ES Software Engineering

Lecture 2
Requirements specification, testing and defects

Previous lecture

1. Embedded systems today:
    1. Multiprocessor systems and systems of systems
2. Software lifecycle:
    1. Waterfall model
    2. Spiral model
    3. RUP

Plan of the lecture

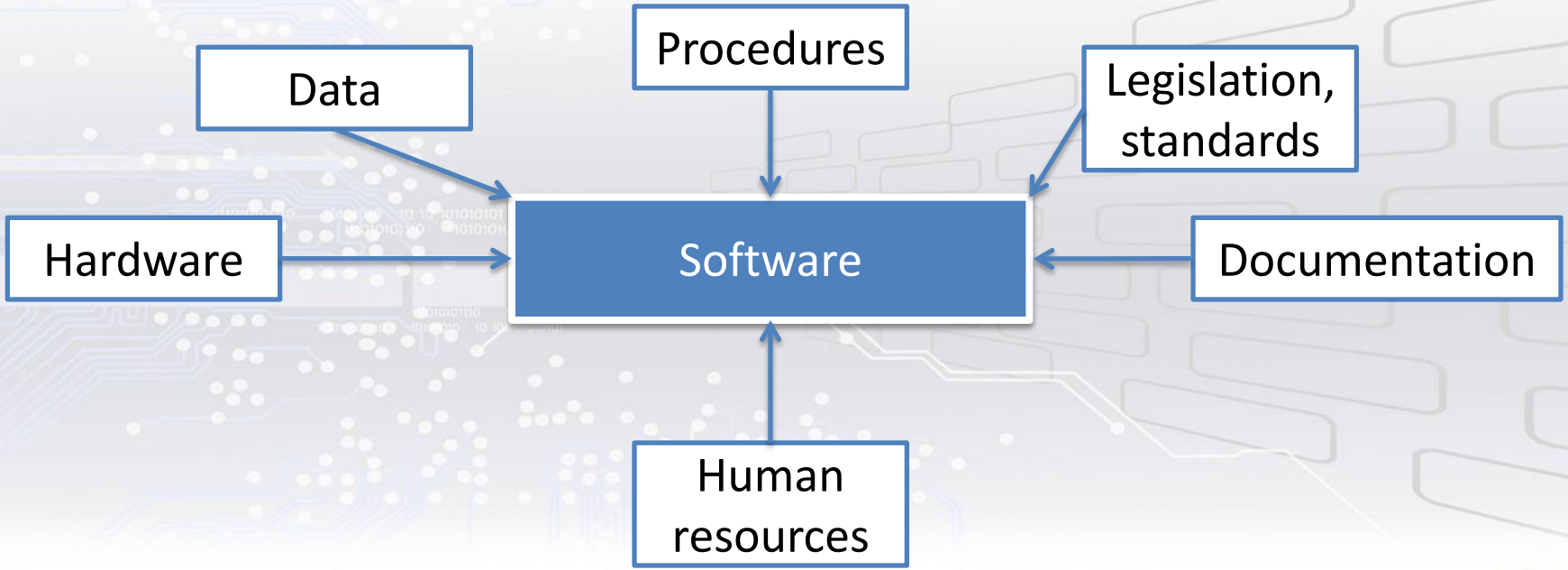
1. Requirements engineering – definition
2. System Requirements Specification
3. Requirements defects
4. Search technique to find requirements defects
5. Examples of defects

Requirements engineering

1. Regardless of the decided software development strategy – the first stage of the process is to define system requirements.
2. Mistakes at this stage significantly extend implementation time of the final product.
3. By design – requirements are not constant, both during implementing the system and during maintenance.
4. Because of point 3. requirements engineering constitutes separate issues of software development.

Requirements engineering

Software requirements aspects:



Requirements engineering

Requirements engineering covers:

1. Determining project goals:
   1. Business goals define customers' benefits reached through implementation of the project
   2. Purpose defines main functions to serve business goals
2. Extracting requirements:
   1. Functional requirements
   2. Non-functional requirements (data, quality, safety etc.)
3. Agreement on rules for product acceptance

Requirements source

Stakeholders:

- Business goals (customers and management)
- Functionality (end users)
- Community, stuff (other people)

Non personal

- Legislation and formal standards
- Informal standards
- Hardware

Stakeholders

- Almost each system is a multidisciplinary system.
- The largest entry work is to develop a common language.
- The problem of communicating with a client from the other end of the world.
- Man as the weakest element.
- A contractor does not appreciate the role of an extensive entry documentation.

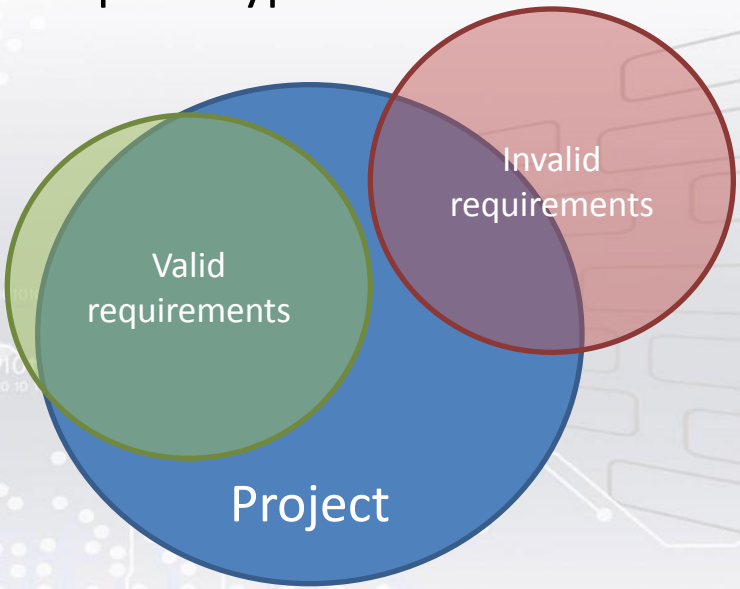
Requirements extraction

- Realization of individual needs by shareholders
- Needs statement
- Transformation of the needs to system requirements
- Information and descriptions needed to understand the requirements
  - Text and diagram description
  - Formulas, charts
  - Combined description
  - Numbering requirements (with division to shareholders) – possibility of referring to them

Results

What is the most important in a prototype:

- Fulfilling requirements?
- 100% of functionality?
- Legislation?
- DEADLINE



System Requirements

System Requirements Specification:

1. Introduction
2. Basic Information
3. Functional Requirements
4. Non-functional Requirements
5. Acceptation Criteria
6. References
7. Appendices

System Requirements

Ad. 1. Introduction:

1. Project identification
2. Short project description
3. Main project goal
4. Foundation
5. System vision – may be described in a separate document

System Requirements

Ad. 2. Basic information:

1. Prioritized Requirement Sources
    1. Stakeholders
    2. other
2. Prioritized Project Objectives
    1. Business goals determined by management
    2. Purpose determined by end users
3. System Context
    1. User roles
    2. External Systems
4. Infrastructure conception (for complex systems)
    1. Subsystems
    2. Components

System Requirements

Ad. 3. Functional Requirements:

1. System function description
2. Organization of requirements by users and by components
3. Requirement properties
    1. Text description
    2. Applies to users or an external system
    3. Source of the requirement
    4. Priority

Ad. 4. Non-functional Requirements

1. Data Requirements
2. Quality Requirements
3. Other Requirements

System Requirements

Ad. 4. Non-functional requirements

Data:

- Main data concepts
- Data terms explanation
- Source of explanation
- Priority

System Requirements

Ad. 4. Non-functional requirements

Quality:

- Reliability (safety, security, error tolerance)
- Performance (execution efficiency, interaction efficiency)
- Flexibility (portability, stability, scalability)
- Usability (learnability, understandability, operability)

System Requirements

Ad. 4. Non-functional requirements

Other:

- Hardware constraints
- Software constraint
- Extraordinary situations (exceptional, critical, failover situations)
- Other

System Requirements

Ad. 5. Acceptation criteria

1. Tests
2. Trial time
3. Maintenance
4. Specific constraints

System Requirements

Ad. 5. References

1. System vision document
2. Technical documentation
3. Articles
4. Standards
5. Existing source code

System Requirements

Ad. 6. References

1. Glossary
2. Tabular data
3. Diagrams

Ad. 7. Appendices

1. Glossary
2. Tabular data
3. Diagrams

Requirements defects

Defects – results of mistakes during creating a product

Sources of mistakes:

- Lack of knowledge
- Lack of attention
- Schedule pressure

How to identify requirements defects to avoid their propagation into the software.

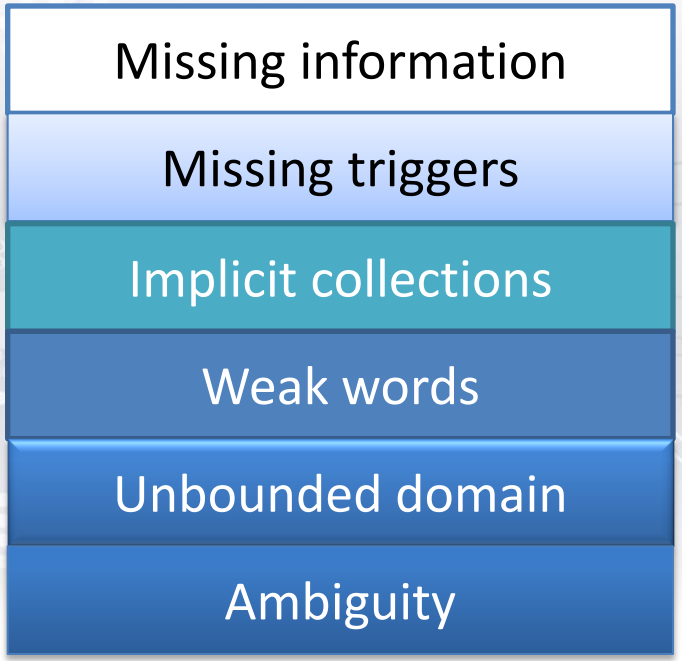
Requirements defects

- Costs caused by requirements defects = 70% of total defects costs [7]
- Requirements defects consume up to 40% of a total budget of many projects [7]
- Poor requirements are 5 of the top 8 reasons for a project failure and accounted for 41-56% of discovered errors [6],[8]
- 80% of all product defects have their reason at the level of defining requirements [9]
- Because of requirements defects – 45% of the developed functions are never used

Phase	Relative Cost
Inspection	1
Design	10x
Testing	25x
Production	>=100x

Requirements defects sources

Sources of requirements defects [6]:



Requirements defects sources

Missing information — everything reader must guess

E.g.

1. Software should run a test according to RRT strategy.

2. Software architecture should be based on our company classes.

Requirements defects sources

Missing triggers

Almost all requirements have events or states needed for them to execute

E.g.
System should display an error message. At which states should it be done?

Requirements defects sources

Implicit collections

Imperfect definitions, not explicit concepts

E.g.

System should operate at 1200 bps, 1400 bps, 1800bps and other speeds.

It should communicate using SPI, I2C, RS and other protocols available under eCOS.

Requirements defects sources

Weak words

Non-precise definitions and designations

E.g.

quick, easy, fast, timely, frequent, often, intuitive, normal, secure, user-friendly, immediate

Requirements defects sources

Unbounded domain

Lack of starting and/or end point

E.g.

at least, such as, or later, including but not limited to

1. System should process data from at least one pair of cores.

2. This part of software should be coded after preparing hardware specification or later.

Requirements defects sources

Ambiguity

Statements or words with multiple meanings

E.g.

Subjectivity, incompleteness, optionality, over-generalization, passive voice, incomplete logic

The method is determined by the user.

Find Requirements Defects

Search technique to find requirements defects:

1. Common Requirements Syntax (CRS)
2. Checklist for Well-Written Requirements (WWR)
3. Ambiguity Checklist
4. Test for Missing Triggers
5. Checking Non-Functional Requirements Testability

Find Requirements Defects

Ad.1. Common Requirements Syntax

[Trigger] [Precondition] Actor Action [Object]

E.g.

*If a data frame was received, after detecting low power the system **must** turn off the transmitter.*

[Trigger]: data frame received

[Precondition]: detection of low power

Actor: system

Action: turn off

[Object]: transmitter

Find Requirements Defects

Ad.2. Checklist for Well-Written Requirements

WWR must be:

1. Complete – sufficient detail
2. Correct – checked by stakeholders and Subject Matter Experts (SME)
3. Concise – only the needed information expressed in few words
4. Consistent – requirements must not conflict with any other requirement
5. Feasible – there is at least one implementation of requirements
6. Necessary – e.g. new product / competitive / customer's needs
7. Prioritized – ordered according to its importance
8. Traceable – identified with a tag and uniquely
9. Unambiguous – single interpretation
10. Verifiable – via demonstration, analysis, inspection, testing

Find Requirements Defects

Ad.3. Ambiguity checklist

1. Vagueness – weak inaccurate words
2. Subjectivity – weak words rely on a personal opinion
3. Optionality – wrong: should, may, if possible; proper: shall, must
4. Over-generalization – all, every, users etc.
5. Non-intelligibility – poor grammar, complex logic, "and" & "or" ambiguity
6. Passive voice – requirement does not name an actor
7. Incomplete logic – missing "else" of "if" loop
8. Time-logic ambiguity – confusion between a logical condition and the time domain

Find Requirements Defects

Ad.4. Test for Missing Triggers

The test is about asking questions concerning triggers or preconditions for all requirements which do not have defined triggers.

Find Requirements Defects

Ad.5. Checking Non-Functional Requirements Testability

NFRs are testable if they contain:

- scale – of measure used to quantify the statement
- meter – process used to establish location on a scale
- goal – minimum level on a scale required for a success

Find Requirements Defects

1st Example

"If a user could not report a ticket, the system should notify the administrator."

1. CRS (T)
2. WWR (F) not complete
3. Ambiguity checklist (F) time-logic confusion, incompleteness
4. Missing triggers (T)
5. NFR testability (NA)

Find Requirements Defects

2nd Example

"The system should return at least 3 possible answers for each question."

1. CRS (T)
2. WWR (F) not feasible, not verifiable
3. Ambiguity checklist (F) incompleteness, over-generalization
4. Missing triggers (F)
5. NFR testability (NA)

Find Requirements Defects

3rd Example

"When the third option is chosen – the system must quickly generate a report of the current position."

1. CRS (T)
2. WWR (F) not correct, not verifiable
3. Ambiguity checklist (F) incompleteness, subjectivity, vagueness
4. Missing triggers (F) quickly
5. NFR testability (T)

Find Requirements Defects

4th Example

"Program should be run while data are transferred from server"

1. CRS (F) location of "while data"
2. WWR (F) not complete, not correct
3. Ambiguity checklist (F) passive voice, optionality vagueness
4. Missing triggers (T)
5. NFR testability (T)

Bibliography

[1] Jarosław Kuchta, “Embedded Systems Software Engineering: Requirements Engineering”

[2] Roger S. Pressman, “Software Engineering. A Practitioner's Approach”

[3] USA Department of Defense, “Systems Engineering Fundamentals”

[4] Klaus Pohl, Chris Rupp, “Requirements Engineering Fundamentals”

[5] IREB (International Requirements Engineering Board) Downloads

[6] John Terzakis, “Identifying Requirements Defects”, Intel Corporation, 2013

[7] Leffingwell & Widrig, 2003

[8] The CHAOS Report, 1995

[9] Hooks and Farry, 2001

[10] Standish Group Report, 1995