Chapter 4 – Requirements Engineering 4.2 – 4.4

Topics covered

4.2 The software requirements document

✧ Official statement of what the system developers should implement.

✧ Should include
- user requirements
- system requirements

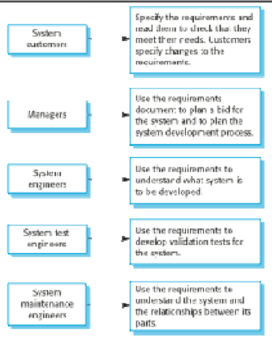
4.2 The software requirements document

✧ Official statement of what the system developers should implement.

✧ Should include
- user requirements
- system requirements

✧ NOT a design document
- **WHAT** the system should do not **HOW** it should do it.

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.

Requirements document

✧ Level of detail depends on

	Less detail	More detail
type of system	e.g. Games	e.g. Critical System
approach to development	Agile development	Traditional development
Location	local	distributed
Project Size	Small to medium	Large

Requirements document standards

✧ Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects with long lifetime.

Structure of a requirements document

✧ **List of Chapters:**

- Introduction
- Glossary
- **User Requirements Definition**
- System Architecture
- **System Requirements Specification**
- System Models
- System Evolution
- Appendices
- Index

4.3 Requirements specification

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

✧ User requirements have to be understood by managers, end-users and customers who have no technical background.

- natural language
- simple tables / forms
- intuitive diagrams

4.3 Requirements specification

✧ System requirements:

- add detail and explain how user requirements should be provided by the system
- Should describe external behavior not design information

4.3 Requirements specification

✧ System requirements:

- add detail and explain how user requirements should be provided by the system
- Should describe external behavior not design information

In practice, requirements and design can't be completely separated.

What not how …

✧ Some initial design is needed to

- to structure requirements specification (important if reuse of software component is planned)
- To interoperate with existing system
- Specific architecture to satisfy non-functional requirements

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

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

4.3.1. Natural language specification

- Natural language supplemented with diagrams and tables.
- Pro:
  - Expressive, intuitive, universal.
  - Can be understood by users and customers.

4.3.1. Natural language specification

- Natural language supplemented with diagrams and tables.
- Pro:
  - Expressive, intuitive, universal.
  - Can be understood by users and customers.
- Con:
  - Lack of clarity
  - Requirements confusion (functional / non-functional)
  - Several Requirements may be expressed together.

Guidelines for writing requirements

- Invent a standard format and use it for all requirements.
- Use language in a consistent way.

  e.g. shall for mandatory, should for desirable requirements
- Text highlighting to identify key parts
- Avoid jargon, acronyms, ..
- Include rationale why requirement is necessary.

Example requirements for the insulin pump software system (Fig 4.9)

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.)*

1.3.2 Structured natural language specifications

✧ Requirements written in standard way using templates (e.g. One card per user requirement)

✧ Works well for some types of requirements e.g. requirements for embedded control system
Can be too rigid for business system requirements.

Information included in template:

✧ Definition of the function or entity.
✧ Inputs (source)
✧ Outputs (destination)
✧ Information about information needed for computation
✧ Action to be taken.
✧ Pre and post conditions (if appropriate)
✧ Side effects (if any) of the function

A structured specification of a requirement for an insulin pump (Fig 4.10)

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.

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 defining a number of possible alternative courses of action.

Tabular specification of computation for an insulin pump (Fig 4.11)

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

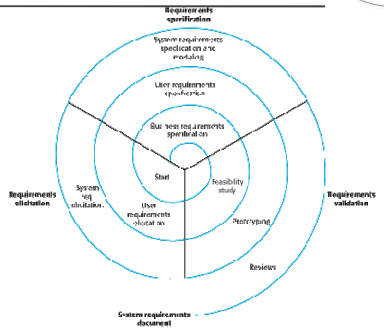
4.4 Requirements engineering processes

✧ Vary widely depending on application domain, people involved and the organisation developing the requirements.

✧ Still, certain generic activities are common to all processes
- Requirements feasibility
- Requirements elicitation
- Requirements analysis + specification
- Requirements validation
- Requirements management

✧ 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
Reviews
Requirements validation
System requirements document

Exercise

✧ List 2 user requirements of your team project

✧ Below each user requirement list the corresponding system requirements

✧ Use numbered sentences, consistent user of language

…

4.5 Requirements elicitation and analysis

✧ Technical staff working with stakeholders to find out
1. about the application domain,
2. services that the system should provide
3. system's operational constraints.

4.5 Requirements elicitation and analysis

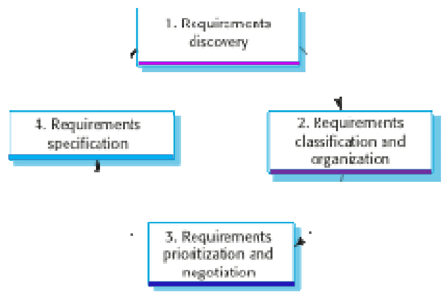
✧ Technical staff working with stakeholders to find out
1. about the application domain,
2. services that the system should provide
3. system's operational constraints.

✧ Who are stakeholders?
end-users, managers, engineers involved in maintenance, domain experts, trade unions, regulators, etc.

Problems of requirements analysis

✧ Stakeholders don't know what they really want
✧ Express requirements in own terms
✧ Conflicting requirements
✧ Organisational and political factors may influence the system requirements
✧ The requirements change during the analysis process
✧ New stakeholders emerge
✧ Business environment changes

The requirements elicitation and analysis process


1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirements specification

1) Requirements discovery

✧ The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.

1) Requirements discovery

✧ The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.

Sources:

✧ Stakeholders

✧ Documentation, spec of similar systems

✧ Domain requirements, . .

1) Requirements discovery

✧ The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.

Sources:

✧ Stakeholders
🟨 Interview 🟧 Scenario 🟧 Use Case 🟥 Ethnography

✧ Documentation, spec of similar systems

✧ Domain requirements, . .

🟨 Interviewing

✧ Part of most RE processes

✧ Types of interview:

formal vs informal

🟨 Interviewing

✧ Part of most RE processes

✧ Types of interview:

formal vs informal

closed vs open

Typically a mix of closed and open-ended interviewing.

Interviewing

Specification

✧ Effective interviewing

Interviewing

Specification

✧ Effective interviewing
- Be open-minded
- Avoid pre-conceived ideas
- Willing to listen

Interviewing

Specification

✧ Effective interviewing
- Be open-minded
- Avoid pre-conceived ideas
- Willing to listen

✧ Ways to get discussions going:
- springboard question
- requirements proposal
- working together on a prototype system.

Interviews in practice

Specification

⊕ Interviews are good for:
- overall understanding what stakeholders do
- how they might interact with system
- identify difficulties with current system

Interviews in practice

Specification

⊕ Interviews are good for:
- overall understanding what stakeholders do
- how they might interact with system
- identify difficulties with current system

⊖ Interviews are not good for
- understanding domain requirements
- Organizational requirements

Scenarios

Specification

✧ Scenarios are real-life examples of how a system can be used.

✧ Small a number of possible interactions

✧ Easy to relate to

✧ Good for adding detail

Scenarios

✧ Scenarios are real-life examples of how a system can be used.

✧ Pro: easy to relate to

✧ They should include

- starting situation;
- normal flow of events;
- what can go wrong;
- Information about other concurrent activities;
- state when the scenario finishes.

Scenario for collecting medical history in MHC-PMS

Initial assumption: The patient has seen a medical receptionist who has created a record in the system and collected the patient's personal information (name, address, age, etc.). A nurse is logged on to the system and is collecting medical history.

Normal: The nurse searches for the patient by family name. If there is more than one patient with the same surname, the given name (first name in English) and date of birth are used to identify the patient.

The nurse chooses the menu option to add medical history.

The nurse then follows a series of prompts from the system to enter information about consultations elsewhere on mental health problems (free text input), existing medical conditions (nurse selects conditions from menu), medication currently taken (selected from menu), allergies (free text), and home life (form).

Scenario for collecting medical history in MHC-PMS

What can go wrong: The patient's record does not exist or cannot be found. The nurse should create a new record and record personal information.

Patient conditions or medication are not entered in the menu. The nurse should choose the 'other' option and enter free text describing the condition/medication.

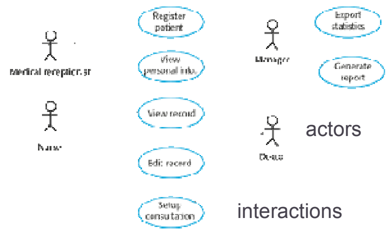
Patient cannot/will not provide information on medical history. The nurse should enter free text recording the patient's inability/unwillingness to provide information. The system should print the standard exclusion form stating that the lack of information may mean that treatment will be limited or delayed. This should be signed and handed to the patient.

Other activities: Record may be consulted but not edited by other staff while information is being entered.

System state on completion: User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start and end time of the session and the nurse involved.

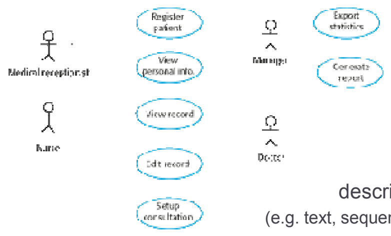
Use cases

✧ Use case diagram (UML);



Use cases

✧ Use case diagram (UML);



description of use-case
(e.g. text, sequence / state diagram..)

✧ Set of use cases should describe all possible interactions

Scenarios and use cases

➕ Effective for eliciting requirements

⛔ Not as effective for eliciting constraints, high-level business requirements, non-functional requirements, or domain requirements

Ethnography

✧ Social scientist spends considerable time observing and analysing how people actually work.

✧ People do not have to explain or articulate their work.

Ethnography

✧ Social scientist spends considerable time observing and analysing how people actually work.

✧ People do not have to explain or articulate their work.

⊕ Effective in discovering requirements that are derived

- from way in which people actually work (alarm)
- From cooperation / awareness of other people's activities

Ethnography

✧ Social scientist spends considerable time observing and analysing how people actually work.

✧ People do not have to explain or articulate their work.

⊕ Effective in discovering requirements that are derived

- from way in which people actually work (e.g. alarm)
- From cooperation / awareness of other people's activities (e.g. screen)

⊖ Cannot identify new features
Might study existing but outdated practices

1) Requirements discovery Review:

✧ The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.

Sources:

✧ Stakeholders

🟨 Interview 🟧 Scenario 🟧 Use Case 🟥 Ethnography

✧ Documentation, spec of similar systems

✧ Domain requirements, . .

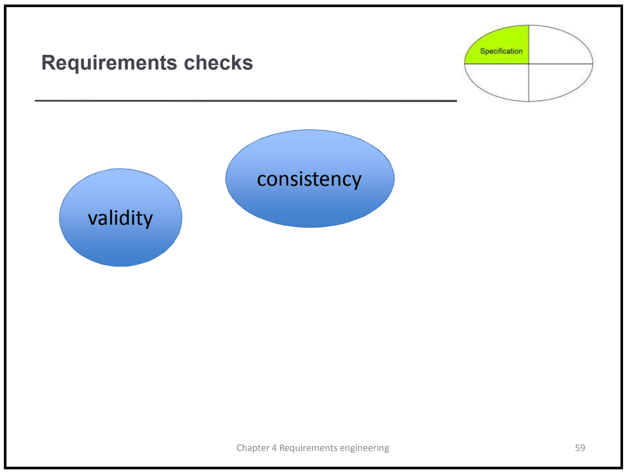
4.6 Requirements validation

✧ To demonstrate that the requirements define the system that the customer wants.

✧ Requirements error costs high => validation important

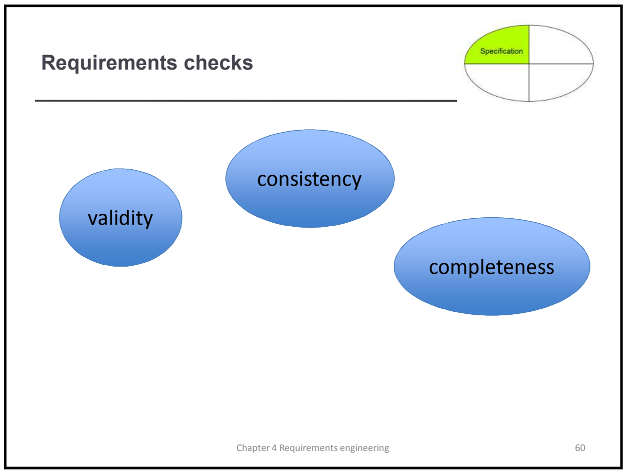
Requirements checks

validity



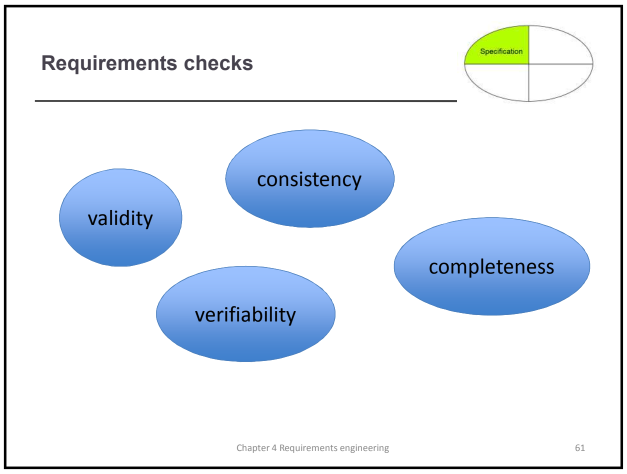
Requirements checks

Chapter 4 Requirements engineering 59



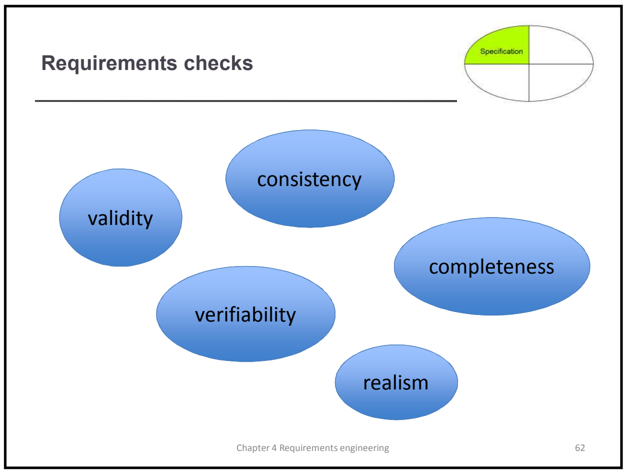
Requirements checks

Chapter 4 Requirements engineering 60



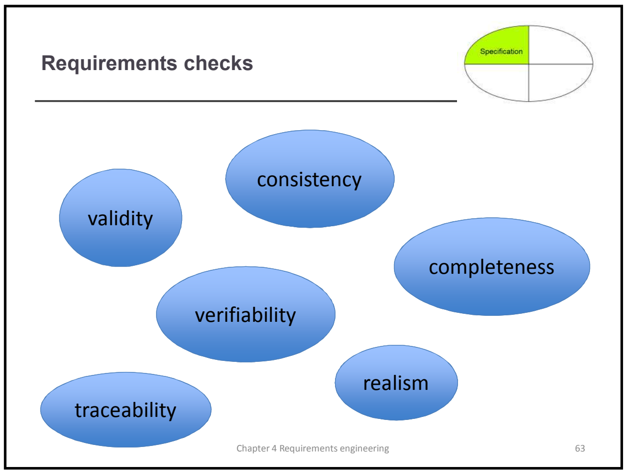
Requirements checks

Chapter 4 Requirements engineering 61



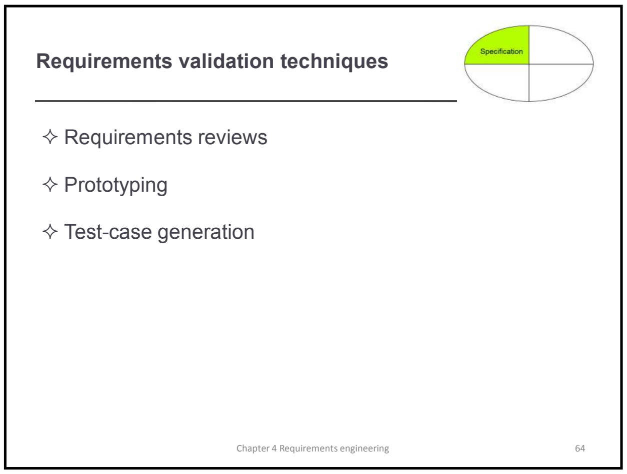
Requirements checks

Chapter 4 Requirements engineering 62



Requirements checks

Chapter 4 Requirements engineering 63

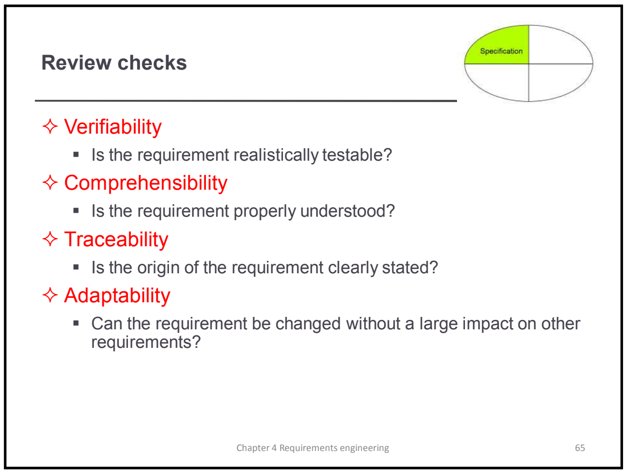


Requirements validation techniques

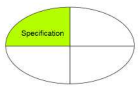
Specification

- ✧ Requirements reviews
- ✧ Prototyping
- ✧ Test-case generation

Chapter 4 Requirements engineering 64



Review checks



- ✧ Verifiability
  - Is the requirement realistically testable?
- ✧ Comprehensibility
  - Is the requirement properly understood?
- ✧ Traceability
  - Is the origin of the requirement clearly stated?
- ✧ Adaptability
  - Can the requirement be changed without a large impact on other requirements?

Chapter 4 Requirements engineering 65