Assignment 1: Reminder

- Deadline: Today, 23:59
- Individual

- Preference: PDF
- How to submit: Peach
  - http://peach.win.tue.nl/

Assignment 2
- Is already open
- Deadline: February 22, 23:59
- Individual

Sources

Chapter 6
Requirements Engineering and Feedback

Chapter 8
Software Evolution and Feedback

Why requirements?

- Errors in requirements are:
  - common:
    - 25% of all the errors (Jones '91)
    - 1 error per function point (~ 80 LOC Java; Jones '95)
  - expensive

Boehm, Papaccio 1998

Questions

- Why do the requirements evolve?
- How suited is a given requirements document for evolution?
- How do the requirements evolve and co-evolve?
Why do the requirements evolve?

- Environment changes
- [Nanda, Madhavji]
- Congruence Evaluation System
- Research prototype

![](image1.png)

Suitability for evolution

- When is evolution difficult?
  - Per requirement:
    - "Bad requirements": vague, subjective, weak, underspecified, overtly complex, unreadable...
    - Volatile requirements
    - Related to dependencies between the requirements
  - Per requirements document:
    - Missing requirements
    - Inconsistent requirements

Bad requirements: Check lists

- Industrial approach: guidelines, checklists, templates
- Manual verification
  - "Each requirement is testable" (SMART)
  - "Each requirement should state consequences of losses of availability and breaches of security" (European eGovernment program)
- Assessment
  - is subjective
  - requires training and experience

Bad requirements: More automation?

<table>
<thead>
<tr>
<th>Problem</th>
<th>Indicators (examples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vagueness</td>
<td>clear, significant, useful, adequate, good, bad</td>
</tr>
<tr>
<td>Subjectivity</td>
<td>similar, as ... as possible, taking ... into account</td>
</tr>
<tr>
<td>Optionality</td>
<td>possibly, if needed, if appropriate, eventually</td>
</tr>
<tr>
<td>Weakness</td>
<td>could, might</td>
</tr>
<tr>
<td>Underspecification</td>
<td>(write/read) access, (data/control) flow, &quot;TBD&quot;</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>and, or</td>
</tr>
<tr>
<td>Implicity</td>
<td>Anaphora (it, these, previous, above)</td>
</tr>
</tbody>
</table>

Bad requirements: More automation?

Problem Indicators (examples)

Vagueness: clear, significant, useful, adequate, good, bad
Subjectivity: similar, as...as possible, taking...into account
Optionality: possibly, if needed, if appropriate, eventually
Weakness: could, might
Underspecification: (write/read) access, (data/control) flow, "TBD"
Multiplicity: and, or
Implicity: Anaphora (it, these, previous, above)


How to make the requirements better?

Readability measurement

- Requirements from a student project (Horus 2007)
  A. An account is an administrator account, a scientist account or an observer account.
  B. An administrator shall be able to configure whether multiple experiments may be executed simultaneously on a particular satellite.
  C. Experiment data can be retrieved from the system.
- Which one is more difficult to read?
  - Flesch-Kincaid grade level:
Flesch-Kincaid grade level

- Number of years of (US) education required to understand the text.

\[0.39 \left( \frac{\text{total words}}{\text{total sentences}} \right) + 11.8 \left( \frac{\text{total syllables}}{\text{total words}} \right) - 15.59\]

- Might be misleading for smaller texts
  - Use for features instead of individual requirements
  - Feature = group of related requirements
  - The NASA study (Wilson, Rosenberg, Hyatt 1997)
  - Mean = 10.76, std dev = 1.59

Second problem: Volatile requirements

How can we identify volatile requirements?

- Feature = group of related requirements
- Features should separate more and less stable requirements
- Intuition: More stable requirements should not depend on less stable ones
- Problems:
  - We cannot predict future change
  - Analysis of natural language requirements is difficult

Relative stability

- “Relative stability” – one requirement is more stable (>) than another one.
- Example: meeting schedule system
  - Notify the participants vs. Notify the participants by SMS
  - Intention or concept > operation or fact
  - Notify the participants vs. Arrange recurrent meetings
  - Core (in any variant/revision) > others
  - Notify the participants vs. The system should have an MS Windows “look and feel”
  - Functional > non-functional
  - NB: Conflicts and choices among different options are usually quite volatile

How can we address volatility?

- At requirements engineering time:
  - Try to find a more stable alternative
  - Put special attention to traceability (backwards – rationale, forwards – design, implementation, tests)
  - Anticipate and record responses for future changes
- At design time:
  - Encapsulate volatile requirements in separate modules

Volatility and dependencies

- “More stable requirements should not depend on less stable ones”
- A affects B (B depends on A) if changing A might require changing B.
Types of dependencies (examples)

- Use
  - A explicitly refers to B
  - “IMSETY shall adhere to Table 2.1 for user rights”

- Generalization/refinement
  - A is a more general case of B
  - “Observers shall not be authorized to manipulate experiments.”

<table>
<thead>
<tr>
<th>Entity</th>
<th>Analyze</th>
<th>Scientist</th>
<th>Observer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload</td>
<td>CB/ED</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>Scientist</td>
<td>CB/ED</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>User</td>
<td>CB/ED</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>Payload command list</td>
<td>CB/ED</td>
<td>E</td>
<td>-</td>
</tr>
<tr>
<td>Experiment</td>
<td>CB/ED</td>
<td>CB/ED</td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>BD/ED</td>
<td>R</td>
<td>BD/ED</td>
</tr>
<tr>
<td>Representation</td>
<td>-</td>
<td>R</td>
<td>CB/ED</td>
</tr>
</tbody>
</table>

Table 2.1: CB/ED matrix

Types of dependencies (examples, continued)

- Temporal
  - Satisfaction of A should precede/follow satisfaction of B
  - “IMSETY shall require users to be logged in before they can use any of the system’s functionality.”

- Satisfiability
  - Satisfaction of A implies satisfaction of B
  - More general than “generalization/refinement”
  - But also
    - “The system shall interface with an MCS for communication with satellites.”
    - “IMSETY shall log all communications with the MCSes.”

How can we derive dependency relations?

- Manually
- If requirements are formalized (à la RADV) – formal techniques, e.g., using model checking
- Using transitivity

Keyword-based [Huffman Hayes, Dekhtyar, Osborne 2003]:
- At any moment, only one scientist is allowed to compose an experiment on a single payload.
- A scientist shall be able to request the scheduling of the execution of experiments on a predefined moment.

Problems (and solutions) with keyword-based

- Terminology vs. regular use
  - solution: domain dictionary
- Synonyms (solution: thesaurus)
- Some requirements are more “essential” for the keyword than others.
- Lexical correlation vs. dependency
  - Which type?

Dependency analysis

- Traceability graph
  - Vertices: requirements
  - Arcs: dependency relations
- What do you think about the requirements document right?
- What does this mean for evolution?
- How would the quality information influence your interpretation?
- What are the limitations of the traceability graph approach?

Inconsistent requirements

- Different kinds of requirements: use cases, process models, natural language requirements
- Different sources of requirements: multiple documents, multiple stakeholders
- Different types of inconsistencies:
  - Terminological (synonyms, different interpretations of the same term): data dictionary, glossary, ontology
  - Logical (strong) – conjunction of the requirements is false
  - Logical (weak) – under some condition conjunction of the requirements is false
Logical inconsistencies: Heuristics

- Logical inconsistencies [van Lamsweerde]
- Heuristic: scrutinize dependent requirements
- Heuristic: public availability vs. confidentiality
  - Grades should be publically available
  - Students should not have access to other students’ grades
- Heuristic: increases vs. decreases
  - Increase access to books and journals
  - Reduce operational costs
- Heuristic: security vs. user-friendliness

Inconsistent requirements: Linguistics+Logics

- CARL [Gervasi, Zowghi 2005]:
  - Translate nat. lang. requirements to logical formulae
  - Analyse consistency

CARL: From text to formula

CARL: Detect inconsistency

1. A medical emergency is either an illness or an accident.
2. When an operator receives a phone call concerning a medical emergency, (s)he should dispatch a nearby available ambulance.
3. When an operator receives a phone call concerning a nonmedical emergency, the operator should not dispatch an ambulance, and he should transfer the phone call to another service.
4. When an operator receives a phone call, if an ambulance is not nearby or not available, then the operator should not dispatch that ambulance.
5. When an operator receives a phone call, (s)he should dispatch a nearby available ambulance.

How can we address inconsistency?

- Weaken or drop one of the conflicting statements
- Specialize the requirement such that the conflict disappears.
  - When an operator receives a phone call concerning medical emergency, (s)he should dispatch a nearby available ambulance.
- For weak logical inconsistencies: avoid the condition
  - Logical (weak): under some condition conjunction of the requirements is false
- Evaluate different options and choose the “best” one
- NB: Source of volatility

Summary so far...

- Requirements evolution can be hindered by
  - “Bad requirements”: vague, subjective, weak, underspecified, overtly complex, unreadable...
- Volatile requirements
  - Related to dependencies between the requirements
- Missing requirements
- Inconsistent requirements
So far...

Next...

How good is requirements document v1?

Evolution and volatility

- Volatility = subject to change?
- Eight functions, different documents

- Mostly linear correlation
- Sublinear: F2, F8
- Superlinear: F5
- Most likely to change

F1 seems to be stable

Closer look at the 8 functions (1)

- F1 is indeed stable.
- F1 is about system architecture
- Conjecture: system architecture is stable

Closer look at the 8 functions (2)

- Different functions are likely to change at different times.

How do the requirements evolve?

- Are Lehman’s laws applicable?
- [Anderson, Felici 2000]: avionics software

Total number of requirements increases...

Lehman’s laws
- Continuing change
- Continuing growth

How do the req. evolve – what have we done?

- Calculated
  - the number of requirements
  - the number of changes
  - the cumulative number of changes
- Studied
  - how these values change with time
  - Last week something similar for size/complexity/…

- Generic approach
  - Metrics: function from software artefacts to numbers
  - Time series: sequence of measurements at successive times
So far...

How do the requirements evolve?

How good is the requirements document v1?

Next...

Co-evolution: Traceability links

(SEP)

Rationale, business needs

Design, implementation, tests

Horizontal dependency

Vertical dependency

Revision

Variant

Intra-version

Inter-version

Forward

Backward

Why do we need traceability?

[Wiegers 2003]

• (Co-)evolution
• Change impact analysis
• Maintenance
• Reengineering
• Certification
• Project tracking
• Reuse
• Risk reduction
• Testing

Backward dependencies [Robertson]

Traceability matrix

• Means of expressing traceability information

Two popular techniques

What are their advantages and disadvantages?

We need tools!

• Large amount of information:
  • requirements, components, traceability links
  • database technology!
• Many commercial/OS tools are available
  • OS requirements management tool: http://sourceforge.net/projects/osrmt/
  • http://requirements.tigris.org/
• You might like to try them!
Conclusions

• Requirements often evolve due to environmental changes
• Suitability for evolution: quality, volatility, dependencies, inconsistency
• Evolution:
  • Continuous change and growth
  • System architecture is “almost” stable
• Co-evolution
  • Need for backward and forward traceability

Assignment 2

• Organizational:
  • Is already open
  • Deadline: February 22, 23:59
  • Individual
  • Please submit in PDF

• Given a set of requirements you’ll assess their quality, study dependencies between them and discuss the implications of your findings on the requirements evolution.