CS5030

Requirements Engineering

Learning outcomes

- On completing this lecture and associated reading, you should
  - Be aware of the concept of requirements in general, and user and system requirements in particular
  - Understand why user and system requirements should be specified differently
  - Be able to explain the differences between functional and non-functional requirements
  - Be able to describe how requirements may be gathered in plan-driven development processes
  - Be aware of different notations for specifying requirements and their relative merits and limitations

Why is this topic important?

“The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.”

- Frederick Brooks, Jr

Requirements of a system

- Descriptions of the services a system should provide and the constraints on its operation
  - Reflect the needs of the customers of the system
- Can range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification
- Can be the basis for
  - bid for contract - should be open to interpretation
  - contract itself - should be precise and detailed

Requirements engineering

- Process of finding out, analysing, documenting and checking requirements (system services and constraints)

  [Sommerville, 2016]

- The branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families.

  [Zave, 1997]

Requirements & development processes

- Plan-driven
  - Initially from user perspective
  - Translated to system perspective

- Agile
  - From user perspective
  - Features that users want from the system

Requirements in plan-driven development

- User requirements
- System requirements
- Domain requirements

User requirements

- Statements in natural language + diagrams
- What the system is expected to provide customers and the constraints under which it will operate
- Written for customers

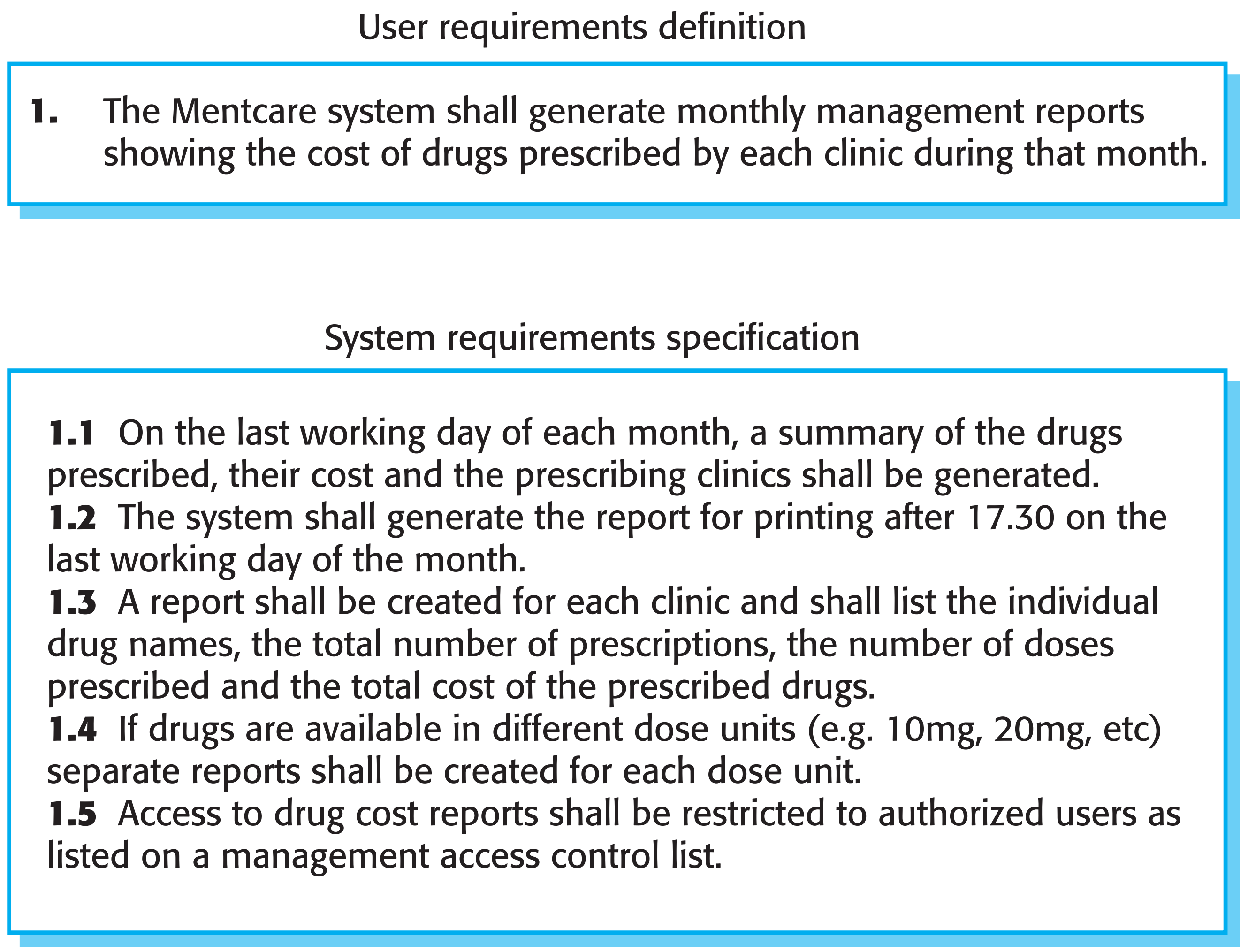
System requirements

- Structured document with details of what should be implemented

- Can be part of contract between customer and developing organisation

- Functional and non-functional requirements
  - Should be atomic

User and system requirements - example



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.

[Sommerville, 2016]

Functional requirements

- What the system should do

- 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 also state what the system should not do

Non-functional requirements

- Specify or constrain characteristics of the system as a whole
  - Often apply to the system rather than individual features or services

- Can be more critical to system success than individual functional requirements

- Examples: usability, performance, security, availability, …

Non-functional requirements - challenges

- Users / customers often propose non-functional requirements as general goals instead of measurable requirements
  - Scope for interpretation and subsequent disputes after system is delivered
  - For example
    - “System should be easy to use”
    - “User errors should be minimised”

Non-functional requirements - challenges

- Some requirements have no simple metrics
  - such as maintainability

- Even if quantitative specification is possible, customers may not be able to relate their needs to them

- Cost of objectively verifying non-functional requirements can be very high

Metrics for non-functional requirements

- Speed
  - Transactions / second, response time to events
- Size
  - Mbytes
- Ease of use
  - Training time, support features
- Reliability
  - Mean time to failure, availability, rate of failure occurrence
- Robustness
  - Time to restart, probability of data corruption, percentage of events causing failure
- Portability
  - Number of target systems

Domain requirements

- Derived from system domain rather than user perspective
- Requirements from operational domain
  - New functional requirements
  - Constraints on existing requirements
  - Specific computations
- If domain requirements are not satisfied, system may be unusable

Domain requirements - challenges

- Understandability
  - Constraints expressed in the language of the application domain
  - May not be understood by development team
    - Omissions and conflicts may be missed

- Implicitness
  - Domain experts may not realise the need to explicitly state these requirements because they understand the domain so well

Requirements engineering activities

- Requirements elicitation and analysis
- Requirements specification
- Requirements validation
- Requirements management

Requirements elicitation and analysis

- Aims
    - Understand the work of stakeholders
    - Understand how the new system might support this work

- Requirements engineers work with stakeholders to find out about
    - Application domain
    - Services the system should provide
    - Operational constraints

Requirements elicitation techniques

- Interviewing
- Observation or ethnography

Requirements elicitation - challenges

- Stakeholders don’t always know
  - What precisely they want from the software system
  - What a software system can and can’t do
- Stakeholders express requirements in their own terms
  - Lack of common vocabulary with requirements engineers
- Different stakeholders may have different / conflicting requirements

Requirements elicitation - challenges

- Political factors within customer organisation may affect requirements

- Requirements may change during the elicitation and analysis process
  - Dynamic business and economic environments
  - Changes in stakeholders

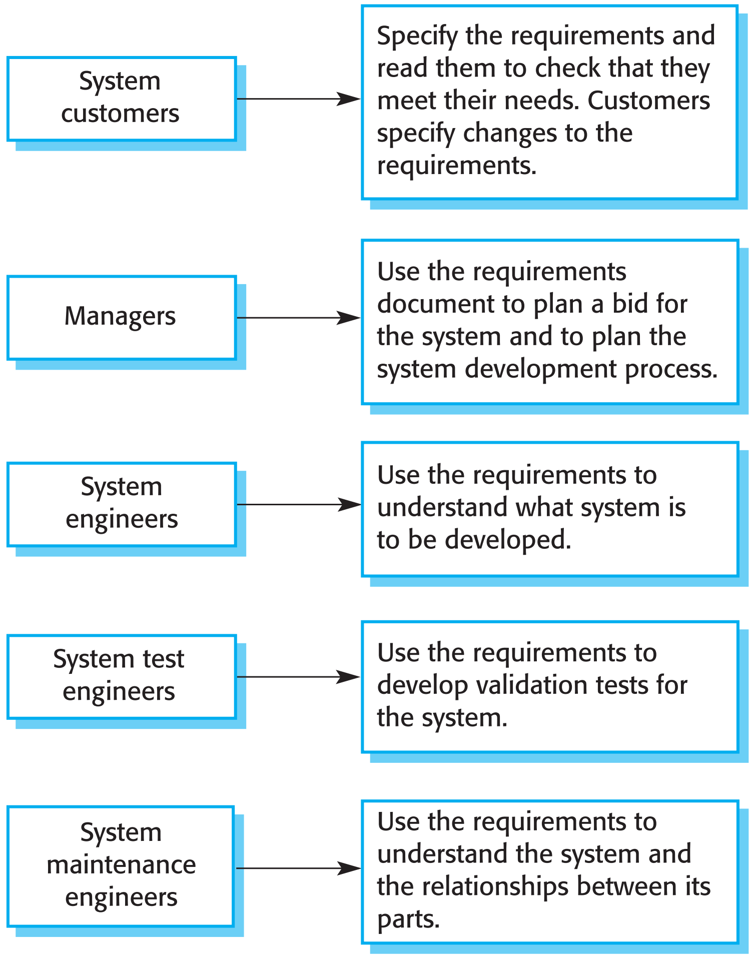
Requirements specification

- Process of documenting user and system requirements
- Notations
  - Natural language
  - Structured natural language
  - Graphical notations
  - Mathematical specifications
- Process and notation may differ according development process

Requirements document

- Official statement of what is required of software developers
  - Typically for plan-driven development
- Should include user and system requirements
- It should say what the system should do rather than how it should do it
  - Not design
- Systems developed incrementally will have less detail in their requirements document
- There are standards defined for requirements documents
  - For eg, 29148-2018 - ISO/IEC/IEEE International Standard
  - Typically used for large engineering projects

Users of requirements documents



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.

[Sommerville, 2016]

Natural language specification

- Example
  - *3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes*
- Numbered sentences in natural language
- Supplemented by diagrams and tables
- Each sentence should express one requirement
- Can be understood by customers and developers

Natural language - guidelines

- Use a standard format for all requirements
- Use language consistently – for example
  - *Shall* for mandatory requirements
  - *Should* for desirable requirements
- Can use highlighting to identify key parts of requirements
- Avoid using software engineering jargon
- Include rationale for requirement

Structured requirements specification

- Requirements are written in natural language in a standard format
  - Each field provides information about a specific aspect of the requirement

- Suitable for requirements where details are essential

Sample structure

- 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)*
  - *Previous two readings (r0 and r1)*
- Source
  - *Current sugar reading from sensor*
  - *Previous readings from memory*
- Outputs
  - *Dose of insulin to be delivered (CompDose)*
- Destination
  - *Main control loop*

Sample structure (contd.)

- Action
    - *CompDose is zero if the sugar level is stable or falling, or if the level is increasing and the rate decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is calculated by . . .*
- Requirements
    - *Two previous readings, so that the rate of change can be computed*
- Precondition
    - *The insulin reservoir contains at least the maximum allowed single dose*
- Postcondition
    - *r0 is replaced by r1, then r1 is replaced by r2*
- Side effects
    - *None*

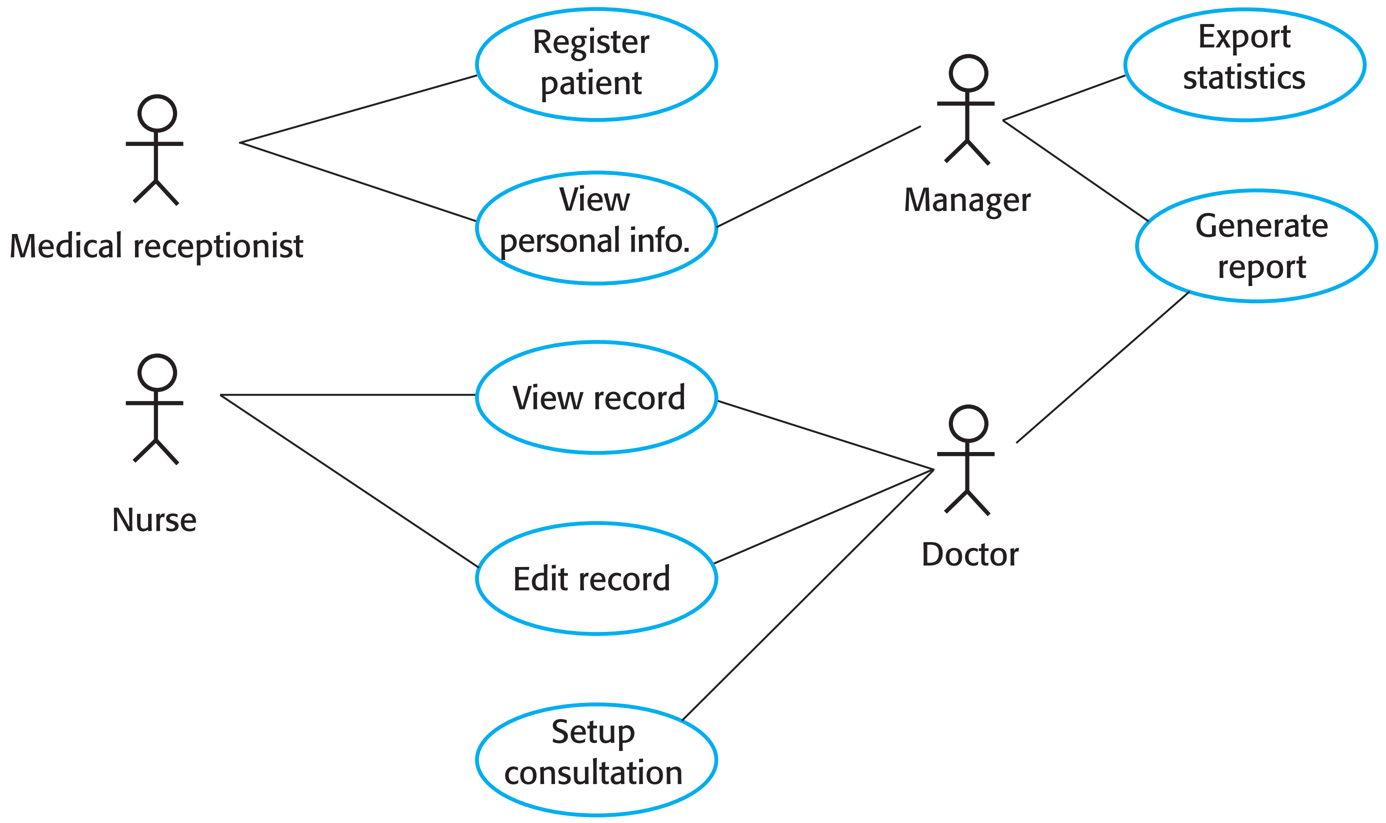
A simpler structure

- Requirement id
- Description
- Priority
- Dependencies
- Source

Use cases

- Scenario / interaction modelling in UML
- High-level graphical models
- Describe system interactions showing actors
- A set of use cases for a system should show all possible interactions with the system
- UML sequence diagrams can be used to add more detail to use cases
  - We will study these later

Use case - example



[Sommerville, 2016]

Qualities of a good requirements specification

- Precision
- Validity
- Pertinence
- Completeness
- Consistency
- Verifiability / measurability
- Comprehensibility
- Feasibility
- Traceability
- Adaptability

Requirements validation

- Aims to demonstrate that documented requirements define the system that the customer wants
- Properties
  - Validity, consistency, completeness, realism, verifiability
- Validation techniques
  - Requirements reviews, prototyping, test case generation
- Important because costs of requirements errors are high

Requirements management

- Process of managing changing requirements during the software engineering process
  - While the system is being developed
  - After it is operational
- Support for requirements management
  - Tracking individual requirements
  - Tracking dependencies among requirements
  - Establishing a change management process

Key points

- Requirements are key to software engineering. Many software failures can be traced back to problems with requirements
- Requirements of a system specify what it should do and define constraints on its implementation and operation
- Requirements engineering for plan-driven development is an iterative process and involves elicitation, analysis, specification, validation and management activities
- Requirements change over time for a variety of reasons; therefore change management is an essential part of requirements engineering
- Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document
- The software requirements document is an agreed statement of the system requirements
- It should be organised so that both system customers and software developers can use it for their purposes