Principles of Software Design
Requirements

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Why do we need requirements

- Input for design and implementation

Why do we need requirements

- Input for design and implementation
- Preparing contract
- Estimation, project management
- Verification
- The basis for the documentation / orientation in the project
- Risk management
- ...

All stakeholders work with requirements

Source of requirements

- Customer

Source of requirements

- Customer
- Domain knowledge
- Other stakeholders (architectural requirements, implementation requirements, testing requirements, ...)
- Regulations and law
- Deployment conditions
- Organizational structure / Internal regulations of the customer
- ...

Hierarchy of requirements

- Business requirements

  We want to decrease paper consumption . . .

- User requirements

  After finishing work on . . . the following parties will be informed: . . .

- System requirements

  Upon signature the document will be handled to . . . using . . .

Traceability - it is important to know the relations between requirements (especially between requirements in different levels).

Hierarchy of requirements

- Business requirements

  We want to decrease paper consumption . . .

- User requirements

  After finishing work on . . . the following parties will be informed: . . .

- System requirements

  Upon signature the document will be handled to . . . using . . .

Traceability - it is important to know the relations between requirements (especially between requirements in different levels).

- Which requirements should we derive the contract from?

Requirement types

- Process requirements
- Product requirements
  - Functional requirements
  - Non-functional requirements

Requirement types

There are various frameworks to categorize requirements. This is useful so we can be sure that each category is covered.

- An example

Properties of good requirements

- Unitary (Cohesive)
- Complete
- Consistent
- Non-Conjugated
- Traceable
- Up to date
- Unambiguous
- Understandable
- Specified priority
- **Testable**

Focus on WHAT, not HOW (however, sometimes, especially when describing a desired process WHAT=HOW)

Properties of good requirements

- **Complete** (for given subsystem)
  - from all stakeholders
  - no gray areas - includes non-requirements, e.g. what is not part of the system
  - Potentially useful tools: templates (IEEE830), classifications.

Examples of requirements and its implications

We want to decrease the paper consumption . . .

- Business requirement, the source is the customer.
- Should be quantified, e.g. “by . . . %”
- Priority?

Examples of requirements and its implications

We want to decrease the paper consumption . . .

- Business requirement, the source is the customer.
- Should be quantified, e.g. “by . . . %”
- Priority?
- Is it testable?

Examples of requirements and its implications

We want to decrease the paper consumption . . .

- Business requirement, the source is the customer.
- Should be quantified, e.g. “by . . . %”
- Priority?
- Is it testable? You need to create a system to measure this. The quality may depend on the priority.

Examples of requirements and its implications

After finishing work on . . . the following parties will be informed:

. . .

- User requirement - we should know which business requirements it is related with.
- We need a dictionary for the project to be able to specify the type of the document and the parties involved.
- How fast the parties should be notified? Can we measure it?
- Can anything go wrong during the process (of significant importance to be an user requirement)?

Examples of requirements and its implications

Upon signature the document will be handled to . . . using . . . .

- How long should it take?
- What happens if the document will not be signed (+ several other exceptional workflows).

How to capture the requirements

How to capture the requirements

- "Victorian novel"
- Using a template
- Spreadsheet / other form of list
- Issue tracking
- Minutes from a meeting (only short time validity, it should be processed into a more robust form) ...

Challenges:

- How much documentation we want to create **and maintain**?
- How to preserve traceability?
- How to find related requirements?
- What is the state of the requirement?

Capturing functional requirements

The most common way of organizing functional requirements are **use cases**. The definitions vary a lot:

- A list of actions or event steps typically defining the interactions between an actor role and a system to achieve a goal. [8]
- A use case is all the ways of using a system to achieve a particular goal for a particular user. Taken together the set of all the use cases gives you all of the useful ways to use the system, and illustrates the value that it will provide. [9]
- ...

Authors disagree mostly on how much value is required for something to be an use case.

Use case

A full use case contains:

- Actor (human or external system)
- A goal - something of a value to the actor
- A complete sequence of steps how to attain the goal **including alternative paths**
- A lot of other stuff

Use cases can be included, extended and generalized.

How to capture functional requirements

Various amount of detail:

- Full use case
- Scenarios - A pass through the use case.
  - We use several scenarios per use case.
- User story - A sentence, often following certain template. Possible template [10]:

  As a <role> I can <capability>, so that <receive benefit>.

How to capture functional requirements

Various amount of detail:

- Full use case
- Scenarios - A pass through the use case.
  - We use several scenarios per use case.
- User story - A sentence, often following certain template. Possible template [10]:

  As a <role> I can <capability>, so that <receive benefit>.

There is a certain degree of freedom on what is an use case. E.g. if you use user stories, it make sense to use more fine-grained use cases.

Use cases - size [7]

Use cases may have various scope

- Organization (Business use case)
- System
- Component

And goals may have various levels

- Very high summary
- Summary
- User goal
- Subfunction
- Too Low

Use cases - usage

- Preferred level is sea level.
- Sometimes, you need higher level use cases (to make a simple view of the system, as a placeholder before eliciting details, to initiate discussion . . .)
  - Such use cases will be very often abstract - it will present a composition of several other use cases.
  - When you use the form of user stories, such stories are called **epics**. You need to split them up before implementing them.
- The (non-abstract) use case (at least the main flow) should not bee too long- otherwise it may be hard to comprehend.

Use cases - usage

- You want to be able to implement a use case in one iteration.
- For subfunction level, it is sometimes very questionable if the use case presents sufficient value to the actor.
- Keeping use cases that are too low level up to date is tedious. You do not want to be too clever here (too much abstraction / use case composition) because you want to keep things simple (use cases are used to communicate with the customer). As a result certain degree of repetition is inevitable which makes managing such use cases particularly tedious.

Requirements workflow

- Stakeholder identification
- Elicitation
- Analysis
- Specification
- Validation

The process is iterative. During each phase a dictionary is created and refined.

Elicitation

- Discussions
- Focus Groups
- Questionnaires
- Observation
- Studying documentation, legislature ...
- Study of similar products
- ...

Analysis and specification

- Actor identification and modeling (including new abstract roles)
- Prototyping
- Traceability matrix
- Avoid requirements smells
  - Subjective language
  - Nebulous adjectives, superlatives
  - Negative requirements ($\neq$ Non-requirements)
  - ...
- Careful write-up helps to identify weak spots
  - Use case analysis -searching for alternative scenarios
  - Generalization

Common problems capturing the use cases

The structure of a use case is too complex?

- UML activity diagram

The use case is repetitive / too long?

- We can divide it and use use-case composition.
- Creates dependencies between your use cases.
- Do not overthink it, use cases need to have simple structure

To give an overview of dependencies between actors and use-cases:

- UML use case diagram

Resources I

- Wikipedia - Requirement
- J. Mifsud: Requirements Gathering Part 1
- Techpedia - Use case
- Wikipedia - Use case example
- UML Use Case Diagrams
- UML Activities

References I

- SWEBOK V3 - Chapter 1
- Wikipedia - Requirement
- Wikipedia - Requirement analysis
- J. Kostičová: Requirements
- J. Mifsud: Requirements Gathering
- uml-diagrams.org
- A. Cockburn - Use case template
- A. Cockburn: Writing Effective Use Cases, Addison-Wesley, 2001
- I. Jakobson: Use Case 2.0

References II

 Wikipedia - User Story