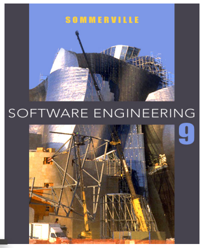
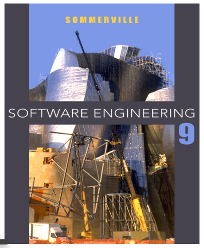
Unit – II
Requirements Engineering



Unit – II **8 Hours**

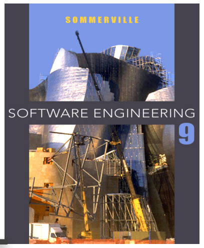
Requirements Engineering: Functional and non-functional requirements: Functional requirements, non-functional requirements, Case studies, The Software requirements document, Introduction to Requirements specification, Requirements Engineering processes: Requirement Elicitation and Analysis.

Topics covered



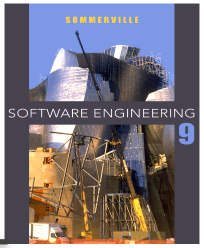
- Functional and non-functional requirements
- The software requirements document
- Requirements specification
- Requirements engineering processes
- Requirements elicitation and analysis
- Requirements validation
- Requirements management

Requirements engineering



- 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.
- The process of finding out, analysing, documenting and checking these services and constraints.
- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

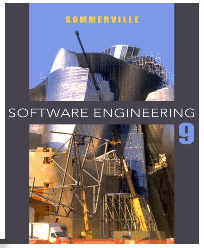
What is a requirement?



- It may range from ***a high-level <u>abstract statement of a service</u>*** or ***of a <u>system constraint</u> to a detailed mathematical functional specification***.

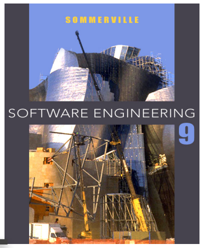
- This is inevitable as requirements may serve a **dual function**
  - May be the basis for **a bid for a contract** - therefore must be open to interpretation;
  - May be the basis for the **contract itself** - therefore must be defined in detail;
  - Both these statements may be called **requirements**.

Requirements abstraction (Davis)

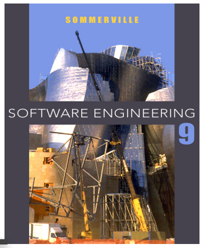


"If a company wishes to **let a contract** for a large software development project, it must define its needs in **a sufficiently abstract way** that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways 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

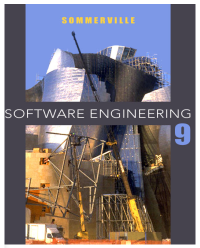


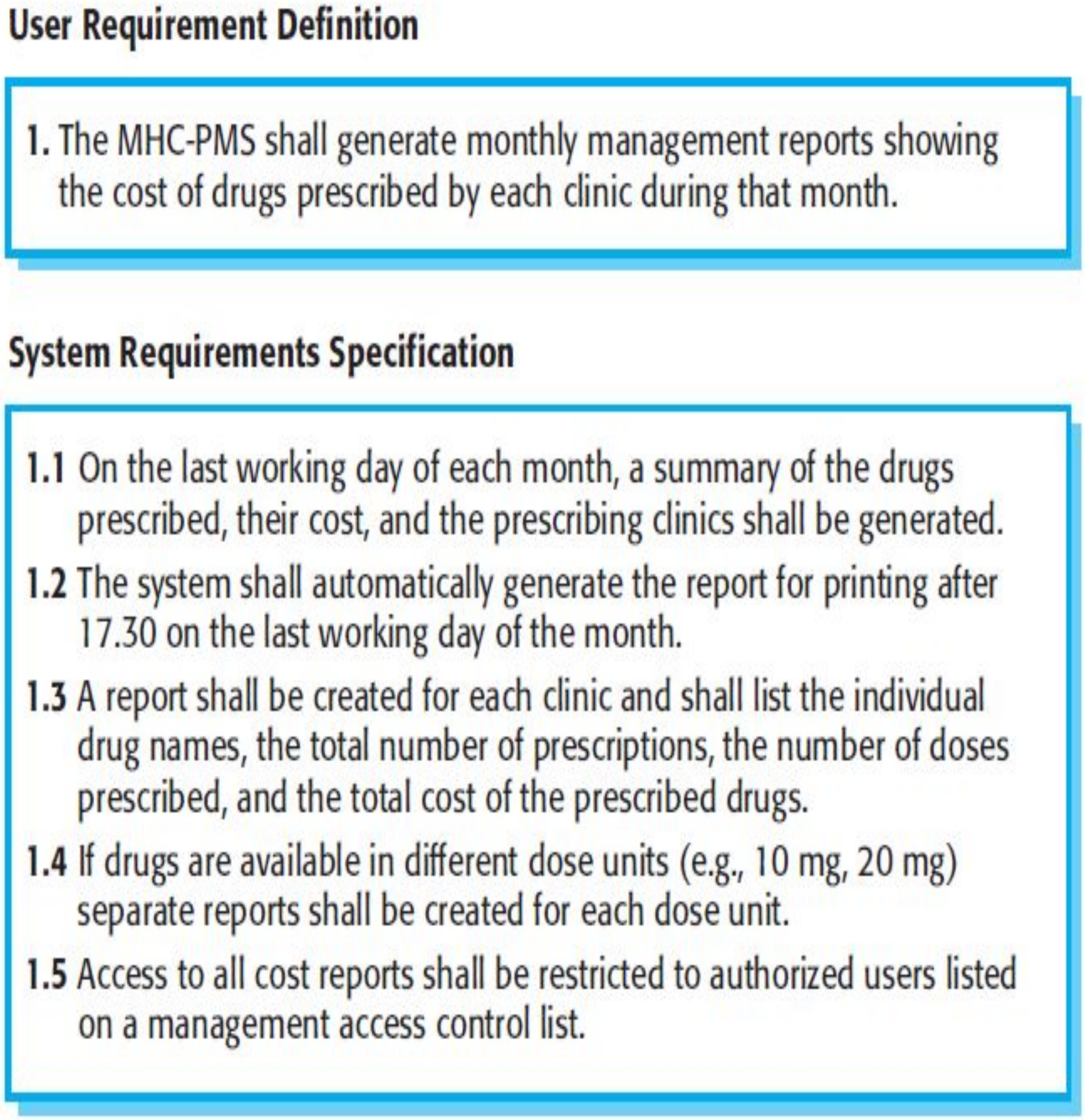
- User requirements
  - statements, in a **natural language plus diagrams**, of ***what services the system is expected to provide to system users*** and the ***constraints*** under which it must operate. Written for customers
- System requirements
  - are ***more detailed descriptions of the software 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.



- Different levels of requirements are useful because they communicate information about the system to different types of reader.

- The system requirements provide more specific information about the services and functions of the system that is to be implemented





User Requirement Definition

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

System Requirements Specification

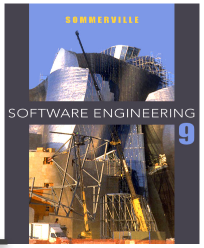
1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.

1.2 The system shall automatically 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., 10 mg, 20 mg) separate reports shall be created for each dose unit.

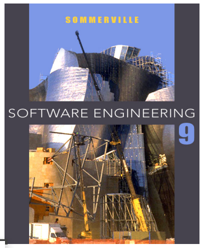
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

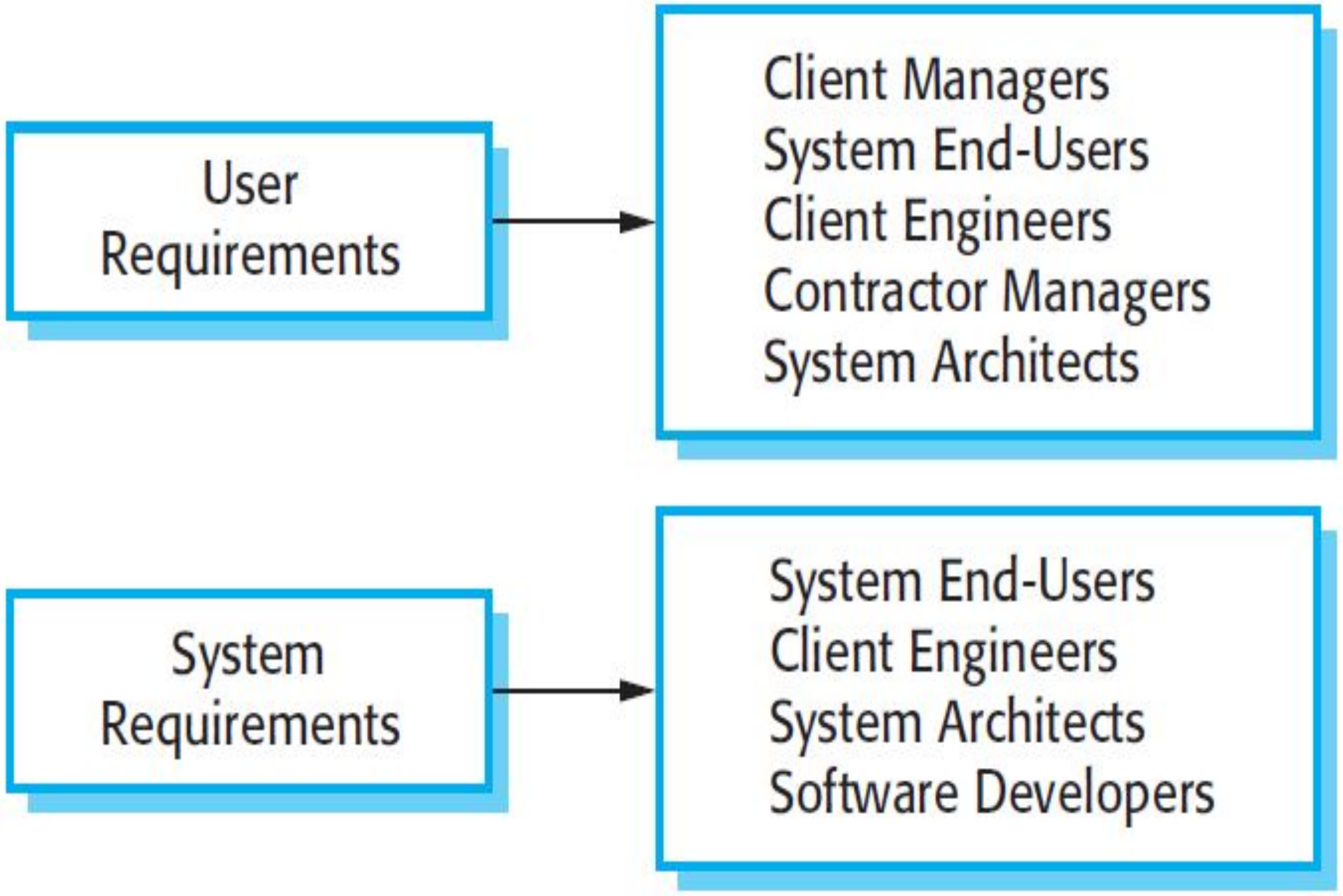


- ***The readers of the user requirements*** are **not usually** concerned with **how the system will be implemented** and may be **managers** who are **not interested** in the **detailed facilities of the system**.

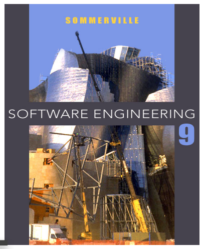
- ***The readers of the system requirements*** need to know more precisely ***what the system will do*** because they are concerned with ***how it will support the business processes*** or because they are involved in the ***system implementation***

Readers of different types of requirements specification



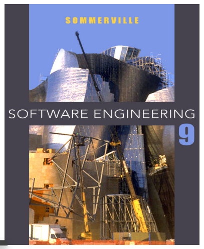


Functional and non-functional requirements



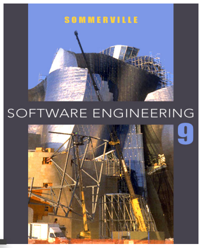
- Functional requirements
  - **Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.**
  - **May state what the system should not do.**
- Non-functional requirements
  - Constraints on the **services 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



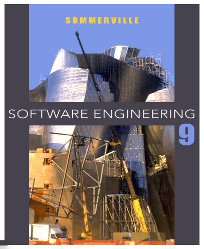
- Describe **functionality** or **system services**.
- Depend on the **type of software**, **expected users** and the **type of system** where the software is used.
- Functional user requirements may be **high-level statements of what the system should do**.
- Functional system requirements should describe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

Functional requirements for the MHC-PMS



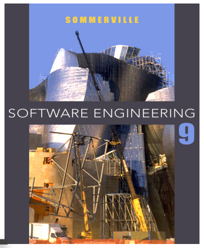
- **A user** shall be able to **search** the **appointments lists for all clinics.**
- The system shall **generate each day, for each clinic, a list of patients** who are **expected to attend appointments that day.**
- Each staff member using the **system** shall be **uniquely identified** by **his or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

Requirements imprecision



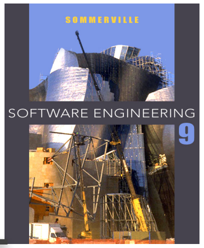
- **Problems arise when requirements are not precisely stated.**
- **Ambiguous requirements** may be interpreted in different ways by developers and users.
- Consider the term **'search'** in requirement 1
  - **User intention** – search for a patient name across all appointments in all clinics;
  -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

Requirements completeness and consistency



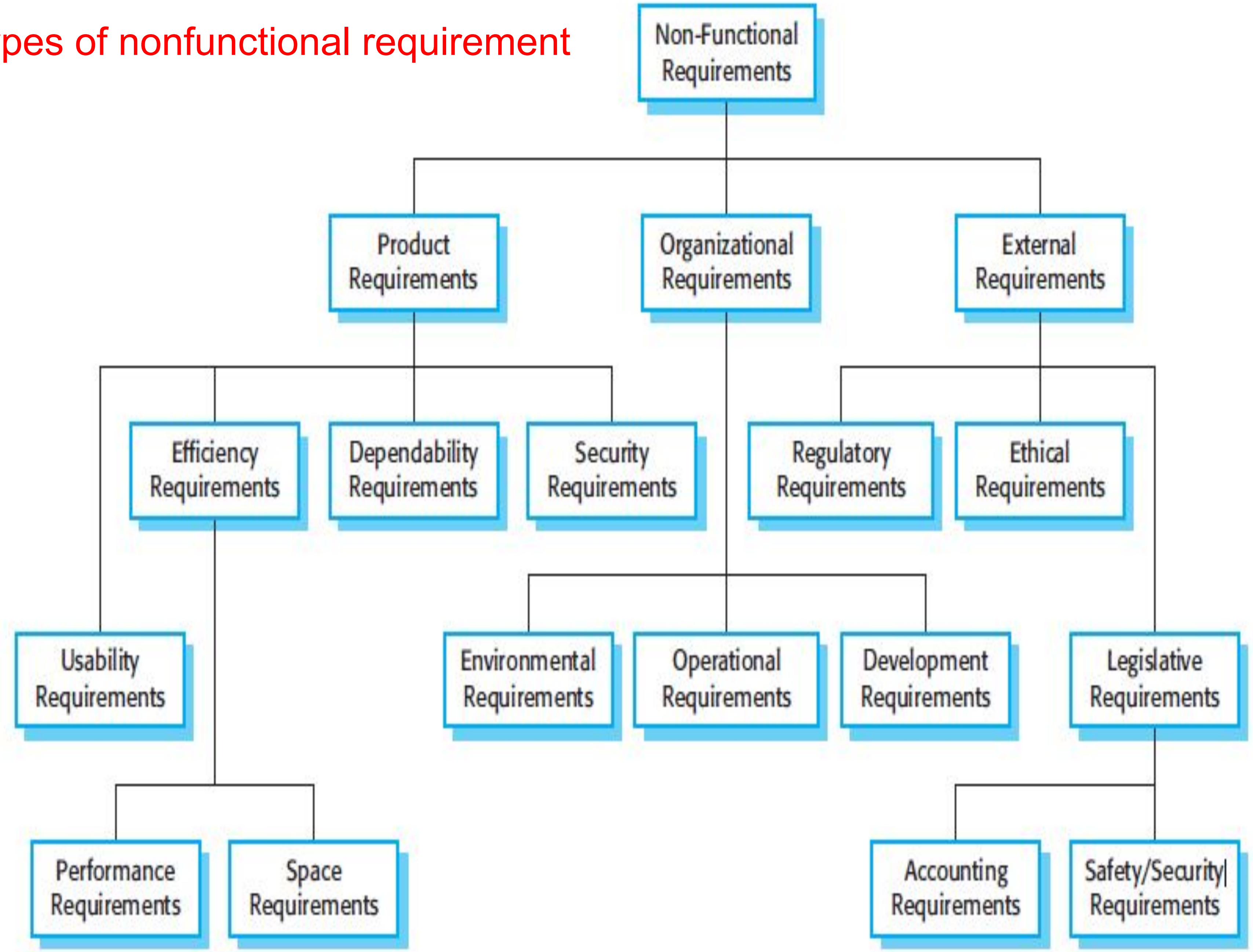
- In principle, requirements should be both complete and consistent.
- **Complete**
  - **They should include descriptions of all facilities required.**
- **Consistent**
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is impossible to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

Non-functional requirements

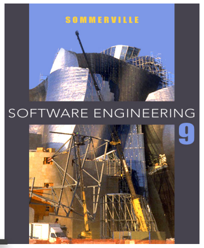


- These define **system properties** and **constraints** e.g. ***reliability, response time*** and ***storage requirements.*** **Constraints** are ***I/O device capability***, ***system representations, etc.***
- **Process requirements** may also be specified **mandating a particular IDE**, **programming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

Types of nonfunctional requirement

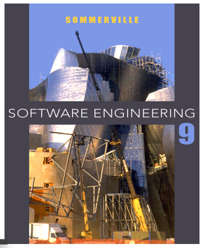


Non-functional requirements implementation



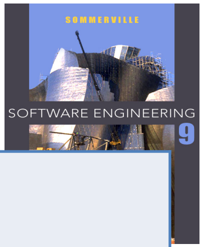
- Non-functional requirements may ***affect the overall architecture of a system*** rather than the individual components.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, 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 MHC-PMS

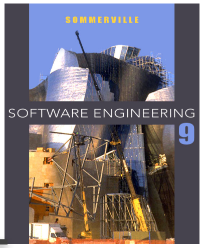


Product requirement
The MHC-PMS shall be ***available*** to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

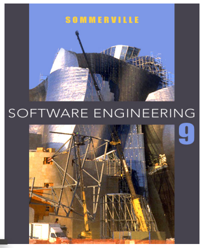
Organizational requirement
Users of the MHC-PMS system shall ***authenticate*** themselves using their health authority identity card.

External requirement
The system shall implement ***patient privacy provisions*** as set out in HStan-03-2006-priv.

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**.

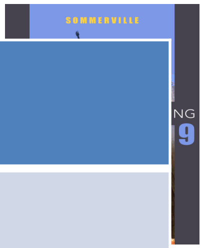


Usability requirements

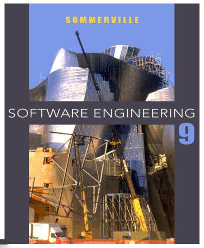
- The system should be ***easy to use*** by ***medical staff*** and should be ***organized*** in such a way that ***user errors*** are ***minimized.*** **(Goal)**

- ***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.*** ***(Testable non-functional requirement)***

Metrics for specifying nonfunctional requirements

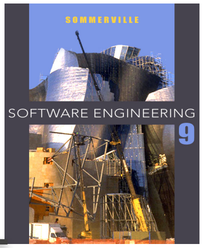


Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart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

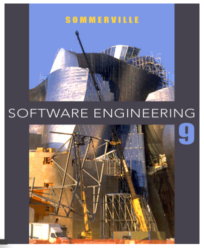


- Non-functional requirements such as ***reliability, safety,*** and ***confidentiality requirements*** are particularly important for critical systems

Key points

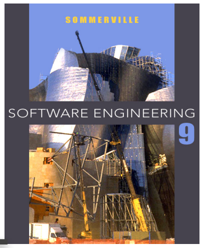


- Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- Non-functional requirements often constrain the system being developed and the development process being used.
- They often relate to the emergent properties of the system and therefore apply to the system as a whole.



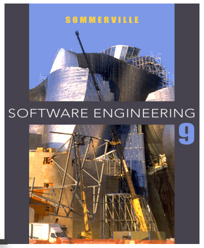
Chapter 4 – Requirements Engineering

The software requirements document
or
Software Requirements Specification (SRS)



- The software requirements document is the **official statement** of ***what the system developers should implement***
- 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 set of **WHAT the system should do** rather than HOW it should do it.
- Requirements documents are **essential** ***when an outside contractor is developing the software system***

Agile methods and requirements



- Many **agile methods** argue that producing a requirements document is a **waste of time** as **requirements change so quickly.**
- The **document** is therefore always **out of date**.
- Methods such as **XP (Extreme Programming)** use ***incremental requirements engineering*** and ***express requirements as*** **'user stories'** (The user then prioritizes requirements for implementation in the next increment of the system.).
- This is **practical for business systems** but **problematic** for **systems that require a lot of pre-delivery analysis (e.g. critical systems)** or systems developed by several teams.

Users of a requirements document

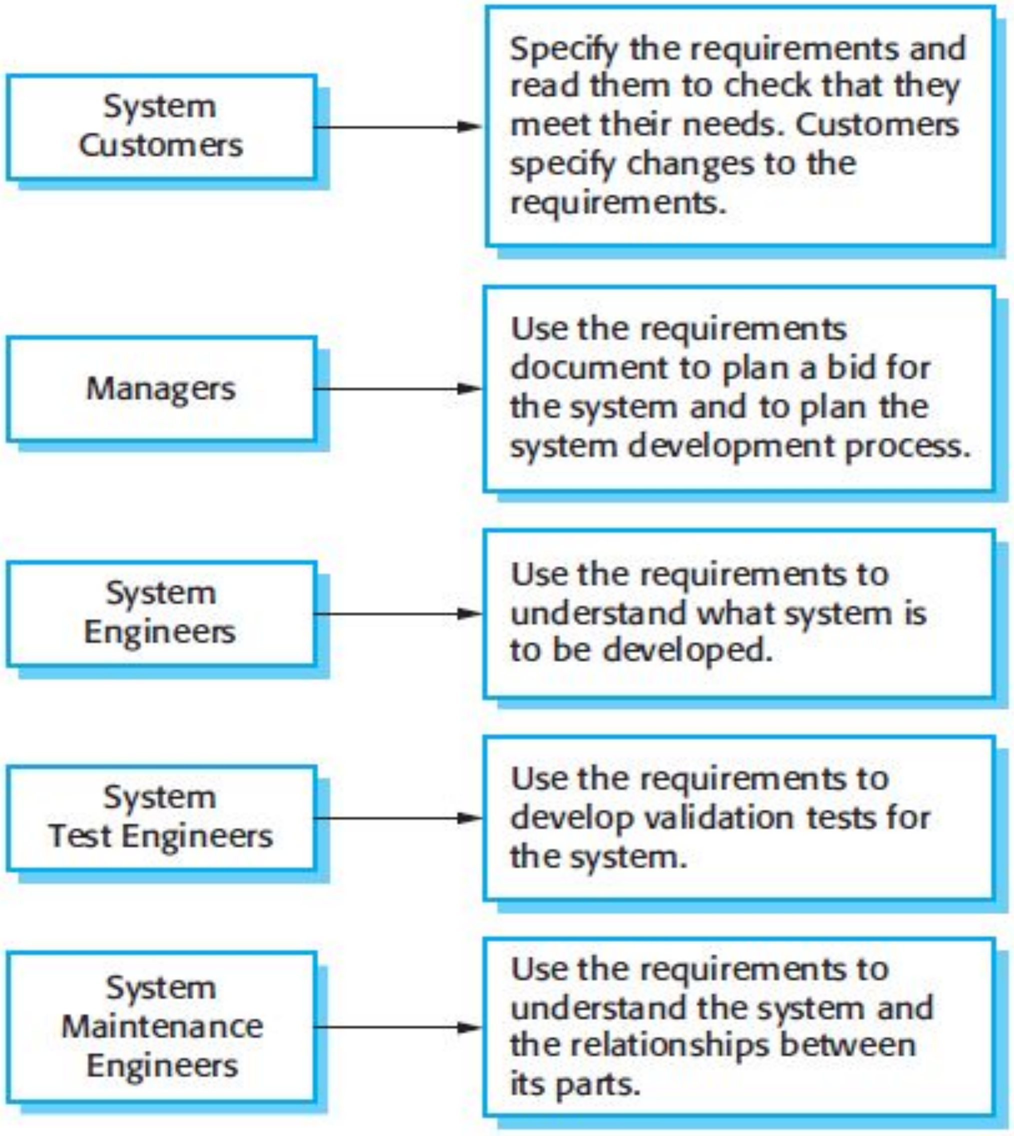
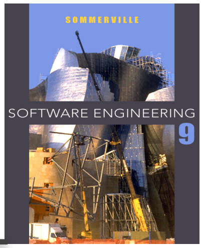

System Customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System Engineers
Use the requirements to understand what system is to be developed.
System Test Engineers
Use the requirements to develop validation tests for the system.
System Maintenance Engineers
Use the requirements to understand the system and the relationships between its parts.

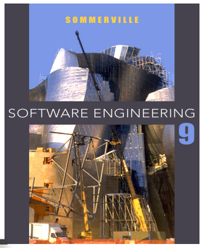
Figure 4.6 Users of a requirements document



- The **diversity of possible users** means that the **requirements document** has to be a ***compromise*** between *communicating the requirements to customers*, *defining the requirements in precise detail for developers* and *testers*, and *including information about possible system evolution.*

- Information on anticipated changes can help system designers avoid restrictive design decisions and help system maintenance engineers who have to adapt the system to new requirements.

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.

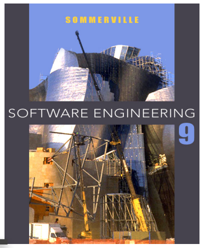
The structure of a requirements document

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 specification

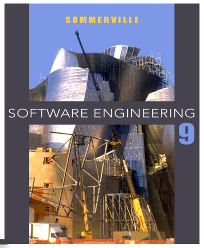


- The process of **writing down** the **user** and **system requirements** in a **requirements document**.
- **User requirements** have to be understandable by **end-users** and **customers** who **do not have a technical background.**
- **System requirements** are **more detailed requirements** and 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 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

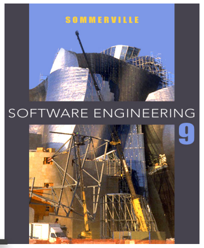
Requirements and design



- 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.

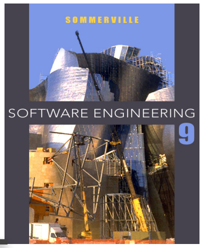
Natural language specification



- Requirements are written as **natural language sentences supplemented by diagrams and tables**.

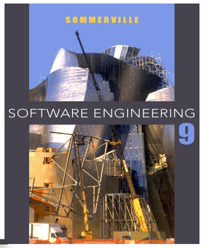
- Used for writing requirements because it is **expressive**, **intuitive and universal**. This means that the requirements can be **understood by users and customers**.

Guidelines for writing requirements



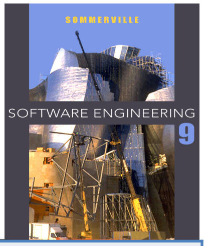
- Invent a **standard format** and use it for all requirements.
- Use language in a **consistent way**. Use '**shall**' for **mandatory requirements**, ***'should'*** for ***desirable requirements.***
- **Use text highlighting** (bold, italic or color) to **identify key parts of** the requirement.
- **Avoid** the **use of computer jargon**.
- Include an **explanation (rationale) of why the requirement has been included.**

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.

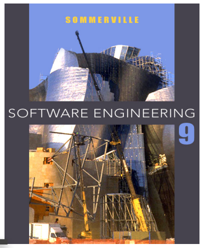
Example requirements for the insulin pump software system



3.2 **The system shall <u>measure the blood sugar and deliver insulin, if required, every 10 minutes</u>.** *(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 <u>self-test routine</u> 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.)*

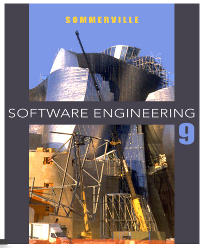
Structured specifications



- An approach to writing requirements where the **freedom of the requirements writer** is **limited** and requirements are written in a **standard way.**

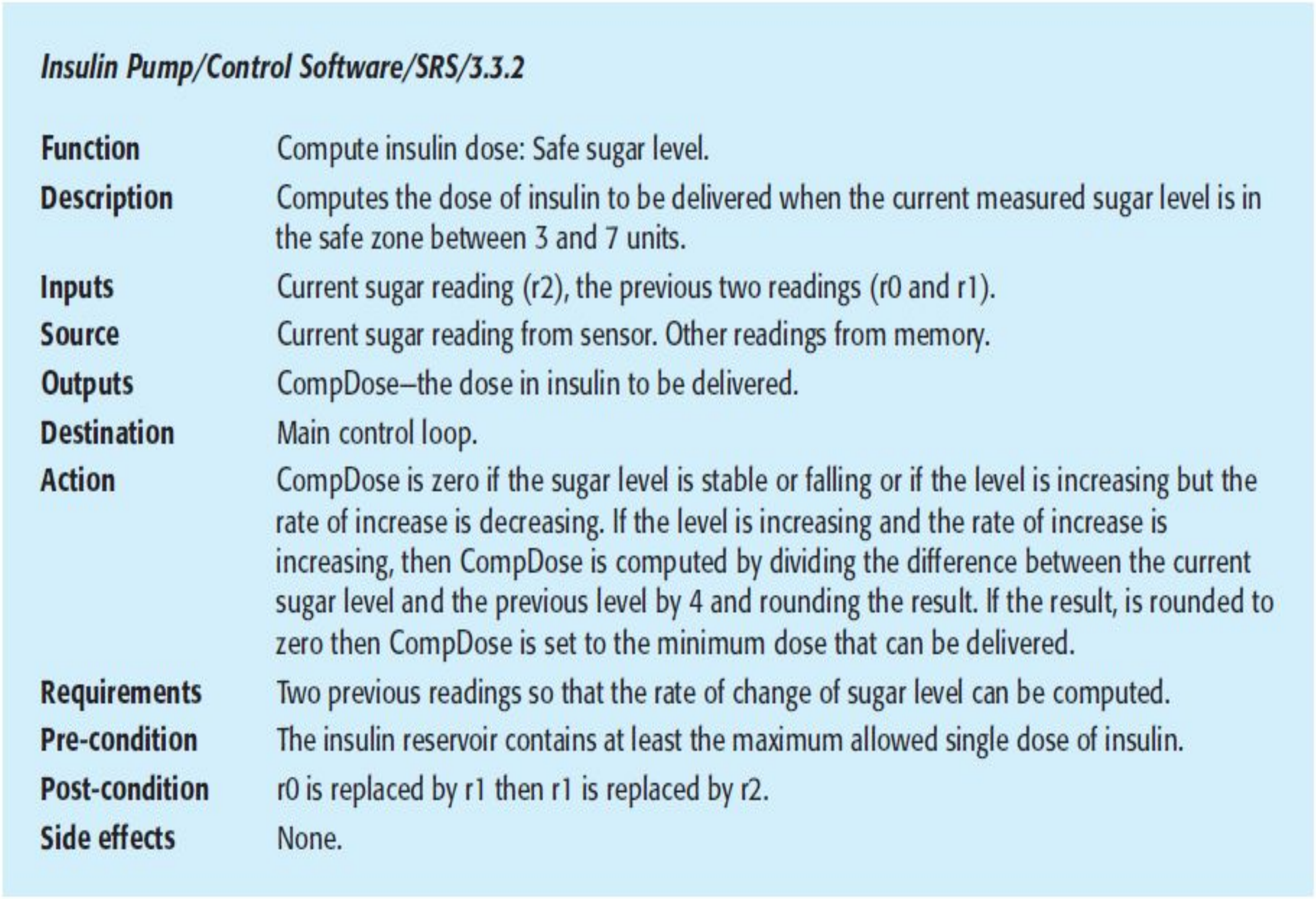
- This **works well for some types of requirements** e.g. requirements for **embedded control system** but is sometimes **too rigid** for **writing business system requirements.**

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.

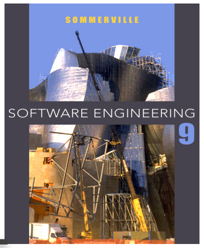
A structured specification of a requirement for an insulin pump



Insulin Pump/Control Software/SRS/3.3.2

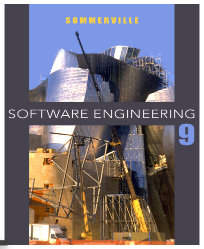
Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose–the dose in insulin to be delivered.
Destination	Main control loop.
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



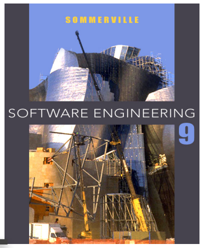
- **Used to supplement natural language**.
- Particularly useful **when you have to define a number of possible alternative courses of action.**
- For example, the ***insulin pump systems bases its computations on the rate of change of blood sugar level*** and the ***tabular specification explains how 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) ≥ (r1 – r0))	CompDose= round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

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

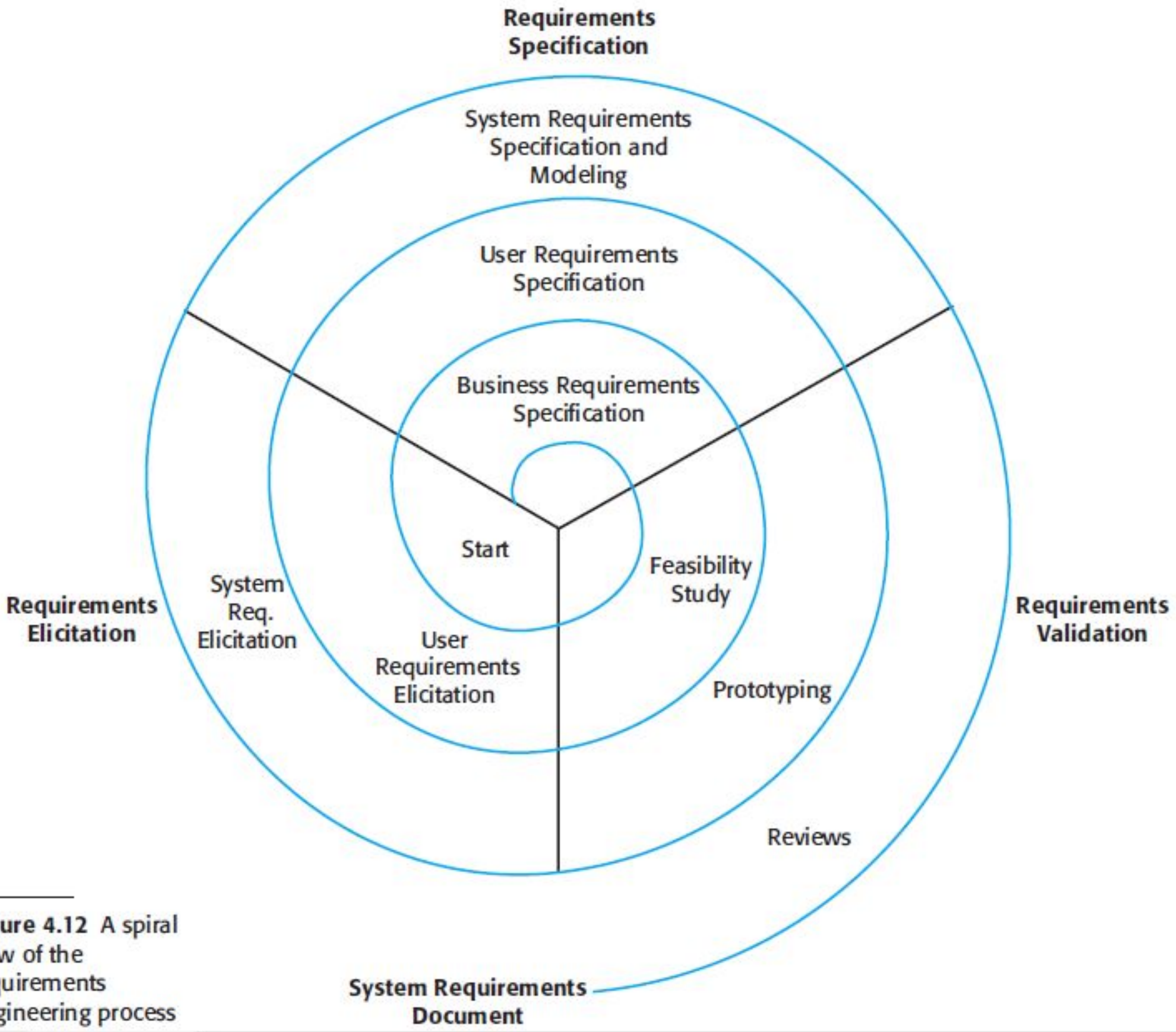
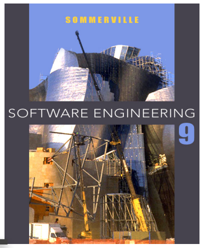

Requirements Specification
System Requirements Specification and Modeling
User Requirements Specification
Business Requirements Specification
Start
Feasibility Study
Requirements Elicitation
System Req. Elicitation
User Requirements Elicitation
Prototyping
Requirements Validation
Reviews
System Requirements Document

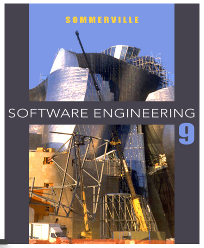
Figure 4.12 A spiral view of the requirements engineering process

Requirements elicitation and analysis



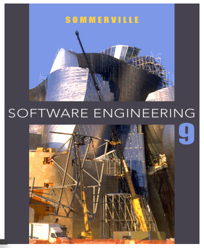
- Sometimes called ***requirements elicitation*** or ***requirements discovery.***
- Involves **technical staff working with customers** to ***find out about the application domain***, ***the services that the system should provide*** and the ***system's operational constraints.***
- May involve ***end-users***, ***managers, engineers*** involved in ***maintenance, domain experts, trade unions***, etc. These are called ***stakeholders.***

Problems of requirements analysis



- Stakeholders ***don't know what they really want.***
- Stakeholders ***express requirements in their own terms.***
- Different stakeholders *may have* ***conflicting requirements.***
- ***Organisational and political factors*** may ***influence*** the ***system requirements***.
- The ***requirements change during the analysis process***. ***New stakeholders may emerge*** and ***the business environment may change.***

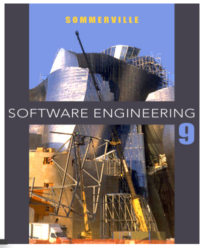
Requirements elicitation and analysis



- **Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.**

- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.

The requirements elicitation and analysis process



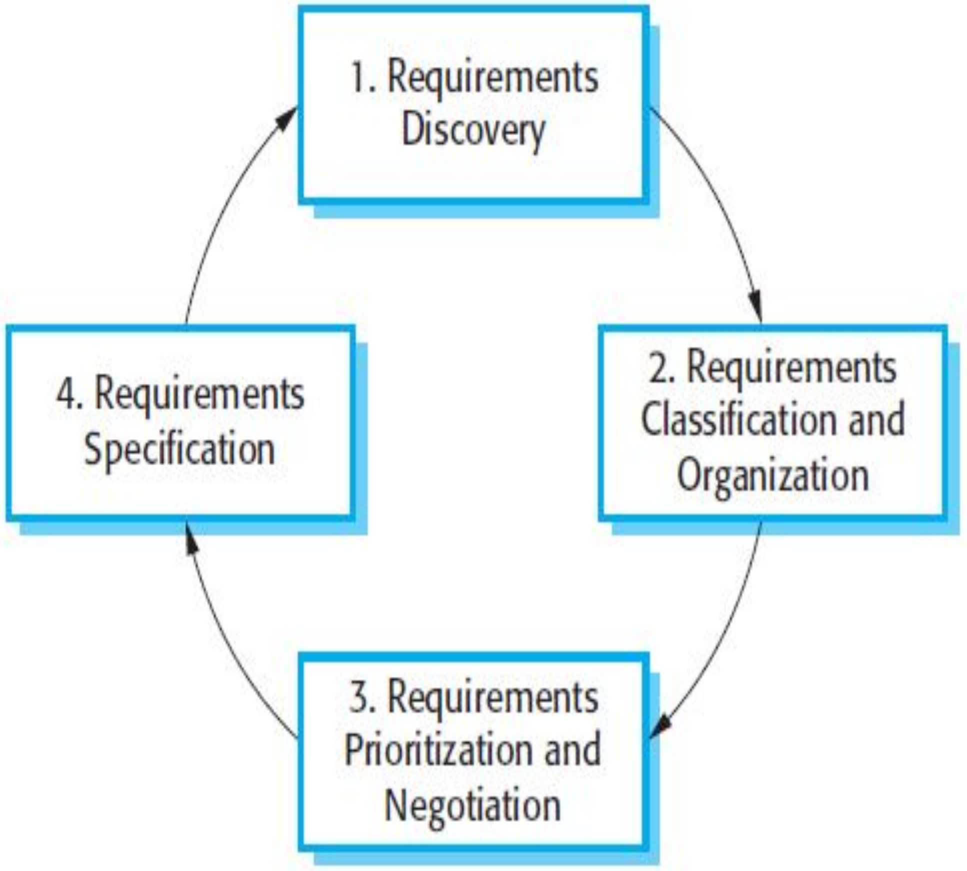
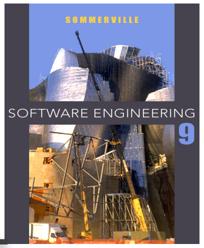

1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

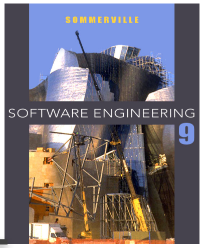
Figure 4.13 The requirements elicitation and analysis process

Process activities



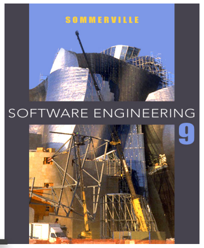
- **Requirements discovery**
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- **Requirements classification and organisation**
  - Groups related requirements and organises them into coherent clusters.
- **Prioritisation and negotiation**
  - Prioritising requirements and resolving requirements conflicts.
- **Requirements specification**
  - Requirements are documented and input into the next round of the spiral.

Problems of requirements elicitation



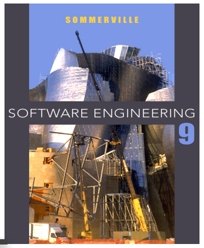
- Stakeholders ***don't know what they really want***.
- Stakeholders ***express requirements in their own terms.***
- Different stakeholders may have ***conflicting requirements***.
- **Organisational and political factors may influence** the ***system requirements.***
- The ***requirements change during the analysis process.*** **New stakeholders** may emerge and the **business environment change.**

Key points

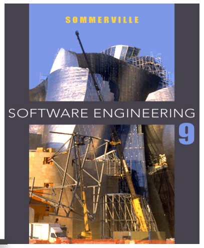


- The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
- The requirements engineering process is an iterative process including requirements elicitation, specification and validation.
- Requirements elicitation and analysis is an iterative process that can be represented as a spiral of activities – requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.

UNIT-II: REQUIREMENTS ENGINEERING
Review Questions



1. **Distinguish between functional and non-functional requirements with examples.**
2. **Explain the types of non-functional requirements with examples.**
3. **List and explain at least 03 functional and 02 Non-functional requirements for Vehicle Toll collection system using Fastag software.**
4. **Identify any 5 Functional and Non functional requirement for bank and Library System?**
5. **5. What are the different metrics for specifying non-functional requirements?**
6. **What is software requirements document, and who are the users of requirements document?**
7. **Explain in brief the structure of a requirements document that is based on an IEEE standard for requirements documents?**



8. Reliance industries is planning to start a cinema multiplexes across India. You have been asked to develop a Film ticket reservation system for their multiplexes online.
   List the system requirements using structured specification format for the same.

9. What is Requirements specification? What possible reasons where all the design information cannot be implemented?

10. What are the different ways of writing system requirements specification?

11. Explain in details the requirements engineering processes with an appropriate diagram?

12. Explain with a neat diagram the different steps in the requirements elicitation and analysis process?

13. What are the various reasons where Eliciting and understanding requirements process is difficult for stakeholders?