Software Engineering

Software Requirements

Descriptions and specifications of a system

Objectives:

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

Topics Covered

- Functional and non-functional requirements
- User requirements
- System requirements
- The **software requirements document**

Software Requirements

- Recall from the last lectures the following decomposition of the software development lifecycle:

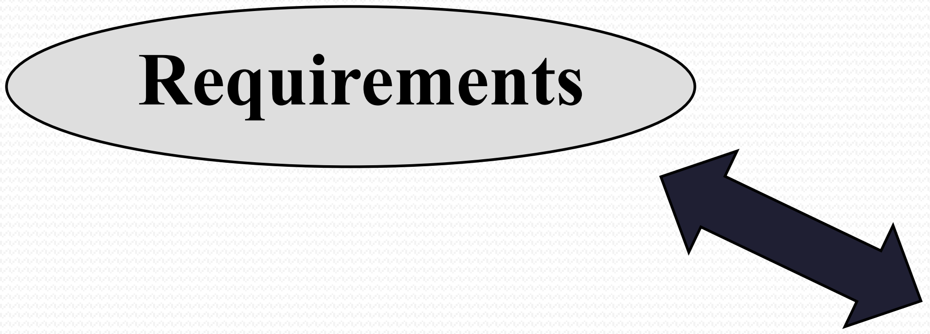
General Steps of Software Development:

- Specifying,
- Designing,
- Implementing,
- Testing.

Requirements Engineering

Requirements engineering is the process of establishing

- the services 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

Why do we need Requirements?

- To ensure a software solution correctly solves a particular problem, we must initially *fully understand* the problem that needs to be solved, discover *why* the problem needs to be solved and determine *who* should be involved.
- Poorly defined requirements can cause major problems to a project in both financial terms as well as added time.
- There are specific techniques we may use in the requirements engineering phase which we shall be considering during the next four lectures.

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 of these statements may be called requirements

Examples (requirements iteration)

- The system will support a wide range of the most commonly used graphics file formats
- The system may support the following file formats: png, jpeg, tiff and giff

- The system will support the following file formats: png, jpeg, tiff and giff

- The system may support the following file formats: png, jpeg, tiff and giff, to a maximum resolution of 1024x1024 pixels

- The system may support the following file formats: png, jpeg, tiff and giff, to a maximum resolution of 1024x1024 pixels

Types of Requirement

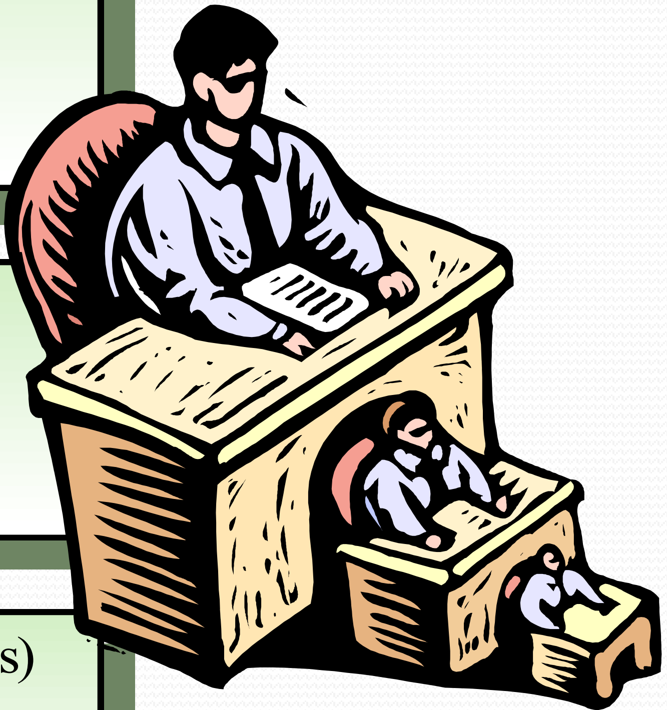
- **User requirements**
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for **customers**
- **System requirements**
  - A structured document setting out detailed descriptions of the system services. Written as a contract between **client** and **contractor**
- **Software specification**
  - A detailed software description which can serve as a basis for a design or implementation. Written for **developers**

Examples

- User requirement
  - We need to be able to spell check documents
- System requirement
  - The system needs to be able to spell check documents and provide autocorrect facilities. Their will be support for the following languages, English, French and German will plug in support for other languages
- Software specification
  - CheckResult spellCheck(String word, Dictionary dictionary)
    - Word is defined in UNICODE formatted string
    - The Dictionary structure is defined in S.1.2
    - The CheckResult is defined in S.1.3 and contains a flag if the word has been found or not, plus a Vector object containing a list of possible other word suggestions depending if the word has been found or not
    - spellCheck will ideally use Hashing tables to improve code efficiency
    - ……

Requirements Readers

User requirements	Client managers System end-users Client engineers Contractor managers System architects
System requirements	System end-users Client engineers System architects Software developers
Software design specification	Client engineers (perhaps) System architects Software developers



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. Usually defined on the system as a whole
- **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 <u>system services in detail</u>

Examples of Functional Requirements

- All users will access the system using a user id and a password
- The system shall support the following document formats: PDF, RTF, Microsoft Word 2010 and ASCII text
- Every order shall be allocated a unique identifier (ORDER_ID)
- The system have a mechanism to help recover a user’s password

Requirements Imprecision

- Problems arise when requirements are not precisely stated
- Ambiguous requirements may be interpreted in different ways by developers and users
- Consider the term ‘**recover password**’ from previous slide..
  - **User intention** – mechanism which allows the user to view the password after going through an authentication procedure
  - **Developer interpretation** – allowing the user to reset their password so that it can be set again (e.g. using email link)
- Before development is to commence requirements should be defined as precisely as possible

Requirements Completeness and Consistency

- In principle requirements should be both complete and consistent:

  Complete

  - They should include descriptions of all facilities required

  Consistent

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

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

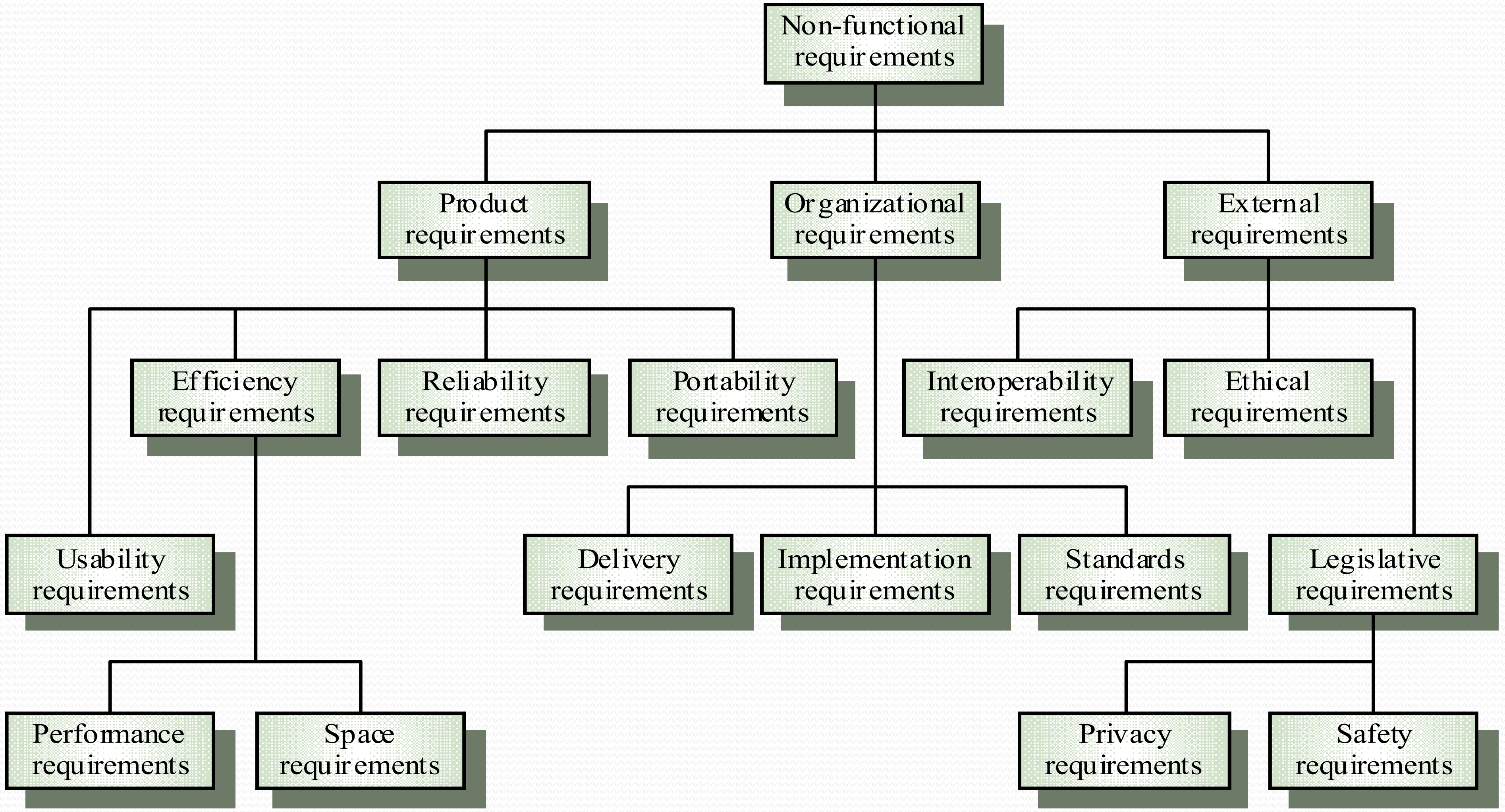
Non-Functional Requirements

- **Define system properties and constraints e.g.** reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc. They are often emergent properties of the system.
- **Process requirements** may also be specified, mandating a particular CASE system, programming language or development method
- **Non-functional requirements** may be more critical than functional requirements. If these are not met, the system is useless (e.g. key length for encrypting secure email must be >=256 bits)

Non-Functional Classifications

- **Product requirements**
  - Requirements which specify that the **delivered product must behave in a particular way** e.g. execution speed, reliability, security etc.
- **Organisational requirements**
  - Requirements which are a **consequence of organisational policies and procedures** e.g. process standards used, implementation requirements, etc. (Java as programming language)
- **External requirements**
  - Requirements which arise from factors which are **external to the system and its development process** e.g. interoperability requirements, legislative requirements, etc. (Must conform to FIPS

Non-Functional Requirement Types



Non-Functional Requirements Examples

- Product requirement
  - All encryption should use the Advanced Encryption Standard
- Organisational requirement
  - The system development process and deliverable documents shall conform to the process and deliverables defined in coding and documentation standard XYZCo-SP-STAN-95
- External requirement
  - The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system
- Performance requirement
  - The system should respond to a user’s request for information in less than 0.1 seconds during “peak-time” and 0.01 seconds during “normal time”.

Goals and Requirements

- **Non-functional requirements** may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- **Verifiable non-functional requirement**
  - A statement using some measure that can be objectively tested
- **Goal**
  - A general intention of the user such as ease of use
- Goals are helpful to developers as they convey the *intentions* of the system users

Examples

- **An example system goal**
  - The system should be easy to use by experienced controllers and should be organised in such a way that user errors are minimised.
- **An example <u>verifiable</u> non-functional requirement**
  - Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

Requirements Measures

Property	**Measure**
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM 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

Requirements Interaction

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

- Username/Password mechanism should be easy for user to user
- All passwords must be hard to guess and ideally require upper/lower case letters and special symbols to ensure high security

Which is the *most critical* requirement?

Example Domain Requirement

- Train Protection System:
  - The deceleration of the train shall be computed as:

$$\boxed{D_{train} = D_{control} + D_{gradient}}$$

where $D_{gradient}$ is $9.81ms^2$ * compensated gradient/alpha and where the values of $9.81ms^2$ /alpha are known for different types of train.

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 (e.g. consider the previous slide)
- **Implicitness**
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit which leads to problems later if software developer implements the requirements in the wrong way

User 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 in order that non-technical clients can better understand the requirements and point out potential problems.

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 in same document
- Requirements amalgamation
  - Several different requirements may be expressed together
  - Leads to problems with testing/debugging

Guidelines for Writing Requirements

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

  shall for mandatory requirements (that must be supported),

  should for desirable requirements (that are not essential).

Use ***text highlighting*** to identify key parts of the requirement

- Avoid the use of computer jargon
- Try and make documents **self contained** (e.g. include glossaries and complete examples)

Lecture Key Points

- Requirements set out what the system should do and define constraints on its operation and implementation
- Functional requirements set out services the system should provide
- Non-functional requirements constrain the system being developed or the development process
- User requirements are high-level statements of what the system should do