Inf2: Software Engineering and Professional Practice
Lecture 4: Requirements Engineering

Cristina Adriana Alexandru

School of Informatics
University of Edinburgh

In week 1, on the SE part of this course . . .

- Objectives, motivation and structure of the Inf2C-SE course
- Why is Software Engineering still hard?
- Software engineering activities
- Brief history of SE
- Software project vs software product engineering
- Software development processes: plan-driven and agile

This lecture

- Requirements engineering
  - What is a requirement?
  - Kinds of requirements
  - Requirements vs. design
  - The concept of a stakeholder
  - Sub-activities of requirements engineering

What is a software requirement?

A software requirement is *"a property that must be exhibited by something in order to solve some problem in the real world"* (From SWEBOK V3, Ch1)

Requirements reflect the needs of different people at various levels of the organisation.

Requirements engineering is often used to describe the systematic handling of requirements

Kinds of requirements

Functional requirements (*services*): What the system should do.

Non-functional requirements (*constraints* or *quality reqs.*): How it should *be*: how fast it shuld be; how seldom it should fail; what standards it should conform to; how easy it is to use; etc. Informally called *"ilities"* because may refer to efficiency, security, portability, usability etc.

Kinds of requirements

Non-functional requirements may be more important than functional requirements!

- Can be workarounds for functional requirements
- User experience often shaped by non-functional.

Distinction not always clear-cut

- Security might initially be a non-functional requirement, but, when requirements refined, it might result in addition of authorisation functionality

Requirements vs. design

Requirements try to avoid design,

- expressing what is desired,
- not how what is desired should be realised

Stakeholders in requirements

Requirements are usually relevant to multiple stakeholders.

Stakeholders are *"any person or group who will be affected by the system, directly or indirectly"*.

(from Sommerville "'Software Engineering' chapter 4).

Stakeholders in requirements

Requirements are usually relevant to multiple stakeholders:

- End users
- Customers paying for software
- Government regulators
- System architects
- Software developers
- Software testers
- ...

Requirements engineering activities

- Gathering (*elicitation*)
- Sorting out (*analysis*)
- Writing down (*specification*)
- Checking (*validation*)

Activities often overlapping, not in strict sequence, and iterated.

Several approaches possible for each activity. Choice is very dependent on nature of software developed and overall software development process.

Requirements engineering is critical

- Faulty requirement processes can have huge knock-on consequences in later software process activities.
  - It is the major source of project failure according to Standish CHAOS reports.
- One motivation for incremental nature of Agile processes.

Requirements elicitation sources

- Goals: high-level objectives of software
- Domain Knowledge: Essential for understanding requirements
- Stakeholders: Vital, but they may find expressing requirements difficult
- Business rules: E.g. Uni regulations for course registration
- Operational Environment: E.g. concerning timing and performance
- Organisational Environment: How does software fit with existing practices?

(From SWEBOK V3, Ch1)

Requirements elicitation techniques

Techniques include:

- Interviews
- Scenarios
- Prototypes
- Facilitated meetings
- Observation

Requirements elicitation: interviews

Traditional method: ask them what they want, or currently do

Can be challenging:

- Jargon confusing
- Interviewees omit information obvious to them

Important to

- be open minded: requirements may differ from those pre-conceived,
- prepare starting questions, e.g. from first-cut proposal for requirements. Helps to focus dialogue

Requirements elicitation: scenarios

Scenarios are typical possible interactions with the system

- Provide a context or framework for questions.
- Allow “what if” or “how would you do this” questions.
- Easy for stakeholders to relate to
- Can be captured as user stories and use cases

Requirements elicitation: prototypes

Can include

- screen mock-ups
- storyboards
- early versions of systems

Like scenarios, but more “real”. High quality feedback. Often help to resolve ambiguities.

Requirements elicitation: facilitated meetings

Get discussion going with multiple stakeholders in a structured manner, to refine requirements

Helps with:

- Requirements that are not about individual activities
- Surfacing / resolving conflicts

Needs a trained facilitator.

Requirements elicitation: observation

Suitable if replacing existing system or business process

- Nuances can make or break a software product

Immersive method. Expensive.

Helps with:

- Surfacing complex / subtle tasks and processes
- Finding the nuances that people never tell you

Not so good if innovating

- Consider 15 years ago capturing requirements for a touchscreen smartphone

Requirements analysis

Requirements elicitation often produces a set of requirements that

- contains *conflicts* (e.g., one stakeholder wants one-click access to data, another requires password protection)
- is *too large* for all requirements to be implemented.

Requirements analysis is the process of getting to a single *consistent* set of requirements, *classified* and *prioritised* usefully, that will *actually be implemented*.

Requirements specification

Requirements can be recorded in various ways, perhaps using:

- very informal means e.g. handwritten cards in user stories in agile development
- a document written in careful structured English, e.g.

  `3.1.4.4 The system shall...` *for an essential feature*
  `3.1.4.5 The system should...` *for a desirable feature*

- use case models with supporting text
- a formal specification in a mathematically-based language.

Requirements validation

Checks include:

- Consistency checks
- Completeness checks
- Realism checks:
  - can requirements be met using time and money budgets?
- Verifiability:
  - is it possible to test that each requirement is met?
  - Applies to both functional and non-functional requirements. Non functional requirements must be measurable.

    Response time must be under 1 sec,
    not *Performance must be good*

Reading

SWEBOK V3, Chapter 1, Software Requirements.

Sommerville SE, Part 1 chapter on Requirements Engineering.

Browse for definitions of different non-functional requirements. You may want to start from Wikipedia.