Requirements Engineering (/elicitations!)

(chapter 4: Sommerville;
Chapter 4: Bruegge)

Dr Adel Taweel

– Software Requirements Specifications–
Descriptions and specifications of a system

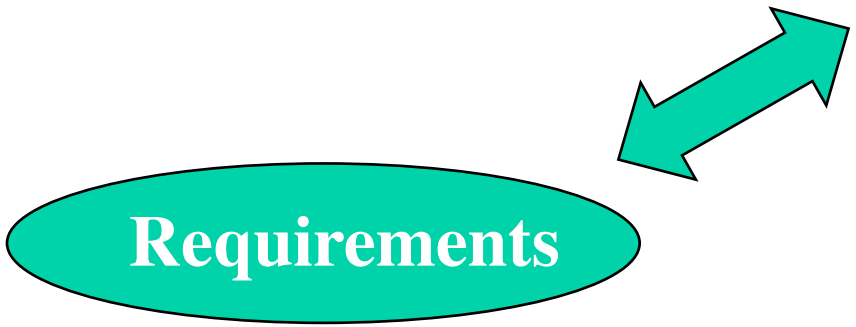
Objectives:

- To introduce the concepts of **user, domain and system requirements**
- To describe **functional / non-functional requirements**
- To explain **techniques** for describing system requirements
- To explain **how software requirements may be organised** in a requirements document
- To introduce some methods for **requirements discovery**

Requirements engineering

Requirements engineering is the process of establishing

- the services (or functionalities) that the customer requires from a system
- the constraints under which it operates and is developed



The descriptions of the system services and constraints

that are generated during the requirements engineering process

What is a requirement?

It may range from a **high-level** abstract statement of a service or of a system constraint to a **detailed** mathematical functional specification

This is inevitable as requirements may serve a **dual function**

- May be the basis for a bid for a contract - therefore must be open to interpretation
- May be the basis for the contract itself - therefore must be defined in detail
- Both these statements may be called requirements

Types of requirement

User requirements

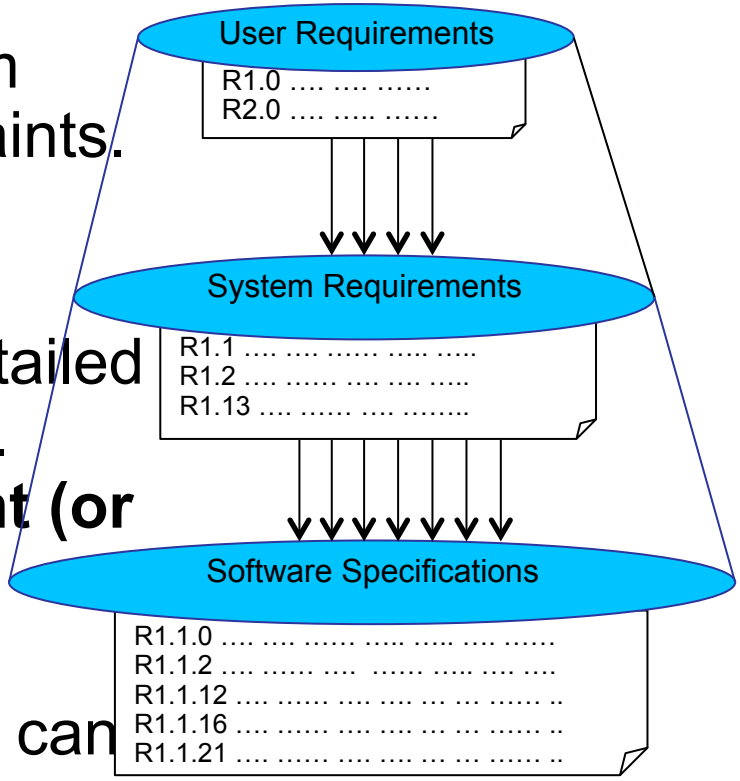
Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for **customers**

System requirements

A structured document setting out detailed descriptions of the system services. Written as a contract between **client (or customer)** and **contractor**

Software specification

A detailed software description which can serve as a basis for a design or implementation. Written for **developers**



User and system requirements

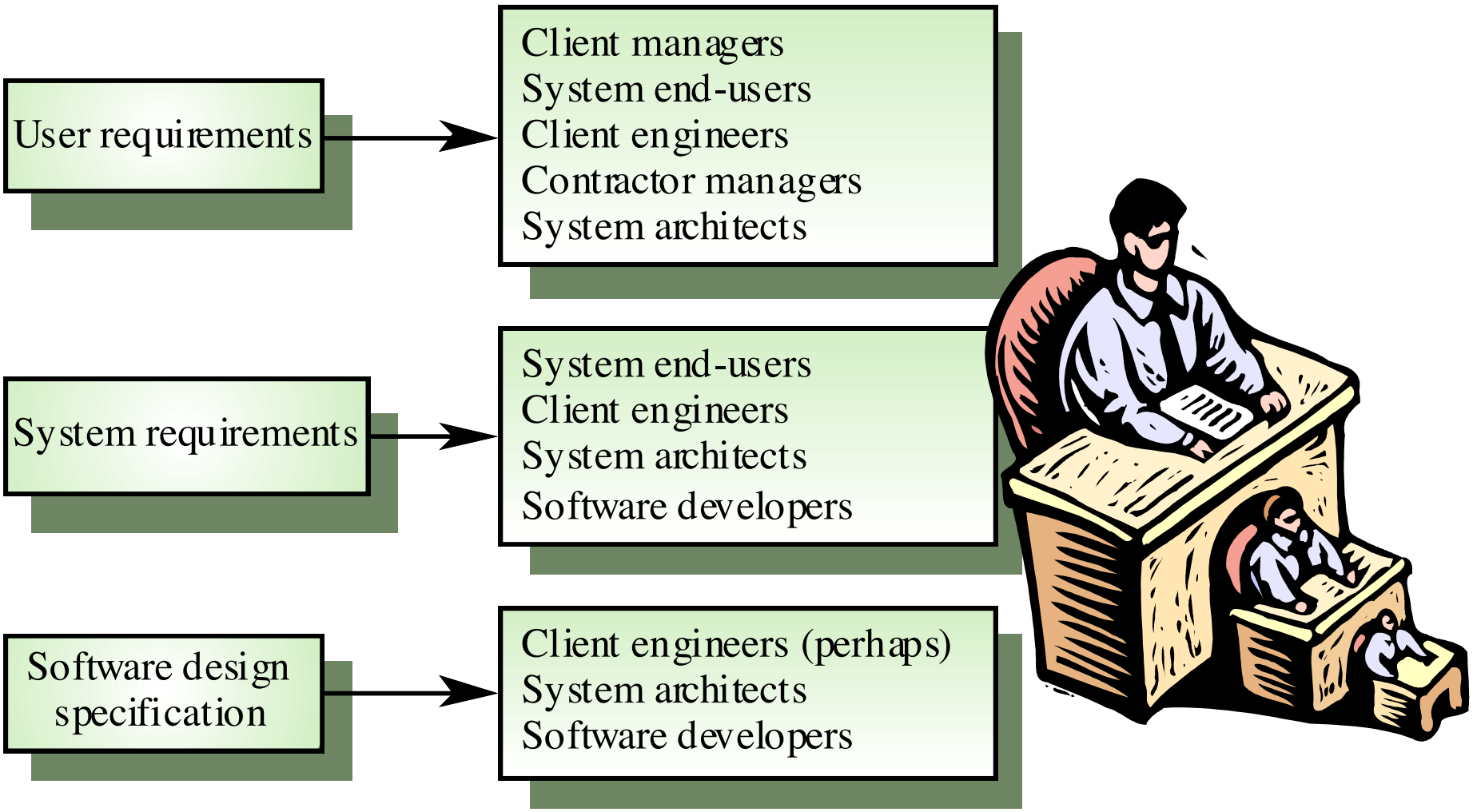
User requirement definition

1. The MHC-PMS 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 automatically 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, 20 mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

Requirements readers



Functional and non-functional requirements

Functional requirements

Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.

Non-functional requirements

constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

Domain requirements

Requirements that come from the application domain of the system and that reflect characteristics of that domain

Functional Requirements

Describe functionality or system services

Depend on the type of software, expected users and the type of system where the software is used

Functional user requirements may be high-level statements of what the system should do **BUT** **functional system requirements** should describe the system services in detail

Functional requirements: Examples

2.3: A user shall be able to search the appointments lists for all clinics.

...

5.2: The PMS system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.

...

6.1: Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Non-functional requirements

Define system properties and constraints

e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.

Process requirements may also be specified mandating a particular development environment (IDE), programming language or development method

Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless to the customer!

Non-functional classifications

Product requirements

Requirements which specify that the **delivered product must behave in a particular way** e.g. execution speed, reliability, etc.

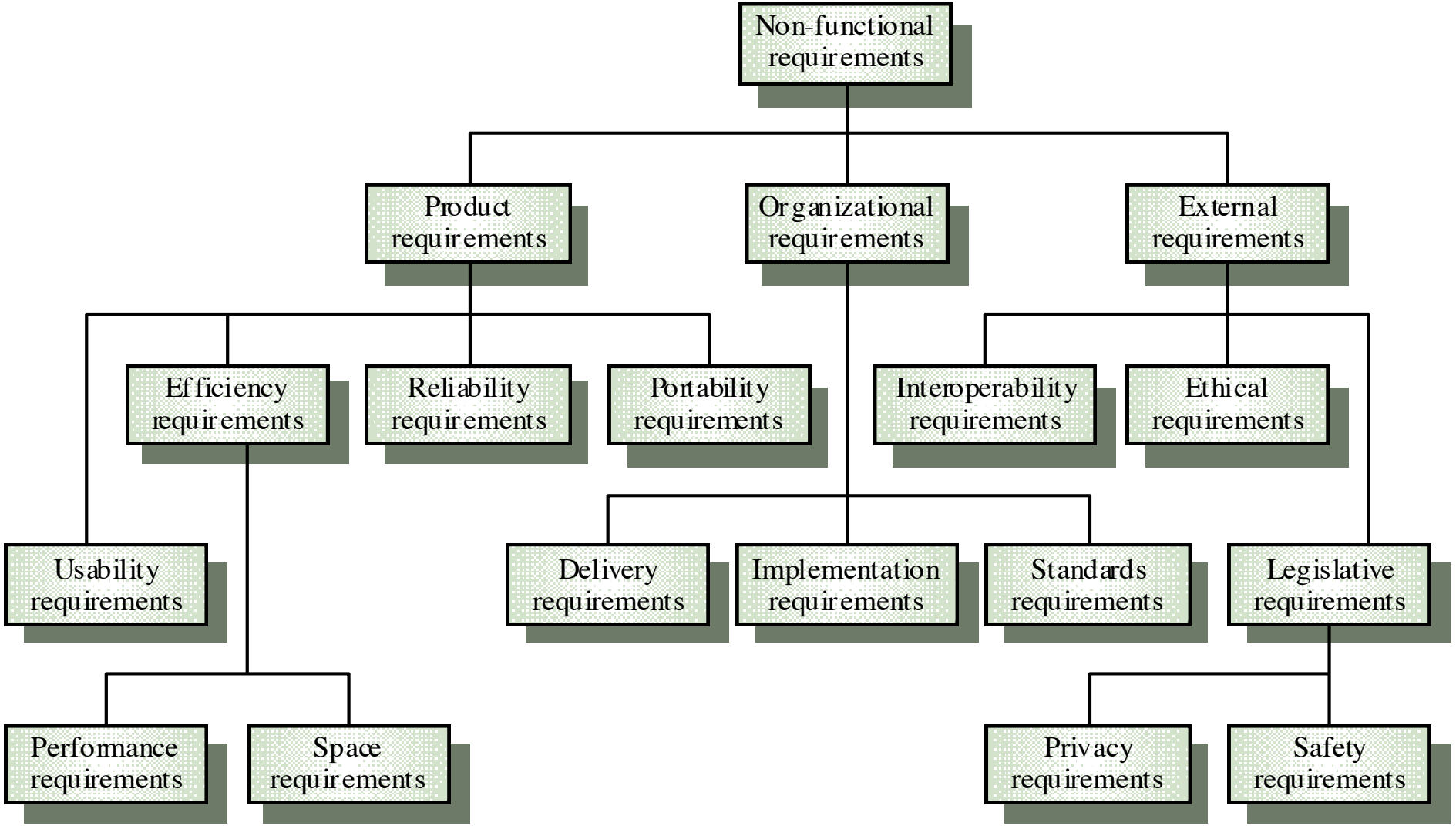
Organisational requirements

Requirements which are a **consequence of organisational policies and procedures** e.g. process standards used, implementation requirements, etc.

External requirements

Requirements which arise from factors which are **external to the system and its development process** e.g. interoperability requirements, legislative requirements, etc.

Non-functional requirement types



Non-functional requirements:
Examples

- **Product requirement**
  3.C.8.The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

- **Organizational requirement**
  5.4.3 Users of the MHC-PMS system shall authenticate themselves using their health authority identity card.

- **External requirement**
  7.2.3 The system shall implement patient privacy provisions as set out in the regulation HStan-03-2006-priv.

Goals and requirements

Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.

Goal

A general intention of the user such as ease of use

Verifiable non-functional requirement

A statement using some measure that can be objectively tested

Goals are helpful to developers as they convey the intentions of the system users

Example: Usability requirements

- **A system goal**
  G.8.1 The PMS system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (**Goal – non verifiable!**)

- **A verifiable non-functional requirement**
  8.4.3 Medical staff shall be able to use all the PMS system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use. (**Testable non-functional requirement**)

Non-functional requirements Metrics

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Domain requirements

Derived from the application domain and **describe system characteristics and features** that reflect the domain

- **Example 1:**
  R.7.1 a train control system has to take into account the braking characteristics in different weather conditions.
- **Example 2:**
  8.4.3 a PMS has to enforce all confidentiality rules in accordance with national medical domain practices

May generate new functional requirements, (non-functional) constraints on existing requirements or define specific computations

If domain requirements are not satisfied, the system may be unworkable

Domain requirements problems

Understandability

Requirements are expressed in the language of the application domain

This is often not understood by software engineers developing the system

Implicitness

Domain specialists understand the area so well that they do not think of making the domain requirements explicit

Requirements Characteristics

Requirements Correctness

A requirement is correct when it is part of the actual needs of the system.

Problems arise when requirements are implied or derived and become beyond the scope of the actual needs of system

> 2.3: A user shall be able to search the appointments lists for all clinics.
> ...

Consider the term 'search' in requirement 2.3

- User requirement– search for a patient name across all appointments in all clinics;
- Implied requirements– search for a patient name, Date of Birth, Address in a clinic.

Requirements precision/ unambiguous

Problems arise when requirements are not precisely stated

Ambiguous requirements may be interpreted in different ways by developers and users

Consider the term 'search' in requirement 2.3

User intention – search for a patient name across all appointments in all clinics;

Developer interpretation – search for a patient name in an individual clinic. User chooses a clinic at a time then search? OR

- search for a patient name in all clinics at once. User enters patient details and search in all available clinics?

Requirements completeness and consistency

In principle requirements should be both complete and consistent

Complete

They should include descriptions of all services required including explicitly stated or externally imposed/implied requirements.

Consistent

There should be no conflicts or contradictions in the descriptions of the system services

In practice, it is very difficult or, in fact, impossible to produce a complete and consistent requirements document

Requirements Consistency: Example

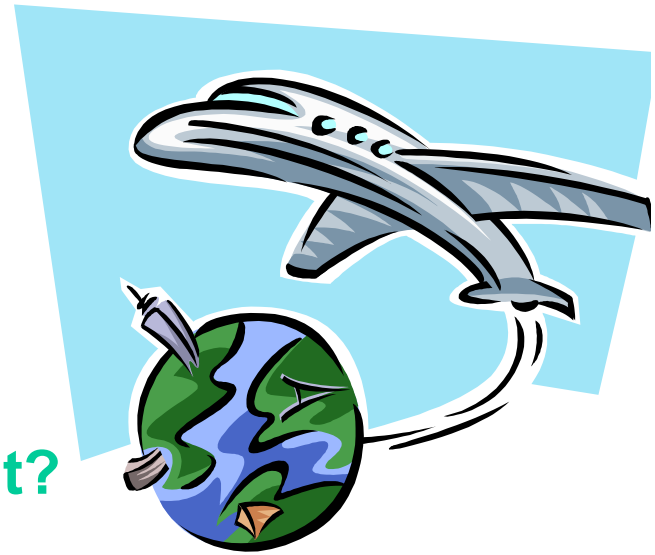
Conflicts between different non-functional requirements are common in complex systems

Spacecraft system

- To minimise weight, the number of separate chips in the system should be minimised
- To minimise power consumption, lower power chips should be used

However, using low power chips may mean that more chips have to be used.

Which is the most critical requirement?



Requirement Traceability

Each requirement must be traceable
Traceability is critical for requirement verification, validation and user acceptance.

Requirements are often uniquely identified by a unique number to be traced/referenced in validation and testing phases.

Example: requirement has its own unique ID/Number

R2.3: A user shall be able to search the appointments lists for all clinics.

Documentation of Requirements

User requirements

Should describe functional and non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge

User requirements are defined using natural language, **tables** and **diagrams** (latter ones will be discussed later)

System requirements

- **More detailed specifications of user requirements**

Serve as a basis for designing the system

May be used as part of the system contract

System requirements may be expressed using, natural language and system models (latter ones will be discussed in later lectures)

Requirements document requirements

Specify external system behaviour
Specify implementation constraints
Easy to change
Serve as reference tool for maintenance
Record forethought about the life cycle of the system i.e. predict changes
Characterise responses to unexpected events

IEEE requirements standard

Introduction
General description
Specific requirements
Appendices
Index

This is a generic structure that must be instantiated for specific systems

Requirements document structure

Introduction
Glossary
User requirements definition
System architecture
System requirements specification
System models
System evolution
Appendices
Index

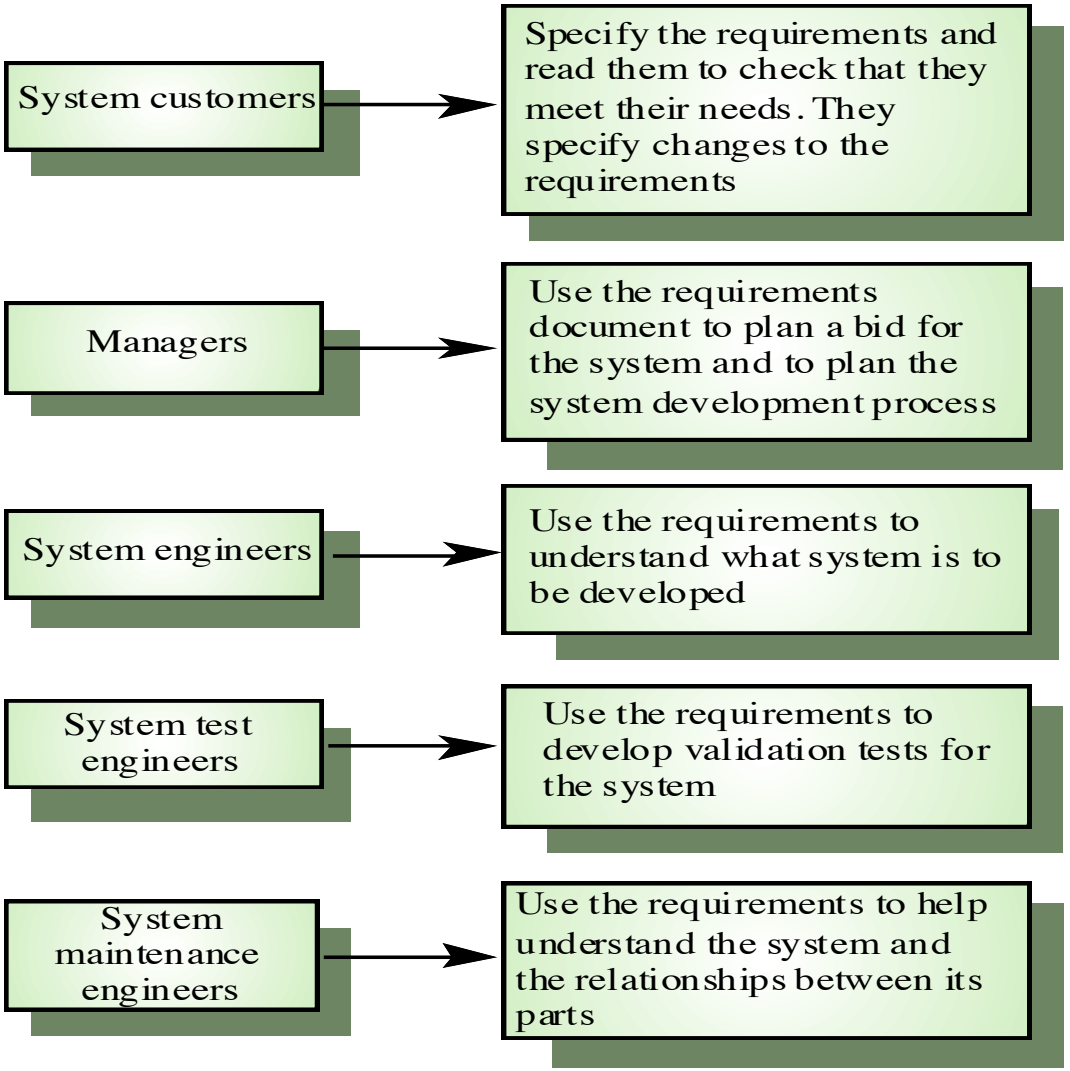
The structure of a requirements document-1

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 structure of a requirements document-2

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.

Users of a requirements document



Guidelines for writing requirements

Invent a standard format and use it for all requirements

Use language in a consistent way. Use

shall for mandatory (or forceful) requirements,

should for desirable requirements

Use text highlighting to identify key parts of the requirement

Include an explanation (rationale) of why a requirement is necessary

Avoid the use of computer jargon !!!

Requirements and design

In principle, requirements should state **what the system should** do and the design should describe **how the system does it**

In practice, requirements and design are inseparable

- A system architecture may be designed to structure the requirements
- The system may inter-operate with other systems that generate design requirements
- The use of a specific design may be a domain requirement

Ways of writing 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 activity 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

Problems with natural language

Lack of clarity

Precision is difficult without making the document difficult to read

Requirements confusion

Functional and non-functional requirements tend to be mixed-up or confused

Requirements amalgamation

Several different requirements may be expressed together

A requirement in Natural Language: Example

R3.2 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.)*

R3.6 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.)*

A structured specification of a requirement: Example 1

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: Example 2

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.

A structured specification of a requirement: Example 3

Title	**Compute Insulin Dose (CompDose)**
Purpose	To compute insulin dose based on the measured sugar level.
Description	Computes the dose of insulin to be delivered when the current measured level of sugar is in the safe zone between 3 and 7 units.
Actors	SystemTimer (actors that interact with this requirement)
Trigger	Automatic (triggered automatically every 10 minutes by SystemTimer)
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin
Workflow	1. obtain current sugar level reading **r2** 2. read stored previous two sugar level readings, **r0** and **r1** 3. compute increasing/decreasing level of sugar within safe zone 4. compute a single dose of insulin based on sugar level 5. if dose of insulin is within allowed limits (5-15 mg), deliver insulin dose 6. else generate beep sound 7. replace previous readings with current readings r0=r1 and r1=r2.
Post-condition	Previous readings replaced and stored.

Tabular specification: Example

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

Requirements Engineering Processes and Discovery

Requirements engineering processes

Generic activities common to all processes
- Requirements elicitation;
- Requirements analysis;
- Requirements validation;
- Requirements management.

In practice, RE is an iterative activity in which these processes are interleaved.

Requirements elicitation and analysis

Sometimes called requirements elicitation or requirements discovery.

Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.

May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders*.

Requirement Analysis

The process of understanding customer requirements and their implications. It comes after requirements discovery

It involves technical staff working on the discovered requirements, iteratively with customers, to understand their technical implication and importance.

It includes the processes of classifying and organising requirements, negotiating (with customers) and prioritizing them in the order of their importance to the customers.

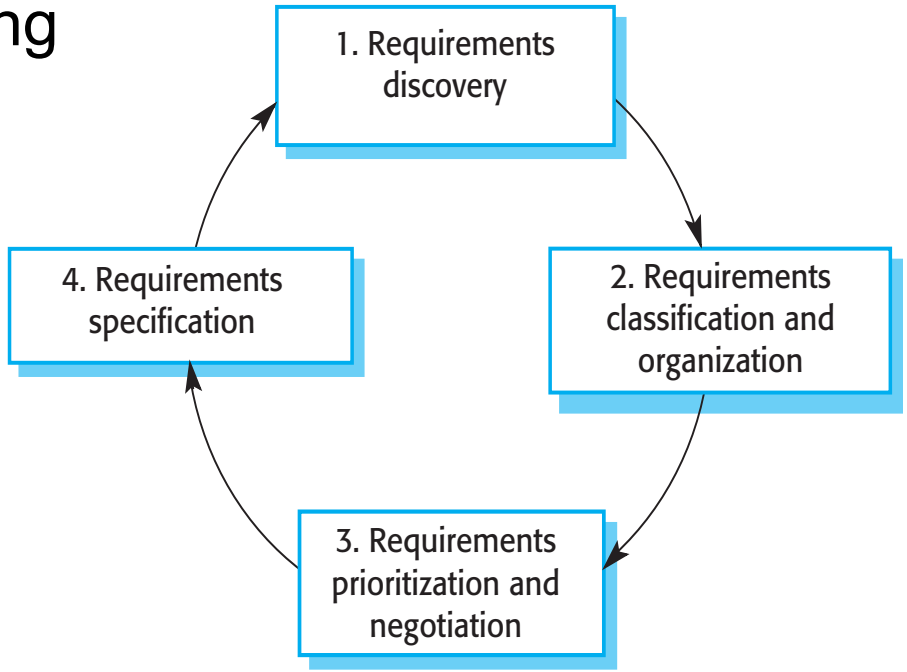
The output of this process is the requirement specification document (**SRS**) negotiated and accepted with customers.

Problems of requirements analysis

Stakeholders don't know what they really want.
Stakeholders express requirements in their own terms.
Different stakeholders may have conflicting requirements.
The requirements change during the analysis process

Stages include:

- Requirements discovery,
- Requirements classification and organization,
- Requirements prioritization and negotiation,
- Requirements specification.



Stakeholders in the MHC-PMS

Patients whose information is recorded in the system.
Doctors who are responsible for assessing and treating patients.
Nurses who coordinate the consultations with doctors and administer some treatments.
Medical receptionists who manage patients' appointments.
IT staff who are responsible for installing and maintaining the system.
etc

Techniques

There are many requirement engineering techniques for requirement elicitation and analysis, some of the often used ones:

- Interviewing
- Scenario generation
- Use case analysis
- Ethnography

Interviewing

Formal or informal interviews with stakeholders are part of most RE processes.

Types of interview

- Closed interviews based on pre-determined list of questions
- Open interviews where various issues are explored with stakeholders.
- Focused interviews, with clusters of stakeholders

Effective interviewing

- Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
- Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

Interviews

Meeting introductory protocol
- Ensure cultural introduction protocols are followed

First meeting
- Aim: to understand the business and its context with a clear aim to understand business processes and services.

Effective meetings:
- Ensure a chair is assigned at the beginning, to keep time-controlled progress
- Ensure an agenda is defined with clear objectives of the target outcome of the meeting
- Ensure a timescale is set for each agenda item and is kept/ controlled by the chair
- Ensure clear actions and decisions (and who is responsible for and by when) are identified and reached by the end of the meeting
- Ensure the actions and decisions are summarised at the end of the meeting

Interviews in practice

Normally a mix of closed and open-ended interviewing.
Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.

Interviews are not good for understanding domain requirements
- Requirements engineers cannot understand specific domain terminology;
- Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

Scenarios

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

They should include

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

Scenario for collecting medical history: Example

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: Example

What can go wrong?	Successful output?
Alternative: The patient's record does not exist or cannot be found. The nurse should create a new record and record personal information.	Yes
Alternative: 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.	Yes
Error: 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.	No?
Other activities: Record may be consulted but not edited by other staff while information is being entered.	
System state on completion: User (nurse) 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

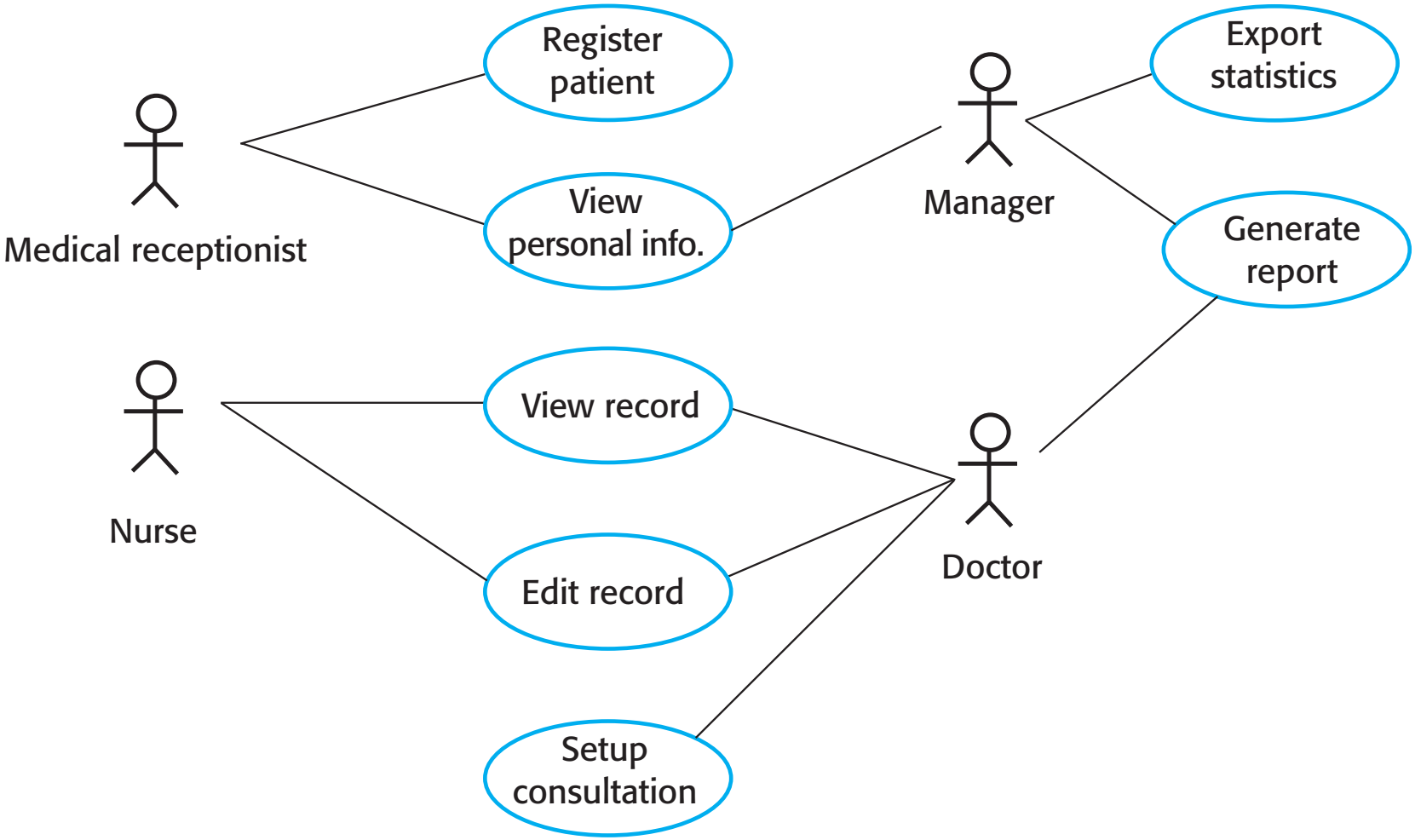
Use-cases are a scenario based technique, in the UML, which identifies the actors in an interaction and describes the interaction itself.

A set of use cases should describe all possible interactions with the system.

High-level graphical model supplemented by more detailed structured or tabular description.

Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use cases for the MHC-PMS



Ethnography

A social scientist spends a considerable time observing and analysing how people actually work.

People do not have to explain or articulate their work.

Social and organisational factors of importance may be observed.

Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Requirements validation

Concerned with demonstrating that the requirements define the system the customer really wants.

Requirements error costs are high so validation is very important

Fixing a **requirements error** after delivery may cost up to **100 times** the cost of fixing an implementation error.

Requirements checking

Correctness/Validity. Does the system provide the functions which best support the actual customer's needs?

Consistency. Are there any requirements conflicts?

Completeness. Are all functions required by the customer included?

Traceability/Verifiability. Can the requirements be verified/validated?

Realism/Feasibility. Can the requirements be implemented within specified time given available budget and technology

Requirements validation techniques

Requirements reviews

- Systematic manual analysis of the requirements.

Prototyping

- Using an “executable model” of the system to check requirements.

Test-case generation

- Developing tests for requirements to check testability.

Teams-T2-2017

Customer

Developer

10 min		10 min	
G1	G2	G10	G1
G3	G4	G2	G3
G5	G6	G4	G5
G7	G8	G6	G7
G9	G10	G8	G9

(G1): LI
(G2): MA
(G3): MA
(G4): IB
(G5): AS
(G6): OS
(G7): SK
(G8): SH
(G9): YI
(G10): MM

Key points

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. They describe WHAT the system should undertake.

Non-functional requirements often constrain the system being developed and the development process being used. They often describe HOW WELL the system should undertake its functional requirements

Requirements often relate to the emergent properties of the system and therefore apply to the system as a whole.

Key points

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

You can use a range of techniques for requirements Engineering or elicitation including interviews, scenarios, use-cases and ethnography.

Requirements validation is the process of checking the requirements for validity, consistency, completeness, correctness and realism, unambiguity 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.

MidTerm (T2-2016/2017)

COMP433 SOFTWARE ENGINEERING

Venue: Bamieh202, Masri404

Date: Tuesday , 18/04/2017 14:00 - 15:30

Effort+Cost Estimation: Very simplified method

Simple developer-driven estimation method
pw= person week; **pm**= person month; **w**= week; **m**= month
effort= the effort required for a person employed all month/week long
Schedule time = time needed to complete including based on working days only (including holidays etc)

UR	Estimated Effort	Estimated No of Developers	Total Effort
UR1	2 pw	2	= 2 * 2 = 4pw
UR2	3 pw	1	= 3 * 1 = 3pw
UR3	2 pw	3	= 2 * 3 = 6pw
UR4 …	1 pw	4	= 1 * 4 = 4pw
Total effort/avg	8 pw	(2+1+3+4)/4=2.5 dev on avg needed	17 pw
Schedule time 30%	8 * 1.30=11 w (**min time** to complete)		17 *1.30=22w (**max time** to complete)
Cost		Avg salary= $250	250 * 22 w = $5500
Profit margin (min=10%; max=30%)		**Min cost →** **Max cost →**	5500 * 1.10= 6050 5500 * 1.30= 7150