

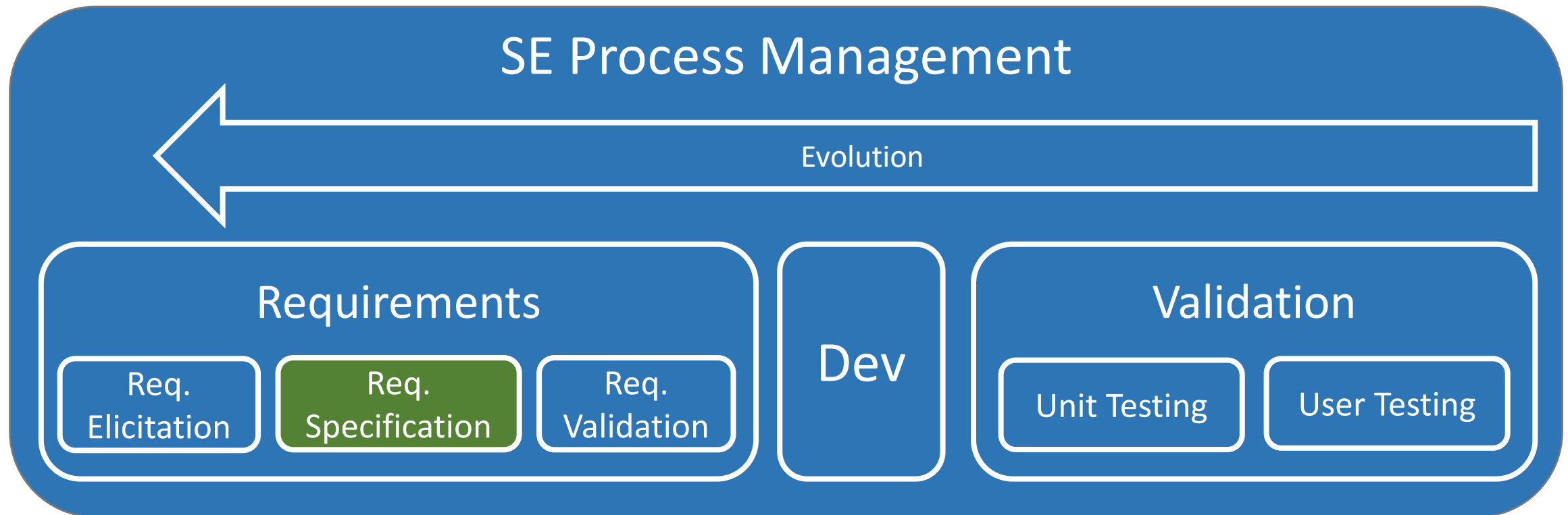
Software Engineering COMP1035

Lecture 06

Requirements Specification



Keeping Track of SE Module

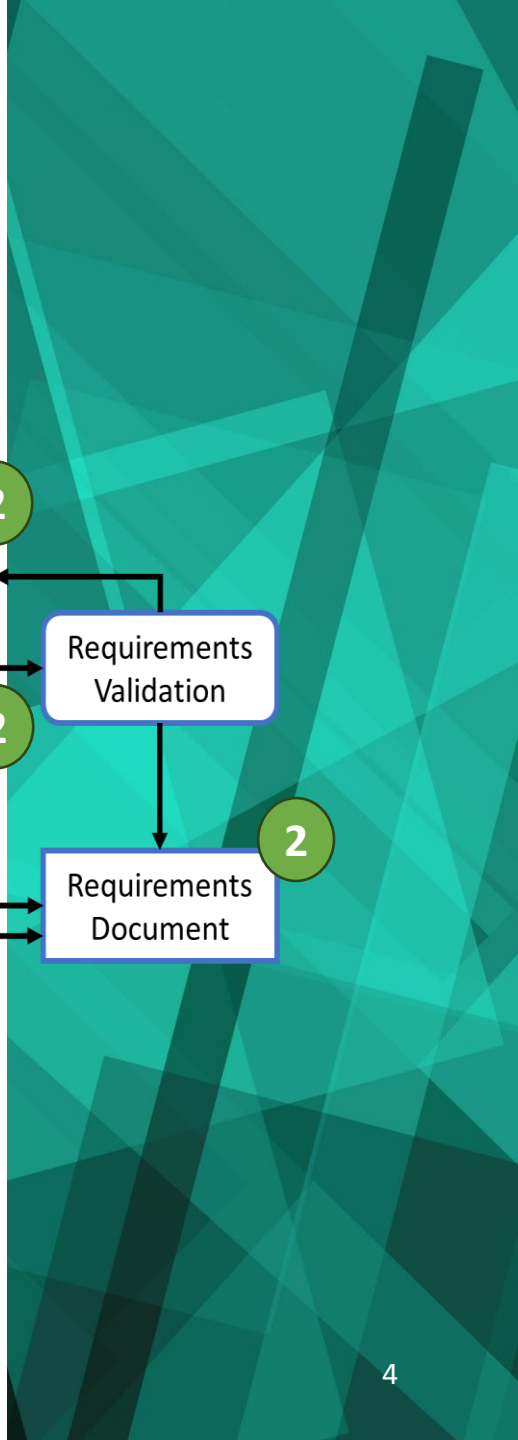




Today's Learning:

1. What requirements specification are (and what they look like)?
2. Difference between user and system requirements?
3. What requirements document is and who uses them?

© 2006 The Authors
Journal compilation © 2006 Blackwell Publishing Ltd



What is Requirements Specification?

- Is the **process of writing down the user and system requirements** in a requirements document.
- Should be clear, unambiguous, easy to understand, complete, and consistent.
- User requirements:
 - Written in **natural language**, supplemented with diagrams and tables in the requirements document.
- System requirements:
 - May also be written in natural language
 - Other **notations** such as forms, graphical or mathematical system models

Notations for Writing System Requirements (Specifications)

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.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML (unified modeling language) 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 they are reluctant to accept it as a system contract. (I discuss this approach, in Chapter 10, which covers system dependability.)

User Requirements

- User requirements should **describe the functional and non-functional requirements** – for stakeholders without technical knowledge.
 - Specify only **external behavior of the system**
 - Should not include details of the system architecture or design
 - Write in natural language:
 - Simple tables, forms and diagrams.

System Requirements

“... detailed descriptions of the software system’s functions, services and operational constraints ... they ... should define **exactly what is to be implemented.**”

- A focus on **WHAT** should be built – but not necessarily HOW.
- Specifying: What a system will do to meet the user requirements.
- This means system requirements should be tied to a user requirement.
 - System requirements are often tabulated.
 - With a column saying which requirement ID it is for, etc.
 - Priorities, risks, etc.

User and System Requirements

Something the stakeholder will need – or a service the software must provide.

What the system must do to meet this user 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.

Natural Language Specification



Natural Language Specification

- Can be **expressive**, **intuitive** and **universal**.
- Can also be ambiguous, vague, and interpreted differently.
- Guidelines
 1. Use a standard format: one or two sentences to link to a user requirement.
 2. Distinguish between mandatory (“shall”) and desirable (“should”).
 3. Emphasis important key parts of the requirement with bold, italic etc.
 4. Avoid jargon, unless clearly specified in a key words section.
 5. Try to associate a rationale (why included?) with each user requirement.
 6. Make sure the **requirement is measurable** in some form
 - Can’t be partly achieved or has a metric for successfully achieved.

Natural Language Specification

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.*)

Fig: Example Requirements for the Insulin Pump Software System

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#).

Note that the force of these words is modified by the requirement level of the document in which they are used.

- 1. MUST** This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- 2. MUST NOT** This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- 3. SHOULD** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- 4. SHOULD NOT** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

Natural Language Specification

From April 1999 – one project defining their terms.
<https://tools.ietf.org/html/rfc2119>

Structured Specifications



Structured Specifications

- Go further than natural language specifications, to tabulate requirements (in **exact** way), or put them in templates, rather than as free-form text.
- Define one or more **standard templates for requirements** – used as structured forms.
- Following **information should be included in a standard format** of specifying functional requirements:
 - Description of the function being specified.
 - Description of its inputs and origin of these inputs.
 - Description of its outputs and the destination of these outputs.
 - Information needed for the computation that are required (the “required” part).
 - Description of action to be taken.
 - Precondition and postcondition of a functional requirement.
 - Description of the side effects (if any).

Structured Specifications

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.*)

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. (see Figure 4.14)
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Precondition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Postcondition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

Fig: The Structured Specification of a Requirement for an Insulin Pump

Structured Specifications

- However – sometimes difficult to write requirements in a clear and unambiguous way, especially when dealing with complex computations (e.g., how to calculate insulin dose).
- When a requirement has options that need explicitly listing.
- Can **add extra information using tables or graphical models**.

Condition	Action
Sugar level falling ($r_2 < r_1$)	CompDose = 0
Sugar level stable ($r_2 = r_1$)	CompDose = 0
Sugar level increasing and rate of increase decreasing ($(r_2 - r_1) < (r_1 - r_0)$)	CompDose = 0
Sugar level increasing and rate of increase stable or increasing $r_2 > r_1$ & $(r_2 - r_1) \geq (r_1 - r_0)$	CompDose = round $((r_2 - r_1)/4)$ If rounded result = 0 then CompDose = MinimumDose

Fig: The Tabular Specification of Computation in an Insulin Pump

Graphical Notations

- **UML models**, diagrams, prototypes.
 - At this stage, though, the emphasis is on saying.
 - "this is how it will work".
 - "this is what you "mr. developer" should build".
- It's often easier to see a UML sequence diagram.
 - Than to read 30 decision-dependent specifications.
- So, when you find specifications are complex – visualize them.
 - Just as when you found a requirement to be complex.
- UML can be used for either.
 - And indeed, for post-development documentation (as said a few lectures ago).

Software Requirements Document



Software Requirements Document

- Also known as software requirements specification or SRS.
- Include **both user and system requirements** for a system.
- IEEE standard 830-1998.

Table of Contents for a SRS Document

1. Introduction

- 1.1 Purpose
- 1.2 Document Conventions
- 1.3 Intended Audience and Reading Suggestions
- 1.4 Project Scope
- 1.5 References

2. Overall Description

- 2.1 Product Perspective
- 2.2 Product Features
- 2.3 User Classes and Characteristics
- 2.4 Operating Environment
- 2.5 Design and Implementation Constraints
- 2.6 Assumptions and Dependencies

3. System Features

- 3.1 Functional Requirements

4. External Interface Requirements

- 4.1 User Interfaces
- 4.2 Hardware Interfaces
- 4.3 Software Interfaces
- 4.4 Communications Interfaces

5. Nonfunctional Requirements

- 5.1 Performance Requirements
- 5.2 Safety Requirements
- 5.3 Security Requirements
- 5.4 Software Quality Attributes

SRS Document

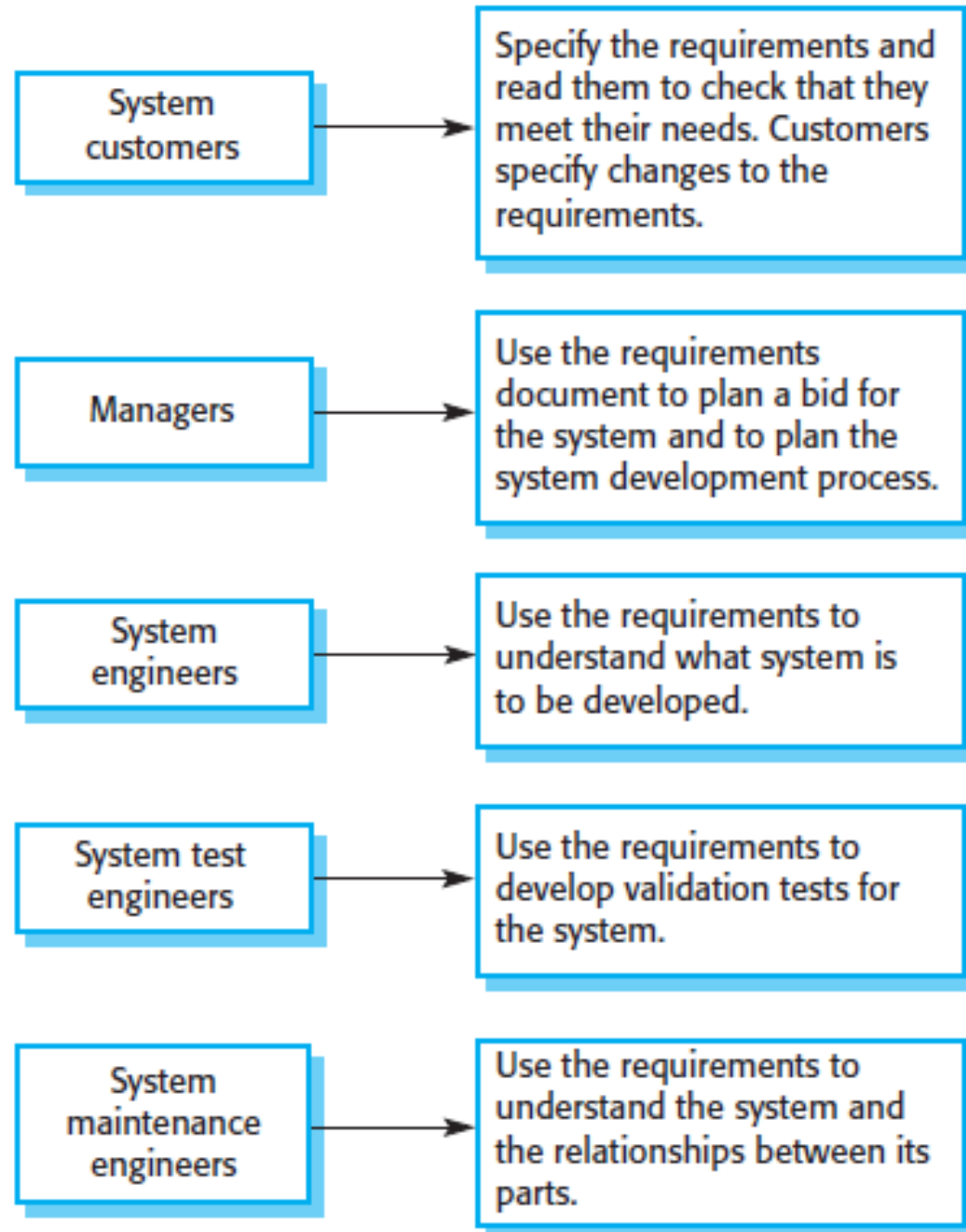
- Translation from anecdotal user wishes to a formal document.
- Usually, **a big set of nested lists, with unique IDs.**
- Diagrams produced go with them (and cross reference those IDs).
- Lists should be **categorised** – e.g., importance, risk etc.
- Usually define the “acceptance testing” at the end of the project.

SRS Document

ID	ID	Description	Importance
F1	1	First requirement description	High
F2	2	Second requirement description	High
NF1	3	Third requirement description	Medium
F3	4	Fourth Requirement description	Low
NF2	5	Fifth requirement description	High
F4	6	Sixth Requirement description	High

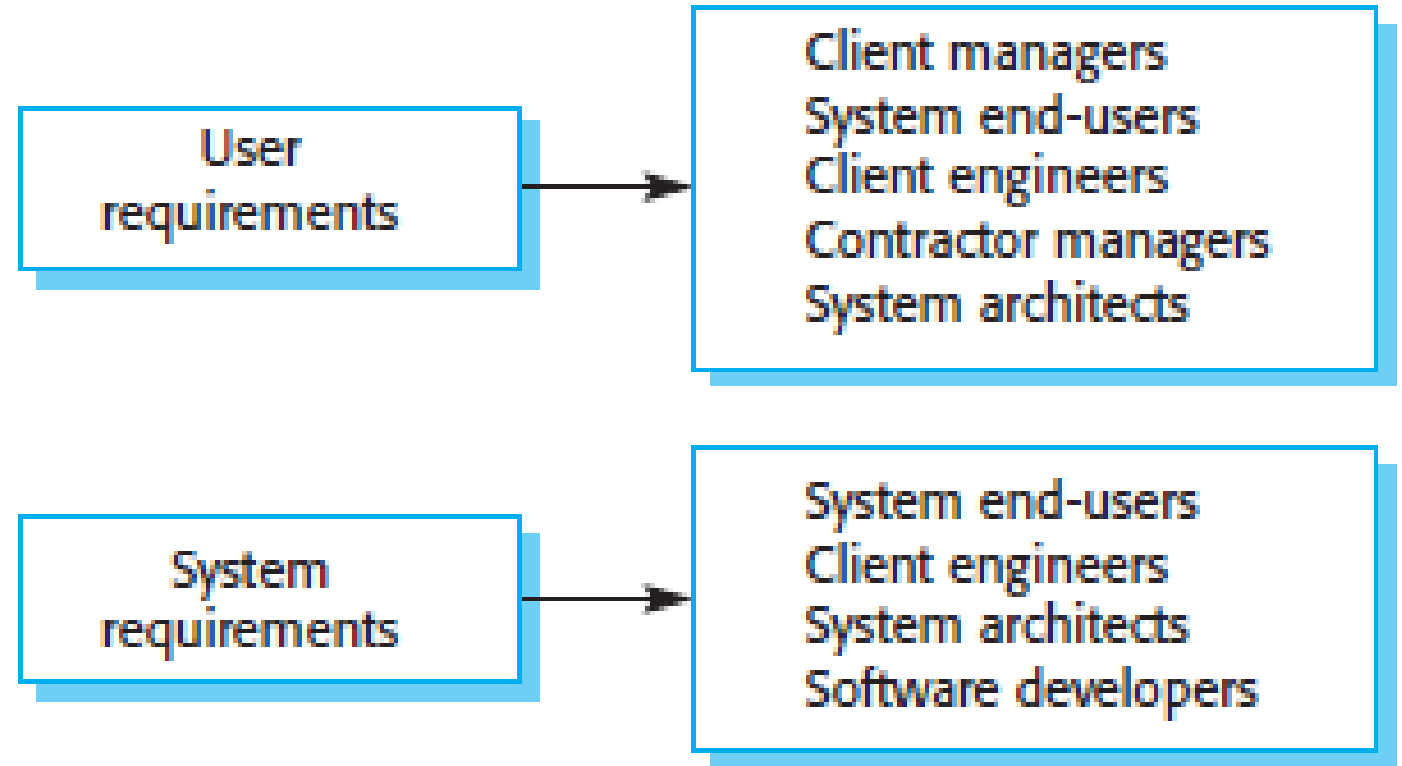
Users of a Requirements Document

- Describe requirements for customers.
- Define requirements for developers and testers.
- May include information about future system evolution.



Readers of Different Types of Requirements Specification

Different system stakeholders have some kind of interest in the system.





Structure of a Requirements Document

Sample SRS document based on an IEEE standard for SRS (IEEE 1998).

Chapter	Description
Preface	This defines 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 describes 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 defines 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 presents 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.
System requirements specification	This describes 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 chapter includes 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 describes 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 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 Document Standards


- Information included in the SRS depends on the type of software being developed (approach).
- SRS is likely to be long and detailed, with comprehensive table of contents and document index (for easy search).



Requirements document standards

A number of large organizations, such as the U.S. Department of Defense and the IEEE, have defined standards for requirements documents. These are usually very generic but are nevertheless useful as a basis for developing more detailed organizational standards. The U.S. Institute of Electrical and Electronic Engineers (IEEE) is one of the best-known standards providers, and they have developed a standard for the structure of requirements documents. This standard is most appropriate for systems such as military command and control systems that have a long lifetime and are usually developed by a group of organizations.

<http://software-engineering-book.com/web/requirements-standard/>



Writing Good System Requirements

Qualities of Good System Requirements

- Analysing the quality of system requirements is also important.
- Good quality system requirements have a few qualities.
- Key: **3C principles**

Correct	Complete
Necessary	Clear
Feasible	Consistent
Unique	Traceable
Concise	Verifiable

Requirements Specification – Traceability

- It's important that all system requirements **can be traced to user requirements**.
- In the SRS document, you should, for every system requirement:
 - State which user requirement(s) it is supporting.
- You may also categorise them by importance, difficulty etc.

Requirements Specification – Testability

- **Verification** – did we build it right? (Will discuss in the next lecture!)
- To have implemented a single system requirement, you'll want to know you achieved what it specified.
 - You can only do this if system requirements are testable.
- Can all the things you specify be tested and proved?
 - 'It must load fast' – unprovable.
 - 'it must load within 10s' – testable and provable.

Specifications – A Stage in the Process

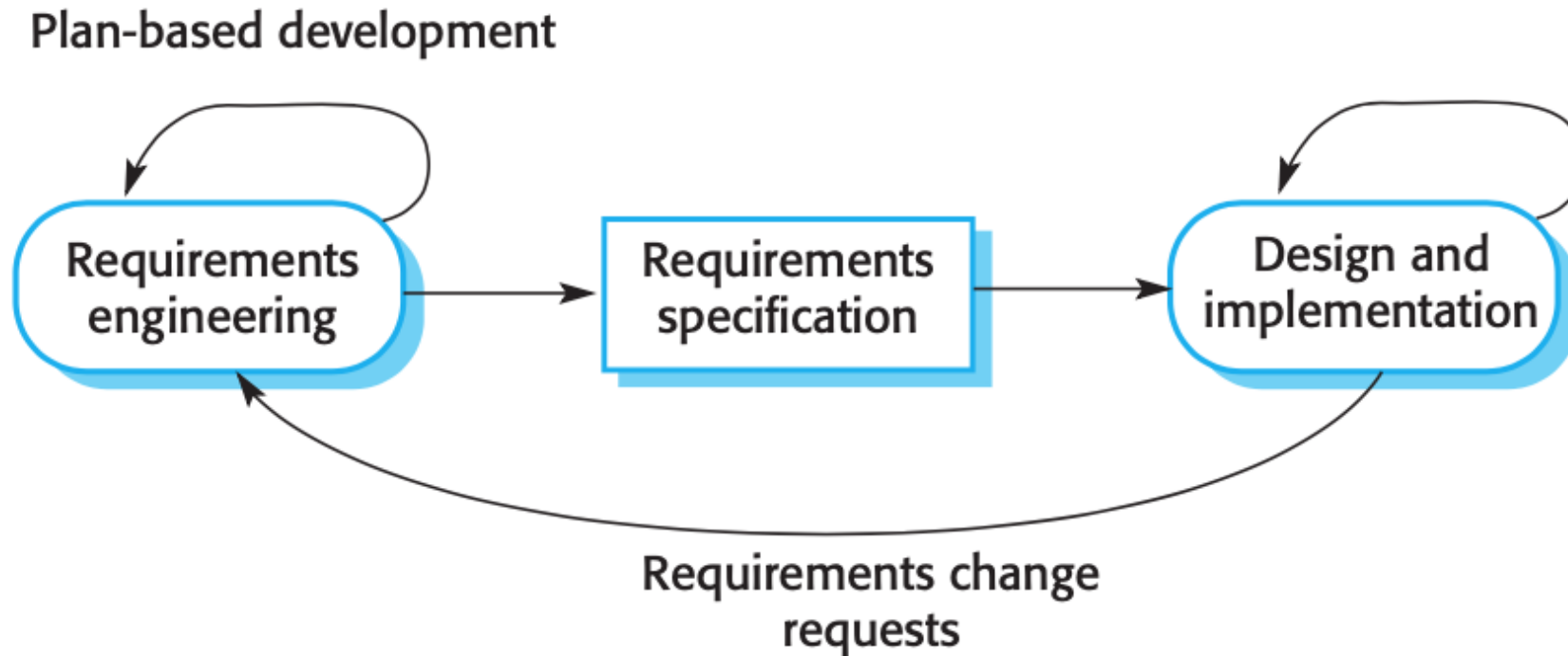


Figure 3.1 Plan-driven and agile development

Summary

“... it’s **difficult to show that a set of [requirements] does in fact meet a user’s need**. Users need to picture the system in operation and imagine how that system would fit into their work.”

- Can a client/user understand a system requirement?
 - Probably not, and they aren’t intended to see them.
- But we do need to show clients the plans for what we are building.
- SRS document combines:
 - The graphical notation – UML models (also scenarios etc. from User Requirements).
 - The textual writing – In the System Requirements Specifications.
 - To demonstrate how they might work together in a system.

Expected Readings



- How to write good specifications?
 - <https://www.joelonsoftware.com/2000/10/15/painless-functional-specifications-part-4-tips/>

Optional Readings

- NASA – Prototyping Software?
 - <https://www.justinmind.com/learn-ux-design/nasa-ux-design-ron-kim>



