Lecture 3 – Requirements Engineering

Karl R. Wilcox
K.R.Wilcox@reading.ac.uk

Refresher

- This year we will look at all aspects of Software Engineering (all sections of Somerville)
  - But only at a high level
  - Revisit everything in more detail next year
- Week 1 – What is Software Engineering?
  - Make up your own mind(!)
- Week 2 – Software Process Models
  - The big picture of what happens in software development
- Today – Requirements Engineering
  - Early phase in the process

Objectives

- To introduce the concepts of user and system requirements
- To describe functional and non-functional requirements
- To explain some techniques for describing system requirements
- To explain how software requirements may be organised in a requirements document
- To carry out a small requirements investigation

Requirements Engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed
- The requirements themselves are 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 and contractor
- **Software specification**
  - A detailed software description which can serve as a basis for a design or implementation. Written for 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.
- **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

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 ‘appropriate viewers’
  - User intention - special purpose viewer for each different document type
  - Developer interpretation - Provide a text viewer that shows the contents of the document

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 impossible to produce a complete and consistent requirements document

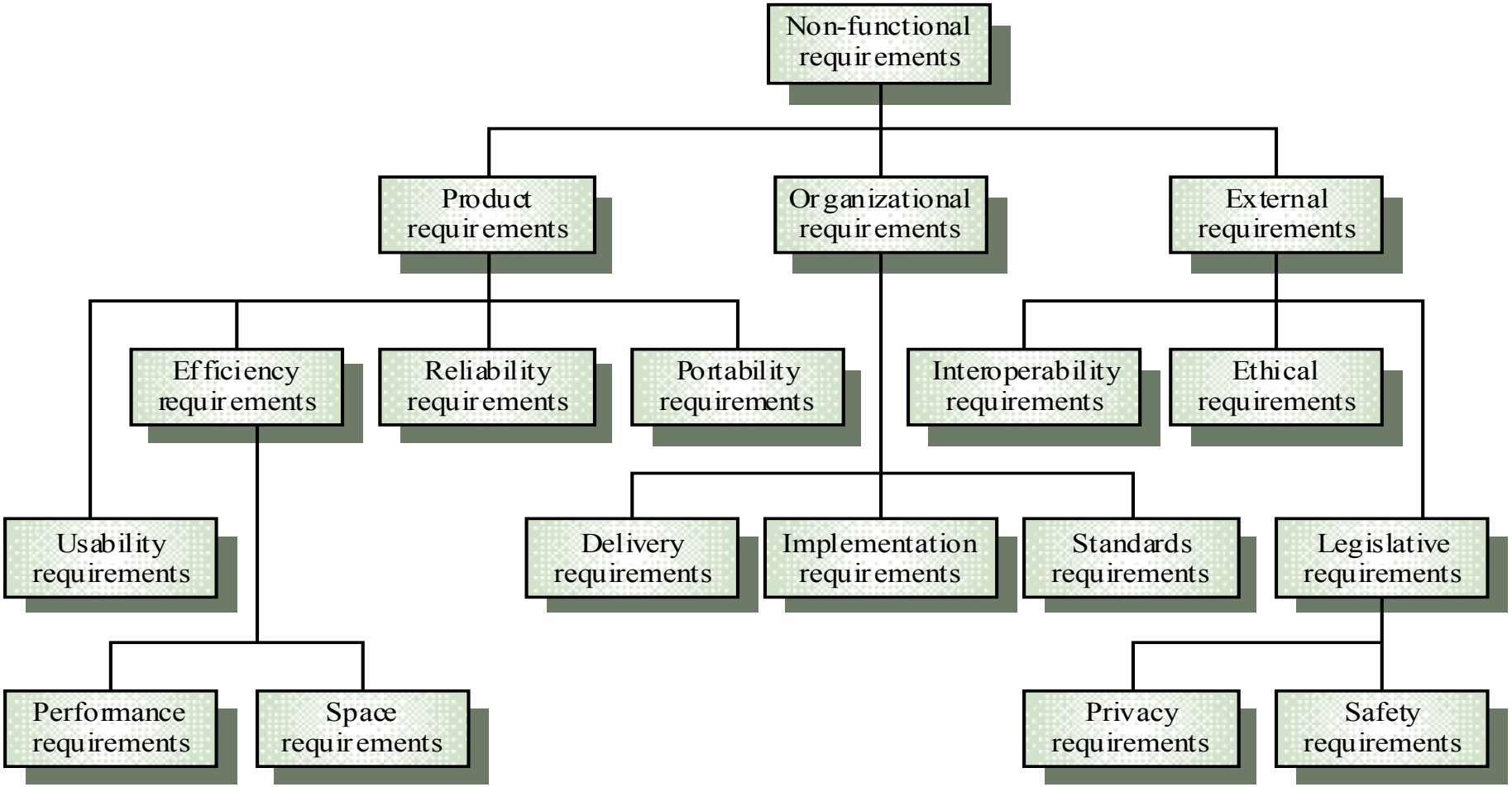
Non-functional requirements

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

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



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

Examples

- A 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.**
- A verifiable 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 interaction

- 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?

Domain requirements

- Derived from the application domain and describe system characteristics and features that reflect the domain
- May be new functional requirements, 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

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

Detailed User Requirement

3.5.1 Adding nodes to a design

3.5.1.1 **The editor shall provide a facility for users to add nodes of a specified type to their design.**

3.5.1.2 The sequence of actions to add a node should be as follows:

1. The user should select the type of node to be added.

2. The user should move the cursor to the approximate node position in the diagram and indicate that the node symbol should be added at that point.

3. The user should then drag the node symbol to its final position.

Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning.

Specification: ECLIPSE/WS/Tools/DE/FS. Section 3.5.1

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
- Requirements amalgamation
  - Several different requirements may be expressed together

Structured language specifications

- A limited form of natural language may be used to express requirements
- This removes some of the problems resulting from ambiguity and flexibility and imposes a degree of uniformity on a specification
- Often bast supported using a forms-based approach

Form-based Specifications

- Definition of the function or entity
- Description of inputs and where they come from
- Description of outputs and where they go to
- Indication of other entities required
- Pre and post conditions (if appropriate)
- The side effects (if any)

PDL-based requirements definition

- Requirements may be defined operationally using a language like a programming language but with more flexibility of expression
- Most appropriate in two situations
  - Where an operation is specified as a sequence of actions and the order is important
  - When hardware and software interfaces have to be specified
- Disadvantages are
  - The PDL may not be sufficiently expressive to define domain concepts
  - The specification will be taken as a design rather than a specification

PDL disadvantages

- PDL may not be sufficiently expressive to express the system functionality in an understandable way
- Notation is only understandable to people with programming language knowledge
- The requirement may be taken as a design specification rather than a model to help understand the system

The Requirements Document

- The requirements document is the official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

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

Requirements Document Structure

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

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

Key Points

- User requirements should be written in natural language, tables and diagrams
- System requirements are intended to communicate the functions that the system should provide
- System requirements may be written in structured natural language, a PDL or in a formal language
- A software requirements document is an agreed statement of the system requirements