Faculty of ECMS / School of Computer Science

Software Engineering & Project
Requirements Engineering (Cont.)

Requirements Engineering (Cont.)

Lecture 4

Chapters 6, 7 (4 in Edition 9)
in the course text book

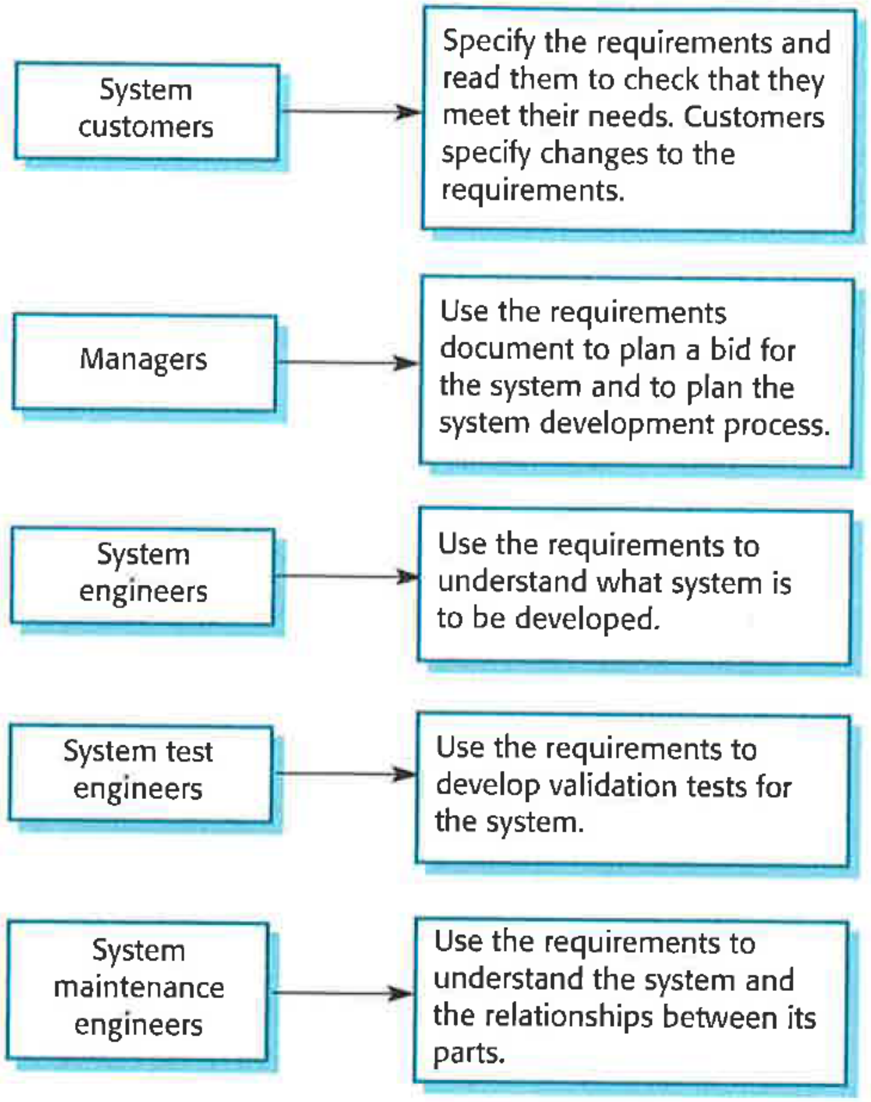
Outline

- Overview of Requirement Engineering
- Types of Requirements
- Techniques for Requirements Elicitation
- Requirements Documentation
- Requirements Management

Requirements Documentation

- The software requirements document is the official statement of what is required of the system developers.
- The document is normally called Software Requirements Specification (SRS)
- Should include both a definition of user requirements and a specification of the system requirements.
- It is *not* a design document. It should be *what* the system should do rather than *how* it should do it.

Users of 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.

Characteristics of A Good Requirements Document

- **Unambiguous**: Every requirement stated in the SRS has only one interpretation.
- **Complete**: All the requirements in the SRS are defined, referenced, labelled and compliant with the SRS standard defined.
- **Consistent**: No requirements in the SRS are in conflict with each other.
- **Verifiable**: All the requirements in the SRS are verifiable (testable) against a predefined checklist or international standards.
- **Modifiable**: The structure and style of the SRS are such that any necessary requirements changes can be made easily, completely, and consistently.
- **Traceable**: The origin and rationale of each requirement in the SRS is stated and the links between requirements, design, implementation and test cases are provided.

Characteristics of A Good Requirements Document (Cont.)

- The use of "shall" statements is encouraged, as this implies a directive to express what is mandatory.
  - The use of "shall" requires a sense of commitment
- All the requirements shall not be defined with ambiguous English
  - “sort of”, “kind of”, “similar”, “more or less” shall not be used
  - The use of “will”, “may”, “should” lead to vagueness
    - “will" statements imply a desire, wish, intent or declaration of purpose.
    - “should" or "may" are used to express non-mandatory provisions.
  - It is recommended that "will, should and may" not be used in writing requirements unless absolutely necessary.

Characteristics of A Good Requirements Document (Cont.)

- 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.
- Think about who is going to be reading the document.

Requirements: examples

- **R001 - Register patient**
  The PRS shall allow designated staff to register new patients.
- **R002 - Assign ID**
  The PRS shall allow the designated staff to assign a unique ID to each patient's record that shall be used throughout their visit in the hospital.
- **R003 - Allocate bed**
  The PRS shall allow the designated staff to allocate an available bed to patient.
- **R004 - Personal Information**
  The PRS shall store the patients id, first name, last name, doctors name, ward number, bed number, expected visit length.

- And so on.

Structure of Requirement Documents

- Requirements documents standards have been designed. These are mostly applicable to the requirements for large systems engineering projects
  - IEEE standard (IEEE/ANSI 830-1998)
    - IEEE830.pdf, available at course website
  - US Department of Defense

- The company you are going to work may have its own standard.

Structure of Requirement Documents (Cont.)

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.

Structure of Requirement Documents (Cont.)

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.

Outline

- Overview of Requirement Engineering
- Types of Requirements
- Techniques for Requirements Elicitation
- Requirements Documentation
- Requirements Management

Requirements Management

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- New requirements emerge as a system is being developed and after it has gone into use.
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes. You need to establish a formal process for making change proposals and linking these to system requirements.
- Changing requirements (at later stages of software development) can be expensive!!!
  - Changing a requirement may cost 10 times more than fixing a bug

Changing Requirements

- The business and technical environment of the system always changes after installation.
  - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.
- The people who pay for a system and the users of that system are rarely the same people.
  - System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

Changing Requirements (Cont.)

- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
  - The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

Requirements Change Management

- Deciding if a requirements change should be accepted
  - *Problem analysis and change specification*
    - During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.
  - *Change analysis and costing*
    - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.
  - Change implementation
    - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Relevance to Your Project

- What techniques to use in order to get requirements?
- How to analyze and document your requirements?

Key points revisited

- 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.

Key points revisited (Cont.)

- 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.

Key points revisited (Cont.)

- You can use a range of techniques for requirements elicitation including interviews, scenarios, use-cases, prototyping, and ethnography.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.