Software Process Requirements

Lecture 5

Requirements **Specification**

The process of writing down the user and system requirements in a requirements document.

- ✓ User requirements must be understandable by end-users and customers who do not have a technical background.
- ✓ System requirements are more detailed requirements and may include more technical information.
- ✓ The requirements may be part of a contract for the system development *(It is therefore important that these are as complete as possible)*.

Requirements **Specification**
Definition of a **Requirement Expression**

A Requirement Expression includes a requirement statement with a set of associated attributes.

[SH234] The ambulance control system shall be able to handle up to 100 simultaneous emergency calls.

Source: *R. Thomas*
Priority: *Mandatory*
Release: *1*
Review Status: *Accepted*
Verifiable: *Yes*
Verification: *By simulation, then by a system test.*

Requirements **Specification**

Ways of Writing

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 **Specification**

Ways of Writing [**Natural Language Spec.**]

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.*

Requirements Specification

Ways of Writing [Natural Language Spec.]

Problems with Natural Language:

1. **Lack of Clarity;** Precision is difficult without making the document difficult to read.
2. **Requirements Amalgamation;** Several different requirements may be expressed together.
3. **Requirements Confusion;** Functional and Non-Functional requirements tend to be mixed-up.

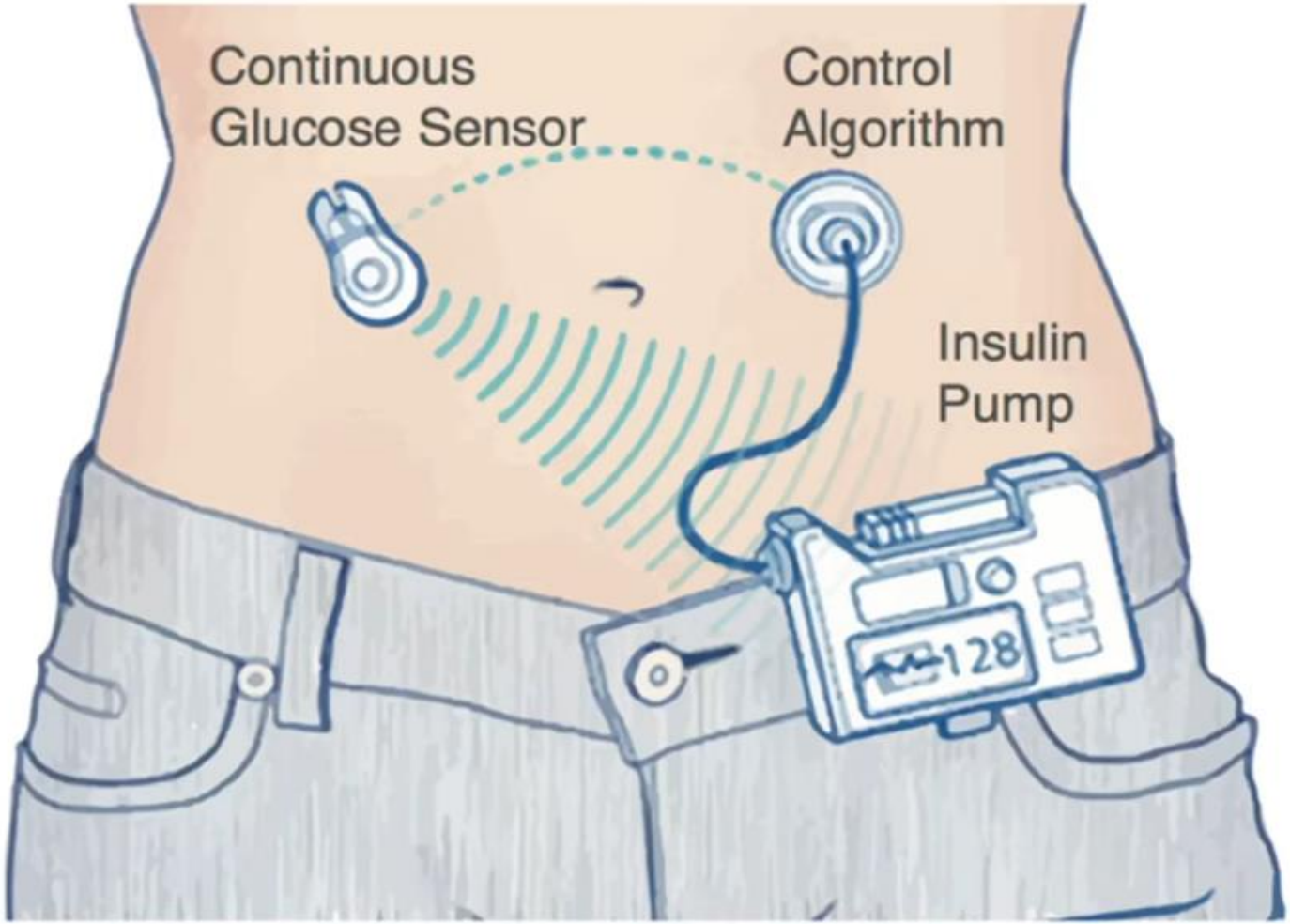
Guidelines for Writing Requirements:

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

Requirements **Specification**

Ways of Writing [**Natural Language Spec.**]

Example: *Requirements for the Insulin Pump Software System*



Requirements **Specification**

Ways of Writing [Natural Language Spec.]

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.)*

Requirements Specification

Ways of Writing [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 an embedded control system)*, but is sometimes too rigid for writing business system requirements.

Requirements Specification

Ways of Writing [Structured Specifications]

For Example: **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- conditions and post-conditions *(if appropriate)*.
- The side effects *(if any)* of the function.

Requirements Specification

Ways of Writing [Structured Specifications]

Example: *Requirements for the Insulin Pump Software System*

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.

Requirements Specification

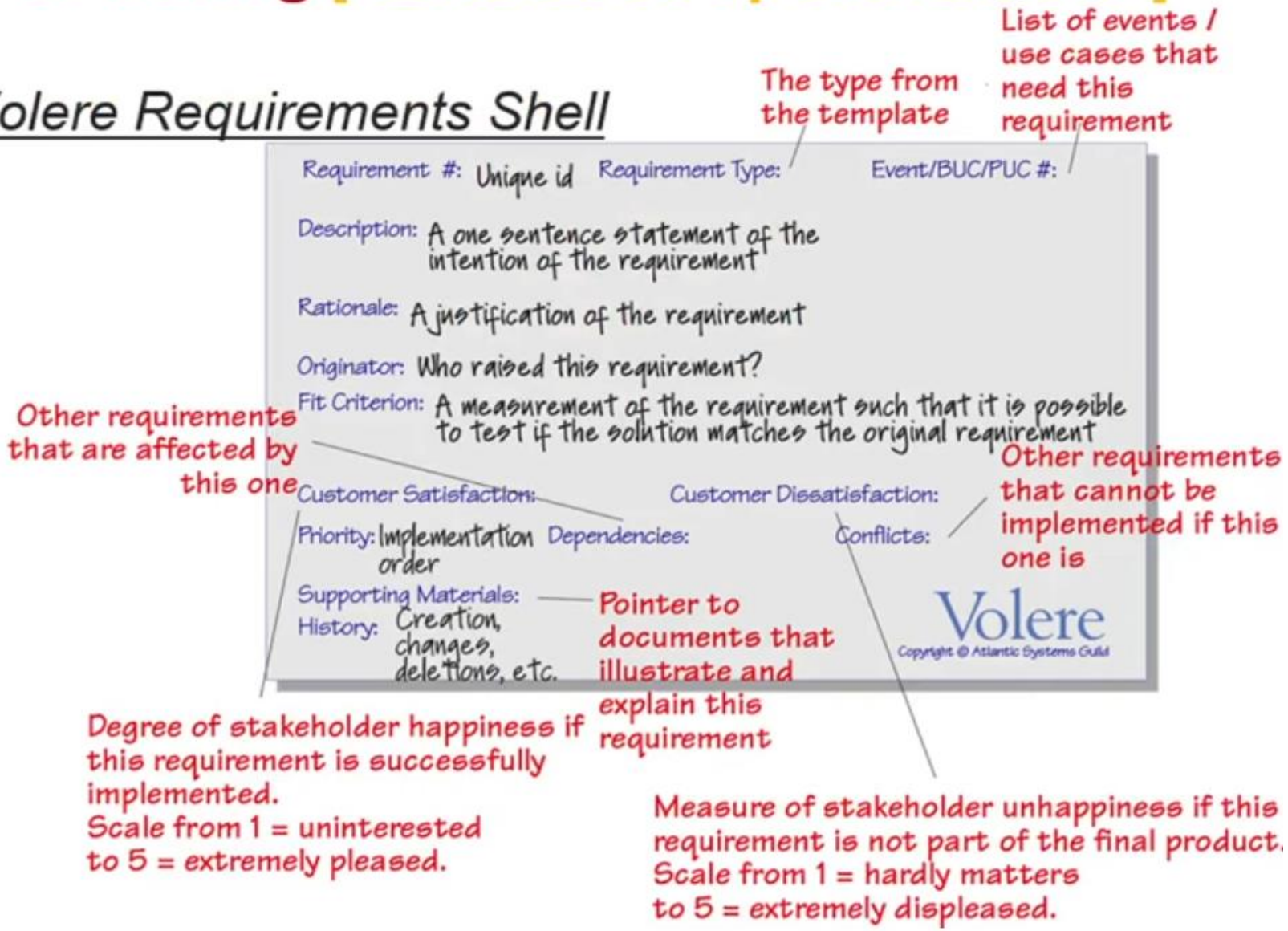
Ways of Writing [Structured Specifications]

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.

Requirements **Specification**

Ways of Writing [**Structured Specifications**]

E.g., Volere Requirements Shell



Requirements Specification

Ways of Writing [Tabular Specification]

Used to supplement natural language.

Particularly useful when you must define several 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.

Requirements Specification

Ways of Writing [Tabular Specification]

Example: *Requirements for the Insulin Pump Software System*

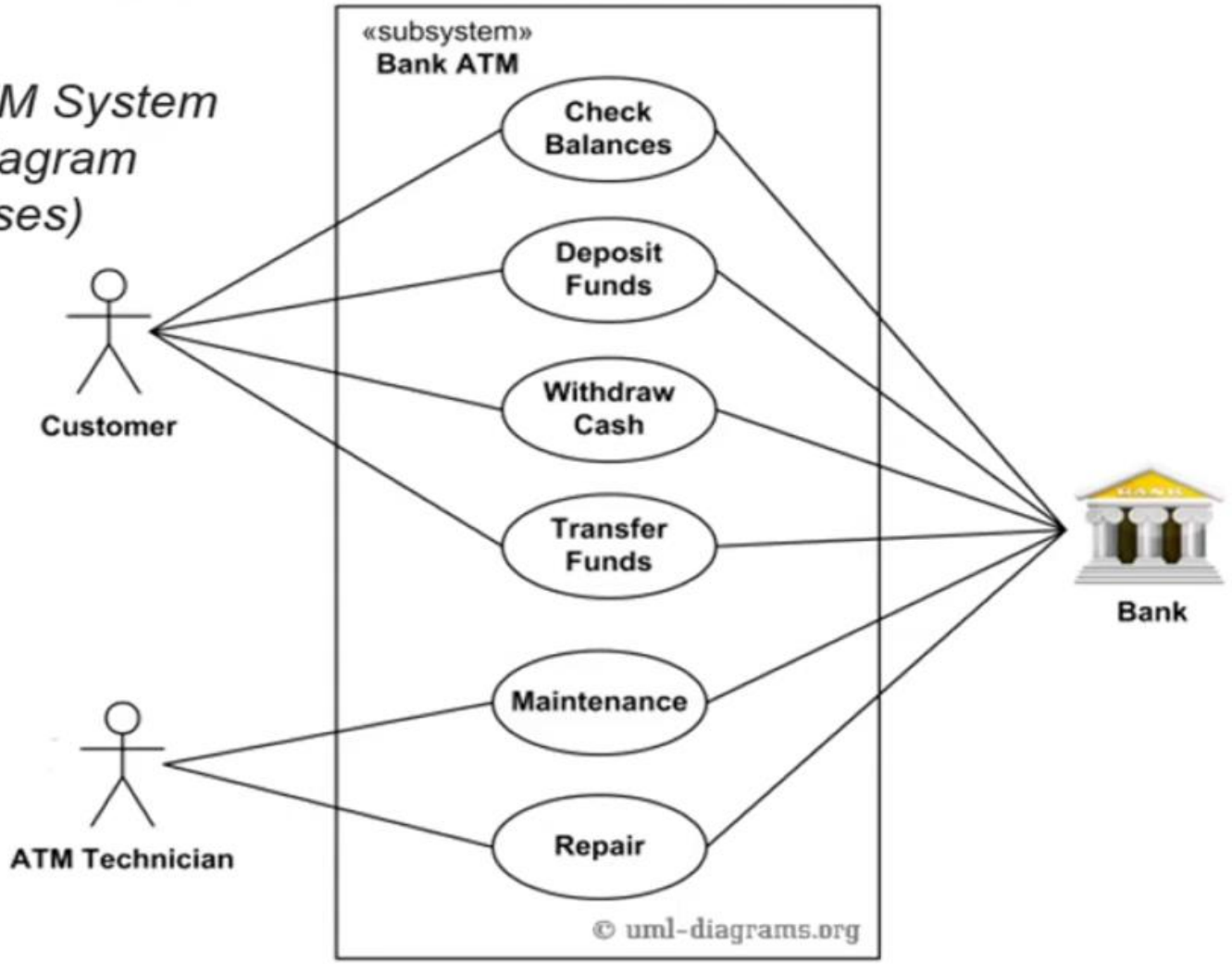
Computation for the Insulin Pump:

Condition	Action
Sugar level falling *(r2 < r1)*	CompDose = *0*
Sugar level stable *(r2 = r1)*	CompDose = *0*
Sugar level increasing & rate of increase decreasing *((r2 – r1) < (r1 – r0))*	CompDose = *0*
Sugar level increasing & rate of increase stable or increasing *((r2 – r1) ≥ (r1 – r0))*	CompDose = *round ((r2 – r1) / 4)* If rounded result = *0* then *CompDose = MinimumDose*

Requirements Specification

Ways of Writing [Graphical Models]

Example: *an ATM System – A Use Case Diagram (top level use cases)*



The **Software Requirements Document**

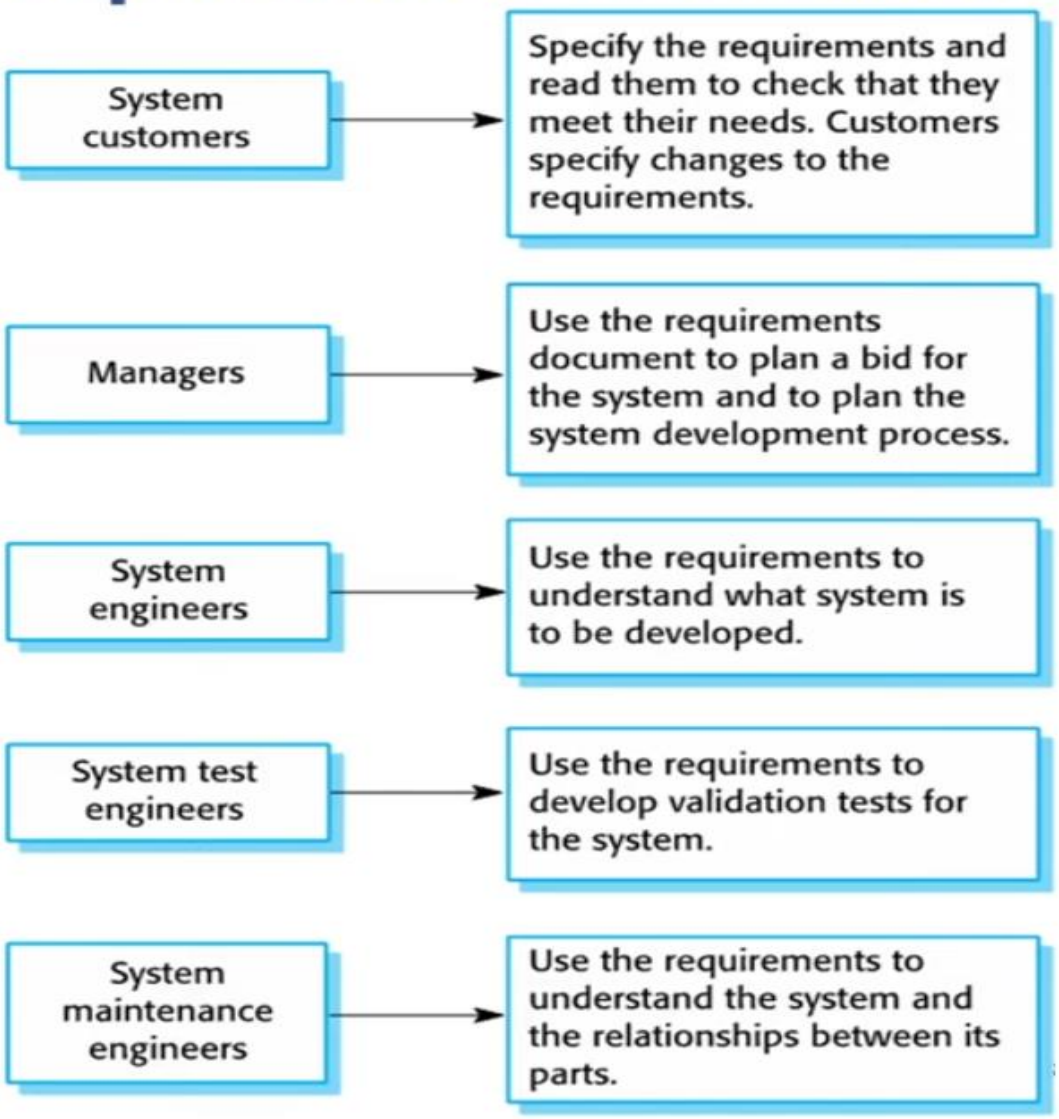
The software requirements document is the official statement of what is required of the system developers.

- 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 set of **WHAT the system should do** *rather than HOW it should do it*.

Note: Information in Requirements Document **depends on the type of system and the approach to development used** *(for instance, systems developed incrementally will, typically, have less detail in the requirements document).*

The **Software Requirements Document**

Typical **Users** of a Requirements Document:



The **Software Requirements Document**

Typical **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 **Software Requirements Document**

Typical **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.

The **Software Requirements Document**

Examples of a Requirements Document:

► **Requirements for the MentCare System** *(a system to support the clinical management of patients suffering from mental illness)*:
https://iansommerville.com/software-engineering-book/case-studies/mentcare/

► **Software Requirements Specification for a Web Publishing System:**
https://www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc

► **A Personal Insulin Pump:**

- **Case Study details:** https://iansommerville.com/software-engineering-book/case-studies/insulin-pump/
- **Insulin Pump – an Overview:** https://www.dropbox.com/s/tzc8shdjmrqo4cz/InsulinPumpOverview.pdf?dl=0
- **Requirements for the Insulin Pump:** https://www.dropbox.com/s/grgaaxtdas4oj1i/InsulinPumpRequirements.pdf?dl=0

The **Software Requirements Document**

Examples of a Requirements Document:

► **Flight Control System (Airbus 340);** *System Specification that describes the architecture of the Airbus 340 flight control system, a safety critical system that implements the fly-by-wire flight system on the Airbus.*

- **Flight Control System (Airbus 340) - Overview:** https://www.slideshare.net/software-engineering-book/airbus-fcs-42647819
- **Flight Control System (Airbus 340) - System Specification [Part 1]:** https://www.dropbox.com/s/o6d7056eh3bvv9u/FCS1.pdf?dl=0
- **Flight Control System (Airbus 340) - System Specification [Part 2]:** https://www.dropbox.com/s/jbe0ajlhxotvqn2/FCS2.pdf?dl=0

.. + More Examples/Samples in the description below the video.

Determine:

- The system stakeholders (5 Marks)
- The functional requirements (write one of them using the natural language) (5 Marks)
- The nonfunctional requirements (5 Marks)
- The system constrains (5 Marks)
- The use case diagram (5 Marks)
- Write the requirements specification using form- based specifications model(5 Marks)

❖ Definition of the function or entity
❖ Description of inputs and where they come from
❖ Description of outputs and where they go to
❖ Indication of other entities required
❖ Pre and post conditions (if appropriate)
❖ The side effects (if any)