Chapter 4.4: Software Application Engineering – Requirements Management
### Goals

- **The goals of this chapter are to be able to**
  - Understand how requirements can be managed
  - Prioritize requirements

### Literature

- Lecture "Requirement Engineering"
Motivation

Why changes?

- Changing requirements during the development
  → Instable requirements
    - initially incomplete ✓
    - badly understood ✓
    - insufficient development resources ⇐ prioritization!

- After delivery (maintenance)
  → Changes are 60-80% of the total life-time development cost

- Types of changes (during evolution = development & maintenance):
  → Missing functionality (requirements): 2/3 ✓
  → Tailoring to specific needs: 1/6 ✓
  → Defects: 1/6 ✓

Changes are central to the development & maintenance!
Motivation

- Change Types
  - Forgotten requirements
  - New requirements
  - Changed requirements (changed market, new customers, etc)

Requirements Management

- Managing changes to requirements
- Managing relationships
  - among requirements (Horizontal Traceability)
  - to other artifacts (Vertical Traceability)
- Prioritizing Requirements

Change management (CM)

- Change process
  - Change Control Board (CCB)
  - Scheduling
  - Prioritization

- Change documentation and tracking
  - Documenting origin, responsible, status, etc

- Change impact
  - Analyze initial change points & dependencies & cost, prioritizations, etc

Requirements engineering & management is a continuous activity throughout the entire development & maintenance life-cycle.
Motivation

Process

Challenge

Traceability

Priorization Methods
- Likert Scale Method
- Cumulative Voting Method
- Hierarchical Cumulative Voting
- Analytic Hierarchy Process
- Kano Model

### AE – Requirements Management

### Change Process

1. Identify change request(s)
2. Plan changes
   - Impact Analysis
   - Estimate cost
   - Decide on change
   - Plan release
3. Implement changes

---

© Prof. Dr. Dr. h. c. Dieter Rombach, Fundamentals of Software Engineering, Winter Term 2011/12
Why is change management difficult?

- Maintenance personnel may have to maintain a system they didn’t develop
  → Depend on documentation only (quality)
  → Lack of (quality) documentation (e.g., bad traceability) creates problems
- The system is optimized for performance, not for maintenance
  → Badly structured (e.g., bad adherence to principles such as divide & conquer, low coupling)
  → Too complex
- Inconsistent and outdated documentation
  → Inconsistent
  → Bad traceability
- Change impacts large and difficult to predict
  → Too complex
  → Lack of traceable documentation
- Defects being introduced when changing
  → Too complex
  → Lack of traceable documentation
- Change process does not exist or is not followed
  → Ad-hoc maintenance
  → Creating inconsistent documentation
## Traceability

### Motivation

- **Important Prerequisite: Make Requirements Identifiable**
  - Requirements should be identifiable via an unambiguous ID that will not change anymore
  - This ID serves for referencing purposes; in order to avoid misunderstandings, it should reflect the type of requirement and the document of origin

### Process

- **Priorization Methods**
  - Likert Scale Method
  - Cumulative Voting Method
  - Hierarchical Cumulative Voting
  - Analytic Hierarchy Process
  - Kano Model

### Challenge

### Examples

- RS-FR11 = functional requirement 11 of the requirements specification
- SD-FR23 = functional requirement 23 of the specification document
- PR17 = performance requirement 17
- RS-FR-Temp19 = functional requirement 19 of the temperature sub-system of the requirements specification
- **Traceability**
  - The ability to trace a requirements through its entire life-cycle, both forward and backwards.
Traceability Types

- **Horizontal Traceability**
  - Each description contains different views (e.g. structure, data flow, event flow)
  - These must be inter-related and consistent

- **Example**
  - A, B, C are examples of objects, methods, or attributes.

Priorization Methods
- Likert Scale Method
- Cumulative Voting Method
- Hierarchical Cumulative Voting
- Analytic Hierarchy Process
- Kano Model
Traceability Types

- **Vertical Traceability**
  - Different descriptions (problem description, user requirements, developer requirements and design) must be mutually consistent
  - Prerequisite for Consistency: every element of level (1, 2, 3) is traceably realized by elements at level i+1
  - User and developer requirements are distinguished less by their content than by their different user-oriented notations
  - Traceable realization means that the relationship between different abstractions is clearly and explicitly documented
Traceability

- **Traces = Relationships**
  - Traces are documented relationships among requirements and other artifacts
  - Relationships exist in all artifacts and they must be understood
  - Decision on what is a relationship must be explicit
    - Relationships are requirements too
    - "Automatic traceability" is only possible in completely formal settings

- **How**
  - Define a traceability model
    - What is a relationship?
    - Which entities have which types of relationships?
    - How to use the traces (e.g. for impact analysis)
  - Develop an infrastructure
    - Notation for relationships
    - Tool support
  - Follow a change process

Priorization Methods
- Likert Scale Method
- Cumulative Voting Method
- Hierarchical Cumulative Voting
- Analytic Hierarchy Process
- Kano Model
Traceability

- **Example Implementations**
  - Nametracing
  - Structural Tracing
  - Explicit Linking (Cross-references)
  - Trace matrix
  - Trace trees

- **Important Prerequisite: Unique Identifier for each Requirement**

- **Tool support can offer help, most so called “Requirements Tools” are Requirements Management Tools**
Prioritization Methods
- Likert Scale Method

- Evaluation of the importance of each requirement on the basis of a nominal scale
- Extremes: “very important“, “very unimportant“ and intermediate intervals

- Advantages
  - Easy and fast to carry out
  - Limited options

- Disadvantages
  - Results are worse than with other methods (too subjective)
  - Gives the user the impression that requirements are evaluated independently
  - Little differentiation is possible
## Priorization Methods - Likert Scale Method

### Example
- Each point on the scale corresponds to a numeric value, here 1 to 7
- All values are summed up and each value is divided by the sum
- The following priorities result:
  
  - Req1: 6/17
  - Req2: 7/17
  - Req3: 4/17

\[
\begin{align*}
  s &= \sum_{i=1}^{n} a_i \\
  b_i &= \frac{a_i}{s}
\end{align*}
\]

- Req2 \rightarrow Req1 \rightarrow Req3

### Table of Priorities

<table>
<thead>
<tr>
<th>Anf1: Das Tool soll kursive Schritt unterstützen</th>
<th>sehr wichtig</th>
<th>wichtig</th>
<th>eher wichtig</th>
<th>weder noch</th>
<th>eher unwichtig</th>
<th>wichtig</th>
<th>sehr unwichtig</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com/x" alt="X" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Anf2: Das Tool soll fette Schritt unterstützen</th>
<th>sehr wichtig</th>
<th>wichtig</th>
<th>eher wichtig</th>
<th>weder noch</th>
<th>eher unwichtig</th>
<th>wichtig</th>
<th>sehr unwichtig</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com/x" alt="X" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Anf3: Das Tool soll farbige Schritt unterstützen</th>
<th>sehr wichtig</th>
<th>wichtig</th>
<th>eher wichtig</th>
<th>weder noch</th>
<th>eher unwichtig</th>
<th>wichtig</th>
<th>sehr unwichtig</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com/x" alt="X" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Priorization Methods
- Cumulative Voting Method

- **Motivation**
- **Process**
- **Challenge**
- **Traceability**

**Priorization Methods**
- Likert Scale Method
- Cumulative Voting Method
- Hierarchical Cumulative Voting
- Analytic Hierarchy Process
- Kano Model

**Advantages**
- Easy and fast to carry out
- More differentiated evaluation than with the Likert scale method
- The value “0” can be assigned

**Disadvantages**
- Limited comprehensibility with high number of requirements and flat requirements hierarchy

## 100 points are distributed over the requirements

### Advantages

- Easy and fast to carry out
- More differentiated evaluation than with the Likert scale method
- The value “0” can be assigned

### Disadvantages

- Limited comprehensibility with high number of requirements and flat requirements hierarchy

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anf1: Das Tool soll kursive Schrift unterstützen</td>
<td>30</td>
</tr>
<tr>
<td>Anf2: Das Tool soll fette Schrift unterstützen</td>
<td>40</td>
</tr>
<tr>
<td>Anf3: Das Tool soll farbige Schrift unterstützen</td>
<td>0</td>
</tr>
<tr>
<td>Summe</td>
<td>70</td>
</tr>
<tr>
<td>Rest</td>
<td>30</td>
</tr>
</tbody>
</table>
Priorization Methods
- Hierarchical Cumulative Voting (HCV)

- Cumulative voting across several requirement levels (for example abstraction levels)

- **Advantages**
  - Improves comprehensibility
  - Abstraction levels are considered
  - Good results

- **Disadvantages**
  - Ideally, lower levels should completely describe the higher levels

### Priorization Methods
- Likert Scale Method
- Cumulative Voting Method
- Hierarchical Cumulative Voting
- Analytic Hierarchy Process
- Kano Model

### Table: Priorization Methods

<table>
<thead>
<tr>
<th>Punkte</th>
<th>Punkte</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schriftart</strong></td>
<td>Anf1: Das Tool soll kursive Schritt unterstützen 30</td>
</tr>
<tr>
<td>70</td>
<td>Anf2: Das Tool soll fette Schritt unterstützen 40</td>
</tr>
<tr>
<td><strong>Farben</strong></td>
<td>Anf3: Das Tool soll farbige Schritt unterstützen 0</td>
</tr>
<tr>
<td>30</td>
<td>Summe 100</td>
</tr>
<tr>
<td><strong>Rest</strong></td>
<td>Summe 70</td>
</tr>
<tr>
<td>0</td>
<td>Rest 30</td>
</tr>
</tbody>
</table>
AE – Requirements Management

Prioritization Methods
- Analytic Hierarchy Process (AHP)

- **Motivation**
- **Process**
- **Challenge**
- **Traceability**

### Prioritization Methods

- **Likert Scale Method**
- **Cumulative Voting Method**
- **Hierarchical Cumulative Voting**
- **Analytic Hierarchy Process**
- **Kano Model**

### Compare all requirements to each other in pairs

- **Extremes:** “Requirement 1 is much more important than Requirement 2” and vice versa

### Advantages

- Very differentiated evaluation possible
- Good results

### Disadvantages

- Complex calculation
- High number of comparisons already with a small number of requirements (300 comparisons with 25 requirements in a flat hierarchy)

<table>
<thead>
<tr>
<th>extrem viel wichtiger</th>
<th>sehr viel wichtiger</th>
<th>viel wichtig</th>
<th>etwas wichtig</th>
<th>gleich wichtig</th>
<th>etwas wichtig</th>
<th>viel wichtig</th>
<th>sehr viel wichtiger</th>
<th>extrem viel wichtiger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Das Tool soll kursive Schritt unterstützen</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Das Tool soll fette Schritt unterstützen</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Das Tool soll farbige Schritt unterstützen</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Prof. Dr. Dr. h. c. Dieter Rombach, Fundamentals of Software Engineering, Winter Term 2011/12
Prioritization Methods - Prioritization according to the Kano Model

- Differentiation of requirements into three categories
  - Satisfiers / one-dimensional requirements (e.g., camera quality)
  - Delighters / attractive requirements (e.g., advanced phone functions)
  - Dissatisfiers / must-be requirements (e.g., regular phone functions)
- Categories have different influence on customer satisfaction
- Advantages
  - Additional categorization is obtained
- Disadvantages
  - Elaborate questionnaire construction
  - Use of Likert scale
Priorization Methods - Prioritization according to the Kano Model

- **First step:** Fill out a questionnaire for the classification of the requirements
  - For each case, a pair of questions:
    -> Functional question
    -> Dysfunctional question
  - Five possible answers
    -> Formulation may vary but must express the same

---

**Likert Scale Method**

1. [ ] I like it that way.
2. [ ] It must be that way.
3. [x] I am neutral.
4. [ ] I can live with it that way
5. [ ] I dislike it that way

**Cumulative Voting Method**

1. [ ] I like it that way.
2. [ ] It must be that way.
3. [ ] I am neutral.
4. [ ] I can live with it that way
5. [x] I dislike it that way

---

**Hierarchical Cumulative Voting**

1. [ ] I like it that way.
2. [ ] It must be that way.
3. [ ] I am neutral.
4. [ ] I can live with it that way
5. [x] I dislike it that way

---

**Analytic Hierarchy Process**

1. [ ] I like it that way.
2. [ ] It must be that way.
3. [ ] I am neutral.
4. [ ] I can live with it that way
5. [x] I dislike it that way

---

**Kano Model**

1. [ ] I like it that way.
2. [ ] It must be that way.
3. [ ] I am neutral.
4. [ ] I can live with it that way
5. [x] I dislike it that way
Prioritization Methods
- Prioritization according to the Kano Model

- **Second step: Classification by means of an evaluation table**
  - Three additional categories
    - Questionable: Contradiction in the answers of the customer
    - Indifferent: Customer is neither satisfied nor dissatisfied when a feature / requirement is present or missing
    - Reverse: Customer’s answer contradicts the intention of the questionnaire creator

<table>
<thead>
<tr>
<th>Customer Requirements</th>
<th>Dysfunctional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. like</td>
</tr>
<tr>
<td>1. like</td>
<td>Q</td>
</tr>
<tr>
<td>2. must-be</td>
<td>R</td>
</tr>
<tr>
<td>3. neutral</td>
<td>R</td>
</tr>
<tr>
<td>4. live with</td>
<td>R</td>
</tr>
<tr>
<td>5. dislike</td>
<td>R</td>
</tr>
</tbody>
</table>

**Customer requirement is:**
- **A:** Attractive
- **M:** Must-be
- **R:** Reverse
- **O:** One-dimensional
- **Q:** Questionable result
- **I:** Indifferent
**Prioritization Methods - Prioritization according to the Kano Model**

- **Third step: Sorting based on the frequency of the classification**
  - M → O → A
  - With same classification, no differentiation is possible
  - Simultaneous collection of importance values using Likert scale method is possible for order formation

<table>
<thead>
<tr>
<th>Customer requirement number</th>
<th>Most frequent response</th>
<th>Second most frequent response</th>
<th>Third most frequent response</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>M</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>O</td>
<td>M</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>O</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>O</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>O</td>
<td>M</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>O</td>
<td>I</td>
</tr>
</tbody>
</table>
Requirements management is a part of requirements engineering
- Prioritize Requirements
- Manage changes
- Manage relationships

Tool support
- Helps to manage and analyze documents
- A tool does not replace thinking and processes!