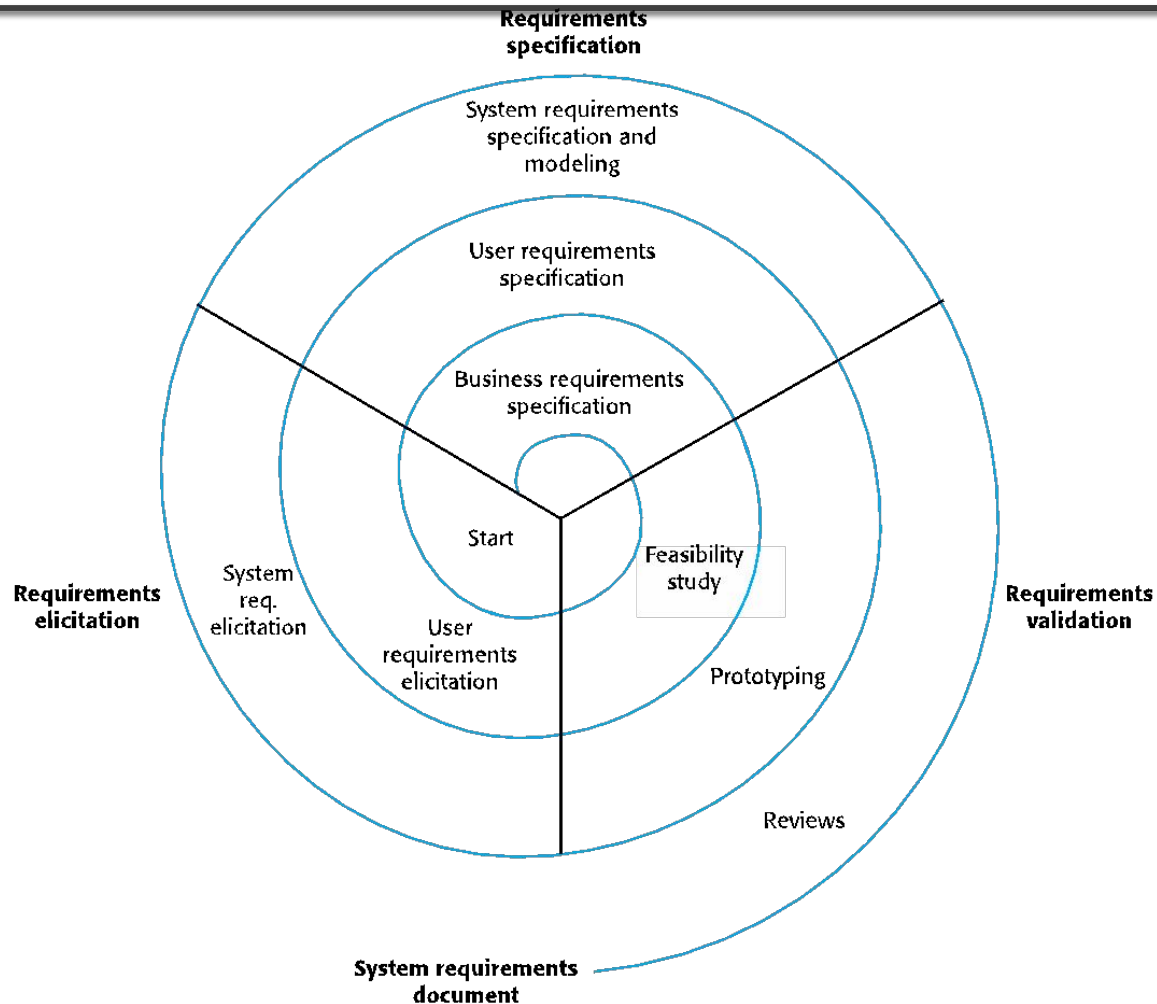
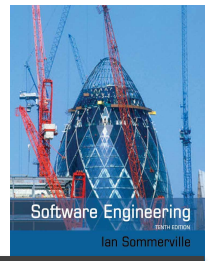
Requirements engineering processes

Requirements engineering processes



- ✧ The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- ✧ However, there are a number of generic activities common to all processes
 - Requirements elicitation;
 - Requirements analysis;
 - Requirements validation;
 - Requirements management.
- ✧ In practice, RE is an iterative activity in which these processes are interleaved.

A spiral view of the requirements engineering process



Requirements specification

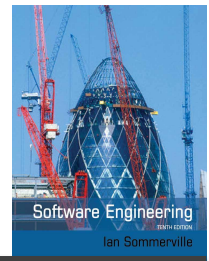
Requirement Elicitation: In requirements engineering, requirements elicitation is the practice of researching and discovering the requirements of a system from users, customers, and other stakeholders. The practice is also sometimes referred to as "requirement gathering".

Requirements specification



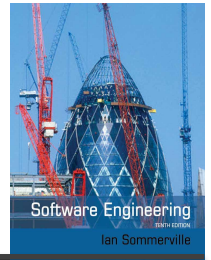
- ✧ The process of writing down the user and system requirements in a requirements document.
- ✧ User requirements have to be understandable by end-users and customers who do not have a technical background.
- ✧ System requirements are more detailed requirements and may include more technical information.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

Ways of writing a system requirements specification



Notation	Description
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.
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

Requirements and design



- ✧ In principle, requirements should state what the system should do and the design should describe how it does this.
- ✧ In practice, requirements and design are inseparable
 - A system architecture may be designed to structure the requirements;
 - The system may inter-operate with other systems that generate design requirements;
 - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
 - This may be the consequence of a regulatory requirement.

Natural language specification



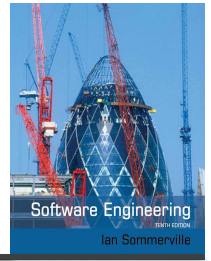
- ✧ Requirements are written as natural language sentences supplemented by diagrams and tables.
- ✧ Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

Guidelines for writing requirements



- ✧ Invent a **standard format** and use it for all requirements.
- ✧ Use language in a **consistent way**. Use shall for mandatory requirements, should for desirable requirements.
- ✧ Use **text highlighting** to identify key parts of the requirement.
- ✧ **Avoid** the use of **computer jargon**.
- ✧ Include an **explanation** (rationale) of why a requirement is necessary.

Problems with natural language



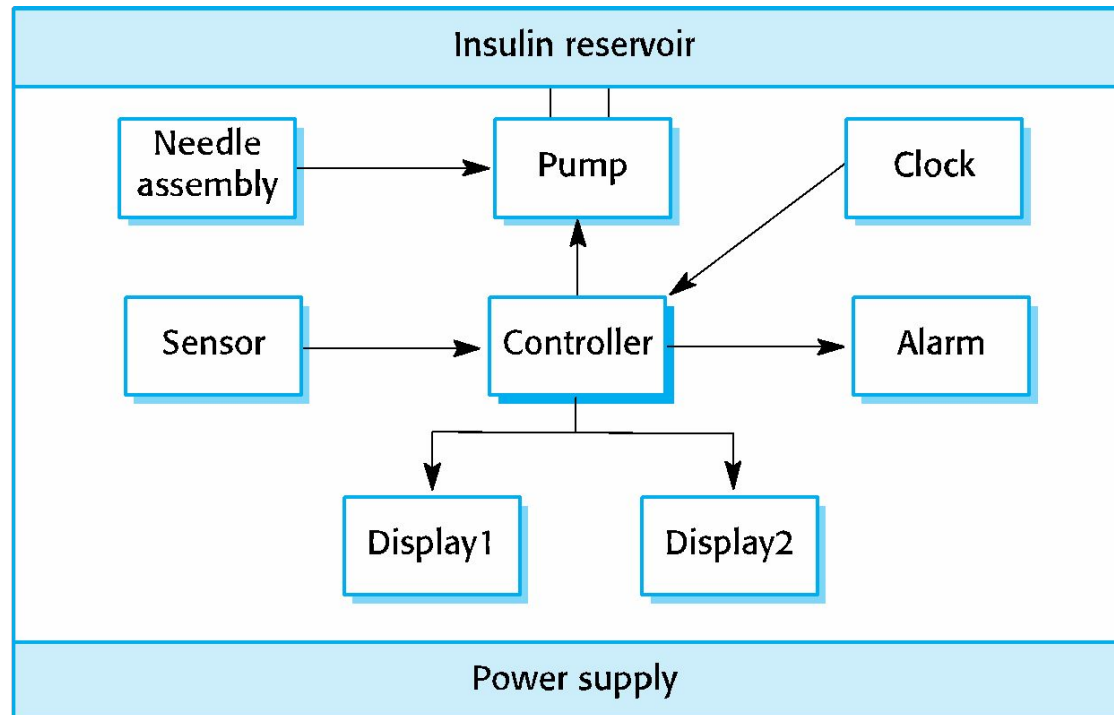
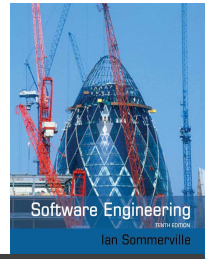
- ✧ Lack of clarity
 - Precision is difficult without making the document difficult to read.
- ✧ Requirements confusion
 - Functional and non-functional requirements tend to be mixed-up.
- ✧ Requirements amalgamation
 - Several different requirements may be expressed together.

Insulin pump control system

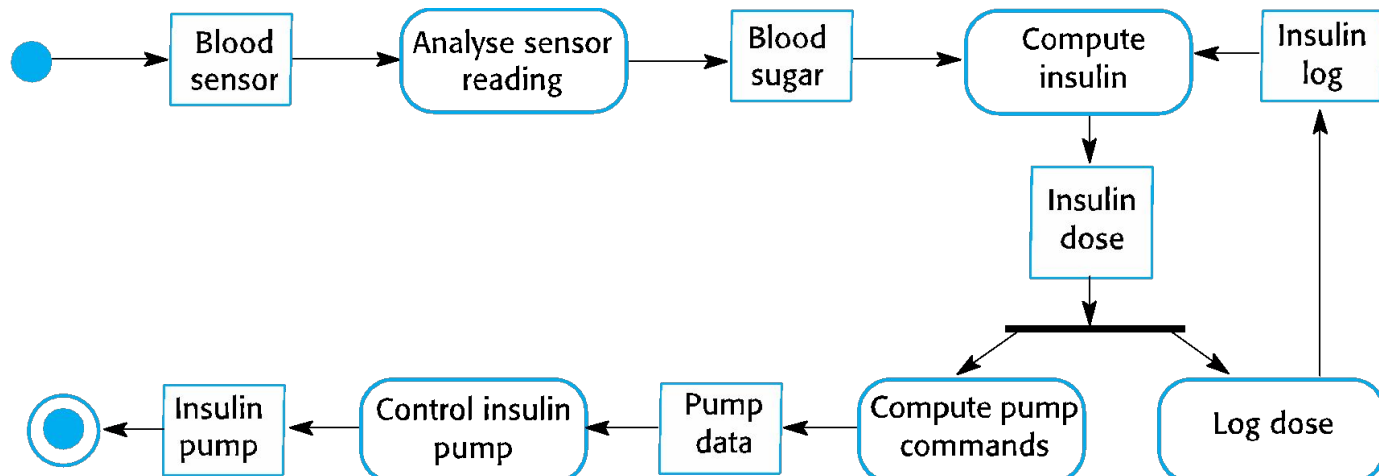
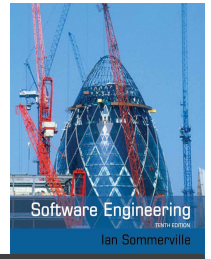


- ✧ Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.
- ✧ Calculation based on the rate of change of blood sugar levels.
- ✧ Sends signals to a micro-pump to deliver the correct dose of insulin.
- ✧ Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage.

Insulin pump hardware architecture



Activity model of the insulin pump



Essential high-level requirements



- ✧ The system shall be **available** to deliver insulin when required.
- ✧ The system shall **perform reliably** and deliver the correct amount of insulin to counteract the current level of blood sugar.
- ✧ The system must therefore be **designed** and implemented to ensure that the **system always meets these requirements.**

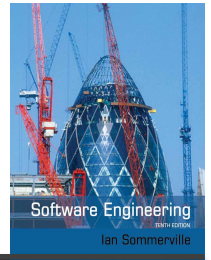
Example requirements for the insulin pump software system



3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

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.

A structured specification of a requirement for an insulin pump



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).

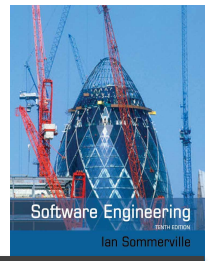
Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose—the dose in insulin to be delivered.

Destination Main control loop.

Function -> Description -> Inputs -> Source -> Output -> Destination

A structured specification of a requirement for an insulin pump



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.

Requirements

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.

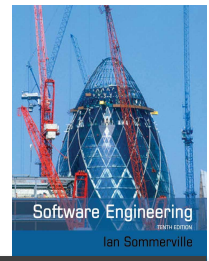
Side effects None.

Tabular specification



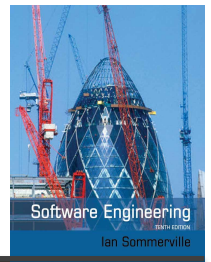
- ✧ Used to supplement natural language.
- ✧ Particularly useful when you have to define a number of possible alternative courses of action.
- ✧ For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Tabular specification of computation for an insulin pump



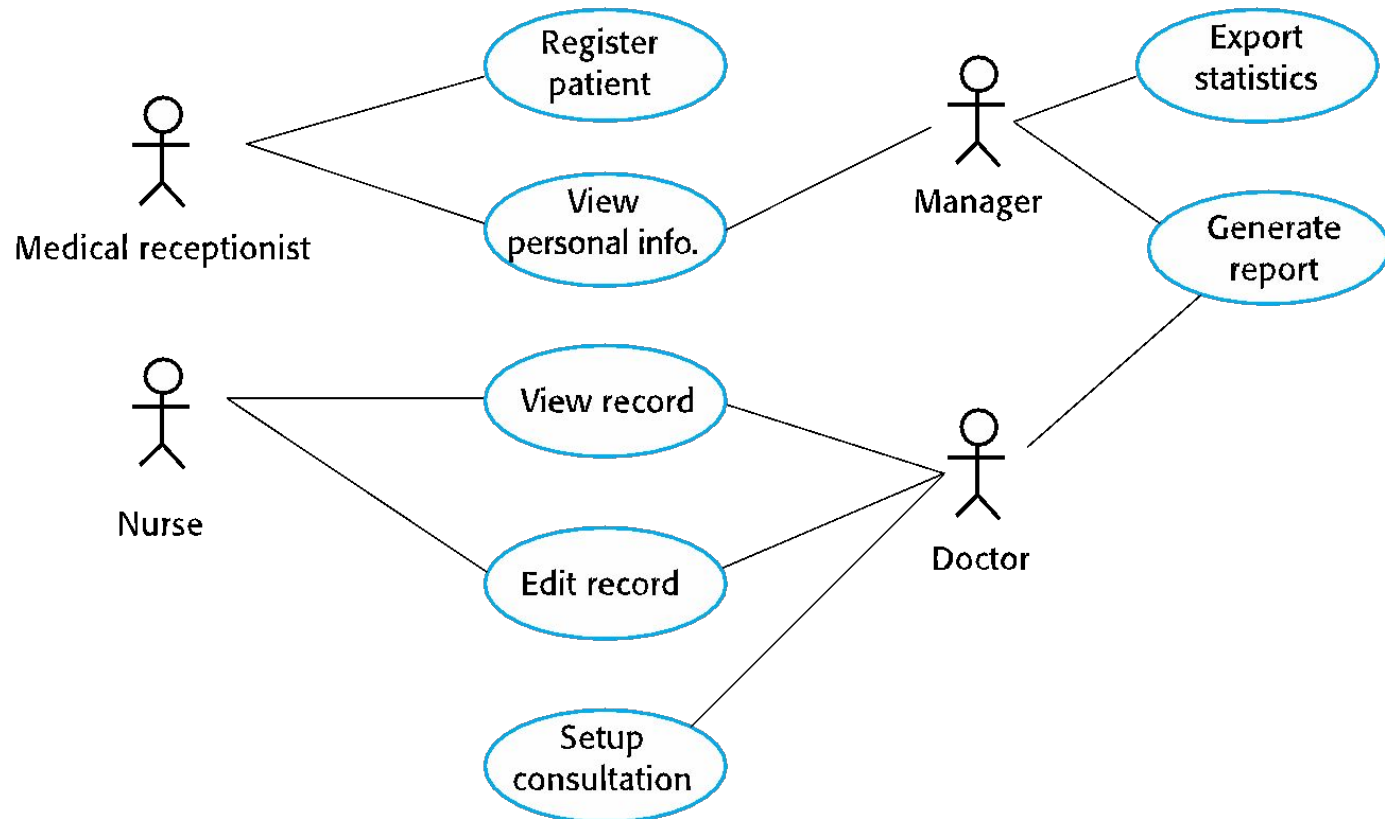
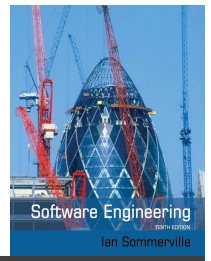
Condition	Action
Sugar level falling ($r2 < r1$)	CompDose = 0
Sugar level stable ($r2 = r1$)	CompDose = 0
Sugar level increasing and rate of increase decreasing ($(r2 - r1) < (r1 - r0)$)	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ($(r2 - r1) \geq (r1 - r0)$)	CompDose = round $((r2 - r1)/4)$ If rounded result = 0 then CompDose = MinimumDose

Use cases Unified Modeling Language



- ✧ Use-cases are a kind of scenario that are included in the UML.
- ✧ Use cases identify the actors in an interaction and which describe the interaction itself.
- ✧ A set of use cases should describe all possible interactions with the system.
- ✧ High-level graphical model supplemented by more detailed tabular description (see Chapter 5).
- ✧ UML sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use cases for the Mentcare system

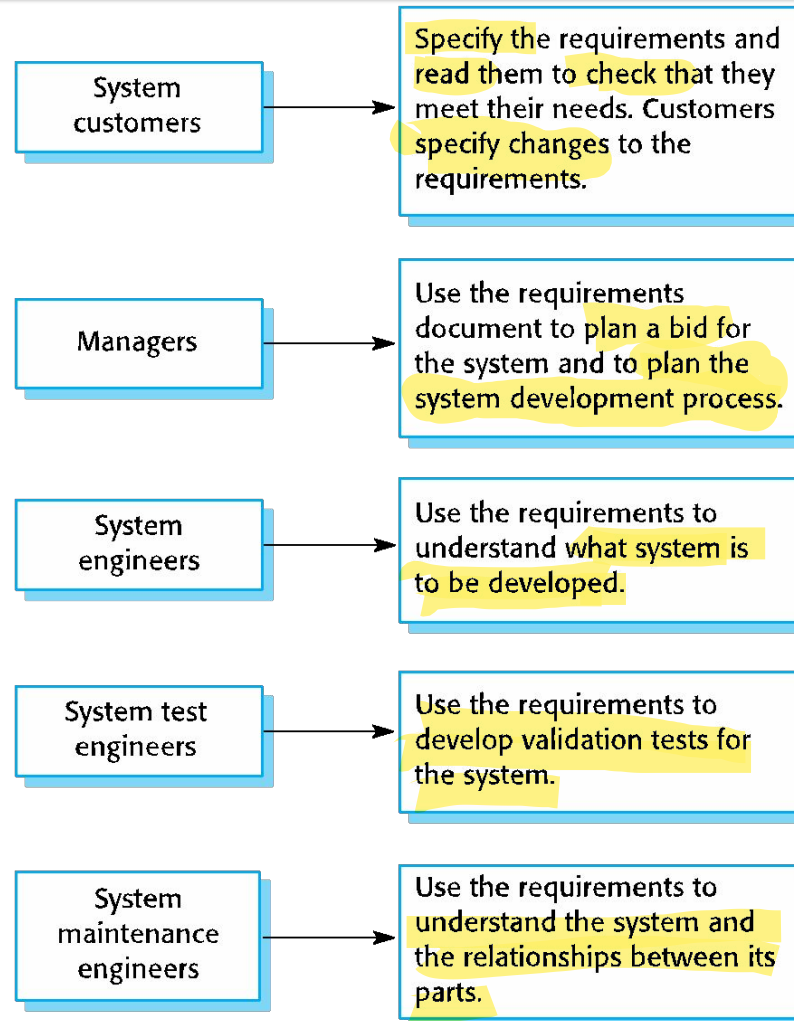
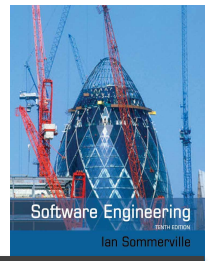


The software requirements document



- ✧ The software requirements document is the official statement of what is required of the system developers.
- ✧ Should include both a definition of user requirements and a specification of the system requirements.
- ✧ It is NOT a design document. As far as possible, it should state WHAT the system should do rather than HOW it should do it.

Users of a requirements document

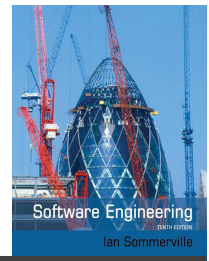


Requirements document variability



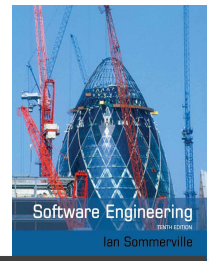
- ✧ Information in requirements document depends on type of system and the approach to development used.
- ✧ Systems developed incrementally will, typically, have less detail in the requirements document.
- ✧ Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

The structure of a requirements document

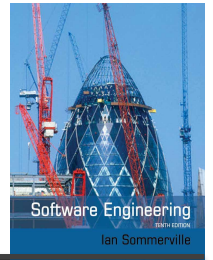


Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The structure of a requirements document



Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.



Requirements validation

Requirements validation



- ✧ Concerned with demonstrating that the requirements define the system that the customer really wants.
- ✧ Requirements error costs are high so validation is very important
 - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking



- ✧ **Validity**. Does the system provide the functions which best support the customer's needs?
- ✧ **Consistency**. Are there any requirements conflicts?
- ✧ **Completeness**. Are all functions required by the customer included?
- ✧ **Realism**. Can the requirements be implemented given available budget and technology
- ✧ **Verifiability**. Can the requirements be checked?

Requirements validation techniques



✧ Requirements reviews

- Systematic **manual analysis** of the requirements.

✧ Prototyping

- Using an **executable model** of the system to check requirements.
Covered in Chapter 2.

✧ Test-case generation

- Developing **tests** for requirements to check testability.