Requirements Engineering

This is the first step in software engineering, and is a large and important subject.

System

A system is a collection of components which co-operate in an organised way to achieve some desired result.

- Components may be machine, software, or human (e.g. air traffic controllers)
- The desired result is specified by the requirements.

Requirements

A requirement is a statement which identifies an attribute, capability, characteristic, or quality of a system that is necessary for it to have value or utility.

Stakeholder

A person (or group of people) with a direct or indirect interest/stake in the system.

- Different stakeholders usually have different concerns and so will give rise to different requirements.

Classifying Requirements

SMART Requirements

Identifying Requirements

Problem 1: customers don't know what they want.

In market-targeted software, marketing department serves as a customer proxy, but may still not know what they want.

Problem 2: Requirements change.

Direct Stakeholder Interaction

1. Interviews
    - open or closed interviews with stakeholders to understand their needs
        - open or closed questions (closed are yes/no answers, open are answers that can be anything)
    - can ask e.g. “describe your day-to-day job”
    - need to interview a lot of people
    - time-intensive
2. Questionnaires
    - open or closed questions that are well-defined
    - typically online now
    - asynchronous, so it may take time to get answers (unlike interviews)
    - can reach a large audience
    - no follow-up (e.g. if you don’t understand answers)
3. Brainstorming
    - informal discussion in group setting with different stakeholders to capture as many ideas as possible
    - good for generating ideas
    - need to overcome people’s shyness
4. Focus groups
    - group discussion that is led by a moderator following a structured approach
    - moderator <- difference with brainstorming

Domain Analysis

Study the application domain to understand typical features. Figure out main concepts and work out how they fit together.

- May be useful for first graded task (e-voting)

Sources:

- Existing documentation/research
- Legacy systems
- Reusable concepts & components

Example: developing a compiler – first compiler took a lot of effort, but nowadays we know (e.g.) what components we need

User Stories & Scenarios

User Stories

A user story is a brief statement that identifies a user and his/her need.

`* originated in Extreme Programming`

Template: "As a I want so that "

Example: "As a user, I want to upload photos, so that I can share photos with others."

Example: " As an administrator, I want to approve photos before they are posted, so that I can make sure they are appropriate."

User stories are very common in agile programming methods.

Scenario

A scenario is a real-life example that illustrates a concrete system interaction. More extensive than a user story.

Includes:

- initial assumptions & expectations
- normal flow of events
- exceptions and errors
- other parallel/background activities
- system state after scenario is finished
  - e.g. database system – once a user has been added there should be one more record than there was

Ethnography

The researcher is immersed in a culture (e.g. a hospital), taking the point of view of the study subject. Good for very complicated systems.

Develop understanding through:

- Interviews
- Observation
- Participation
- Longitudinal immersion

Prototyping

Can be good for early feedback, checking what the customer might want. Prototyping is another agile method, and we'll be talking about it again.

- See reading.

Prototype based on:

- Preliminary requirements
- Existing examples/similar systems

Useful for:

- User-interfaces
- Greenfield development
- Overcome IKIWISI syndrome (I know it when I see it)

Summary

No one way is the best way, it depends on the context. The right approach is the one that works.

Requirement Specification

SRS Structure

- Preface
  - define readership, version history
- Introduction
  - context and need for system including business/strategic objectives
- Glossary
  - Define technical terms used in document
- User requirements
  - Services / functionality provided to user
- System architecture
  - High level overview of expected system architecture
- System requirements specification

[...]

Requirement Notations

Can be expressed in various ways:

- Natural language
- Structured language
- Graphical notations

There are others we won't discuss, e.g.:

- Formal languages (e.g. Z)
  - Used in specific domains, e.g. embedded

Whatever notation, S.M.A.R.T. requirements are good.

Natural Language

Expressive and universal, but also potentially vageu and ambiguous.

Suggested guidelines:

- standardised format

[...]

Structured Language

Natural language with a template, e.g. specific categories like Inputs, Source, Precondition, Postcondition, Description.

Can also use tables, which may be good in very narrow, computational domains.

Graphical Notation

Best known is UML, unified modelling language.

- In the 80s, there were many different methods, and UML is the standard which was created later.

UML version 1.0 in 1997.

As UML originated in methods for OO analysis and design, so it's strongly focused on OO.

- Not as useful e.g. for C or FORTRAN

It is suitable for:

- Visualising
- Specifying
- Constructing
- Documenting (all artefacts, including requirements)

UML is a standard, but most people don't use it formally, just choose bits of it they want.

Requirement Prioritisation

Simple Ranking

Rank requirements in order of decreasing priority.

MoSCoW

Group:

- Must have
- Should have
- Could have
- Won't have (this time)
  - Would like, but can't do now

Planning Poker

Comes from Extreme Programming (discussed later).

Simple way to reach consensus on effort estimation.

A team estimate the effort from 1 to 100, and come to an agreement/consensus.

- Can be interesting if different team members give very different effort values.

When 2 features have the same value, prioritise the one with the least effort.

$100 Method

Give all stakeholders an imaginary $100.

They need to distribute these over the requirements – higher ones get higher value.

For each requirements count the sums – put the results in prioritised order.

Bubble Sort

Sort the requirements using bubble sort (according to their priority).