Requirements Management
Overview of Main Activities

Management

- Elicitation
- Documentation
- Validation & Negotiation

© Fraunhofer IESE
AGENDA

- Assigning Attributes to Requirements
- Views on Requirements
- Prioritizing Requirements
- Traceability of Requirements
- Versioning of Requirements
- Management of Requirements Changes
- Tool Support
Requirements Attributes (1/2)

- Describe information about a requirement in a structured manner
  - Information of the same type can always be found in the same position
  - It is harder to overlook important information

- Frequently used attribute types
  - Identifier (ID)
  - Name
  - Description
  - Version
  - Author
  - Source
  - Stability
  - Criticality
  - Priority
Requirements Attributes (2/2)

- Additional attribute types
  - Requirement type
  - Person responsible
  - Status
    - Regarding the content
    - Regarding the validation
    - Regarding the agreement
  - Effort
  - Release
  - Legal obligation
  - Cross references
  - General information
Example: Attributes of a Requirement

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>F-04</td>
</tr>
<tr>
<td>Name</td>
<td>Change travel data</td>
</tr>
<tr>
<td>Description</td>
<td>The system should provide the Travel Management with the ability to change travel data.</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Author</td>
<td>R. the Requirements Engineer</td>
</tr>
<tr>
<td>Source</td>
<td>T. Travelmanager</td>
</tr>
<tr>
<td>Stability</td>
<td>fixed</td>
</tr>
<tr>
<td>Priority</td>
<td>must-have</td>
</tr>
<tr>
<td>Person Responsible</td>
<td>D. Developer</td>
</tr>
<tr>
<td>Status Content</td>
<td>concept</td>
</tr>
<tr>
<td>Status Validation</td>
<td>in validation</td>
</tr>
<tr>
<td>X-Ref</td>
<td>F-01; F-06; F-08</td>
</tr>
</tbody>
</table>
**Attribute Scheme**

- The set of all defined attributes for a class of requirements
  - E.g., a use case has attributes such as “Name” and “Pre-condition”
- Each class of requirement can have a tailored set of attribute types, depending on:
  - specific properties of the project
  - constraints of the organization
  - properties and regulations of the application domain
  - constraints and restrictions of the development process
- For each requirement, an attribute value is provided
  - E.g., an attribute value for “priority” could be “high”
Attribute Structures

- Table structure (template)
  - Simplest way
  - Manual

- Information models (model-based)
  - Usually when employing tools for requirements management
  - Allows for determining relations between attribute types of different attribute schemes (relational database)
  - Requirement dependencies help maintaining consistency
  - Templates can be generated
REQUIREMENTS MANAGEMENT

Views on Requirements
Selective and Condensed Views

- Views keep the complexity of the requirements manageable
- Views require the use of information models

**Selective views**
- Select particular requirements and/or mask certain attributes
- Create a view depending on role or sub-activities of that role, e.g.:
  - views for architects, programmers, project managers, testers
  - managing only those requirements that one is responsible for
  - selecting or viewing requirements by an attribute value

**Condensed views**
- Aggregate or generate data to obtain statistics, e.g.:
  - calculating the percentage of high-priority requirements
  - generating graphs by summing up attribute values
REQUIREMENTS MANAGEMENT
Prioritizing Requirements
Method for Requirements Prioritization

- Start by defining a goal / purpose of prioritization
- Prioritization can be determined by one or more attributes
- Typical prioritization criteria
  - Cost of implementation
  - Risk
  - Damage due to unsuccessful implementation
  - Volatility
  - Importance
  - Duration of implementation
- Involve different stakeholders in the prioritization process
- Select which requirements must be prioritized
- Selected requirements must stem from the same level of abstraction
Requirements Prioritization Techniques

- Selecting multiple suitable prioritization techniques is possible
- Two main categories
  - Ad hoc techniques
  - Analytical techniques (more elaborate)
- Trade-off: effort and costs vs. objective and comprehensive results
Ad-Hoc Prioritization Techniques

- Ranking technique
- Top-ten technique
- Single-criterion classification
  - E.g., mandatory, optional, nice-to-have
- Kano classification
- Likert scale
- Cumulative voting
- Hierarchical Cumulative Voting (HCV)
Analytical Prioritization Techniques

- Analytical hierarchy process (AHP)
- Cost-value-analysis
- Quality function deployment (QFD)
- Wiegers’ prioritization matrix
Likert Scale Method (1)

- Evaluation of the importance of each requirement on the basis of a bipolar 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
- Gives the user the impression that requirements are evaluated independently
- Little differentiation is possible

<table>
<thead>
<tr>
<th>Anforderung</th>
<th>sehr wichtig</th>
<th>wichtig</th>
<th>eher wichtig</th>
<th>weder noch</th>
<th>eher unwichtig</th>
<th>unwichtig</th>
<th>sehr unwichtig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anforderung 1: Das Tool soll kursive Schrift unterstützen.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anforderung 2: Das Tool soll fette Schrift unterstützen.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anforderung 3: Das Tool soll farbige Schrift unterstützen.</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Likert Scale Method (2)

- 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
  - \( \rightarrow \text{Req2} > \text{Req1} > \text{Req3} \)

\[
S = \sum_{i=1}^{n} a_i
\]

\[
b_i = \frac{a_i}{S}
\]

<table>
<thead>
<tr>
<th>Anforderung</th>
<th>sehr wichtig</th>
<th>wichtig</th>
<th>eher wichtig</th>
<th>weder noch</th>
<th>eher unwichtig</th>
<th>unwichtig</th>
<th>sehr unwichtig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anf1: Das Tool soll kursive Schrift unterstützen.</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anf2: Das Tool soll fette Schrift unterstützen.</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anf3: Das Tool soll farbige Schrift unterstützen.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Cumulative Voting Method

- 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

\[
S = \sum_{i=1}^{n} a_i
\]

\[
b_i = \frac{a_i}{S}
\]

<table>
<thead>
<tr>
<th>Anf1: Das Tool soll kursive Schrift unterstützen.</th>
<th>Punkte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anf2: Das Tool soll fette Schrift unterstützen.</td>
<td></td>
</tr>
<tr>
<td>Anf3: Das Tool soll farbige Schrift unterstützen.</td>
<td></td>
</tr>
</tbody>
</table>

Summe 70
Rest 30
Hierarchical Cumulative Voting (1)

- 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

<table>
<thead>
<tr>
<th>Schriftart</th>
<th>Punkte</th>
<th>Anforderung</th>
<th>Punkte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>70</td>
<td>Anf1: Das Tool soll kursive Schrift unterstützen.</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Anf2: Das Tool soll fette Schrift unterstützen.</td>
<td>40</td>
</tr>
<tr>
<td>Farben</td>
<td>30</td>
<td>Anf3: Das Tool soll farbige Schrift unterstützen.</td>
<td>0</td>
</tr>
<tr>
<td>Summe</td>
<td>100</td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>Rest</td>
<td>0</td>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>
Hierarchical Cumulative Voting (2)

Hierarchical Cumulative Voting (HCV) - Formulas

\[ LLR \]: Low Level Requirement  
\[ HLR \]: High Level Requirement  
\[ R_k \]: Requirement  
\[ H(R_k) \]: Parent requirement of \( R_k \)  
\[ p_a \]: Assigned priority  
\[ p_i \]: Intermediate priority  
\[ p_f \]: Final priority

(1): Direct calculation  
(2): Compensated calculation  
(3): Calculation of intermediate priority in the hierarchy

(1): \[ p_{i,LLR_u} = p_{a,LLR_u} \cdot p_{a,HLR_v} \]  
(2): \[ p_{i,LLR_u} = c_{HLR_v} \cdot p_{a,LLR_u} \cdot p_{a,HLR_v} \]  
(3): \[ p_{i,R_k} = p_{a,R_k} \cdot p_{i,H(R_k)} \]  
\[ p_{f,LLR_u} = \frac{p_{i,LLR_u}}{\sum_k p_{i,LLR_k}} \]
Hierarchical Cumulative Voting (HCV)

- Direct calculation (1) in case of
  - Complete set of requirements
  - Requirements of the lower level represent the complete decomposition of the requirements of the higher hierarchical level

- Otherwise compensated calculation (2)
  - Compensation factor \( c \)
    - Prevents requirements in small groups from receiving values that are too high compared to requirements in large groups

\[ (1): p_{i,LLR_u} = p_{a,LLR_u} \cdot p_{a,HLR_v} \]
\[ (2): p_{i,LLR_u} = c_{HLR_v} \cdot p_{a,LLR_u} \cdot p_{a,HLR_v} \]

\[ p_{i,R_k} = p_{a,R_k} \cdot p_{i,H(R_k)} \]

\[ p_{f,LLR_u} = \frac{p_{i,LLR_u}}{\sum_k p_{i,LLR_k}} \]
Hierarchical Cumulative Voting (4)

<table>
<thead>
<tr>
<th>KATEGORIE/SUBKATEGORIE</th>
<th>Punkte</th>
<th>Zwischenpriorität</th>
<th>Normalisierte Priorität Subkategorie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kat1/ Subkat1</td>
<td>70/40</td>
<td>70*40 = 2800</td>
<td>2800/10000 = 0.28</td>
</tr>
<tr>
<td>Kat1/ Subkat2</td>
<td>70/30</td>
<td>4200</td>
<td>0.42</td>
</tr>
<tr>
<td>Kat2/ Subkat2</td>
<td>30/30</td>
<td>900</td>
<td>0.09</td>
</tr>
<tr>
<td>Kat2/ Subkat3</td>
<td>30/20</td>
<td>600</td>
<td>0.08</td>
</tr>
<tr>
<td>Kat2/ Subkat4</td>
<td>30/30</td>
<td>1500</td>
<td>0.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBKATEGORIE/ANFORDERUNG</th>
<th>Punkte Anforderung</th>
<th>Zwischenprio. Kategorie/Subkategorie</th>
<th>Zwischenprio. Subkategorie/Anforderung</th>
<th>Normalisierte Prio. Anforderungen</th>
<th>Rang</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subkat1/ Anf1</td>
<td>100</td>
<td>2800</td>
<td>2800/10000 = 0.28</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Subkat2/ Anf2</td>
<td>20</td>
<td>5100</td>
<td>0.102</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Subkat2/ Anf3</td>
<td>80</td>
<td>5100</td>
<td>0.408</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Subkat3/ Anf4</td>
<td>30</td>
<td>600</td>
<td>0.018</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Subkat3/ Anf5</td>
<td>70</td>
<td>600</td>
<td>0.042</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Subkat4/ Anf2</td>
<td>100</td>
<td>1500</td>
<td>0.15</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Analytic Hierarchy Process (AHP)

- 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></th>
<th>extrem viel wichtiger</th>
<th>sehr viel wichtiger</th>
<th>viel wichtiger</th>
<th>etwas wichtig</th>
<th>gleich wichtig</th>
<th>etwas wichtiger</th>
<th>viel wichtiger</th>
<th>sehr viel wichtiger</th>
<th>extrem viel wichtiger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Das Tool soll kursive Schrift unterstützen.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Das Tool soll fette Schrift unterstützen.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Das Tool soll farbige Schrift unterstützen.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: The Kano Model

Customer delighted

Expectations exceeded

Expectations not fulfilled

Customer disappointed

Delighters

Satisfiers

Dissatisfiers
Example: Wiegers’ Prioritization Matrix (1/2)

- The priority of a requirement is calculated based on four properties
  - The **benefit** when implemented
  - The **detriment** when not implemented
  - The **cost** to implement
  - The **risk** to implement
**Example: Wiegers’ Prioritization Matrix (2/2)**

$$p = \frac{\%_{\text{Value}}}{\%_{\text{Cost}} \cdot \text{Weight}_{\text{Cost}} + \%_{\text{Risk}} \cdot \text{Weight}_{\text{Risk}}}$$

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Benefit</th>
<th>Detriment</th>
<th>Total Value</th>
<th>% Value</th>
<th>Cost</th>
<th>% Cost</th>
<th>Risk</th>
<th>% Risk</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>R01</td>
<td>5</td>
<td>3</td>
<td>13</td>
<td>8.4</td>
<td>2</td>
<td>4.8</td>
<td>1</td>
<td>3.0</td>
<td>1.345</td>
</tr>
<tr>
<td>R02</td>
<td>9</td>
<td>7</td>
<td>25</td>
<td>16.2</td>
<td>5</td>
<td>11.9</td>
<td>3</td>
<td>9.1</td>
<td>0.987</td>
</tr>
<tr>
<td>R03</td>
<td>5</td>
<td>5</td>
<td>15</td>
<td>9.7</td>
<td>3</td>
<td>7.1</td>
<td>2</td>
<td>6.1</td>
<td>0.957</td>
</tr>
<tr>
<td>R04</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>3.2</td>
<td>1</td>
<td>2.4</td>
<td>1</td>
<td>3.0</td>
<td>0.833</td>
</tr>
<tr>
<td>R05</td>
<td>4</td>
<td>9</td>
<td>17</td>
<td>11.0</td>
<td>4</td>
<td>9.5</td>
<td>4</td>
<td>12.1</td>
<td>0.708</td>
</tr>
<tr>
<td>R06</td>
<td>4</td>
<td>3</td>
<td>11</td>
<td>7.1</td>
<td>3</td>
<td>7.1</td>
<td>2</td>
<td>6.1</td>
<td>0.702</td>
</tr>
<tr>
<td>R07</td>
<td>6</td>
<td>2</td>
<td>14</td>
<td>9.1</td>
<td>4</td>
<td>9.5</td>
<td>3</td>
<td>9.1</td>
<td>0.646</td>
</tr>
<tr>
<td>R08</td>
<td>9</td>
<td>8</td>
<td>26</td>
<td>16.9</td>
<td>7</td>
<td>16.7</td>
<td>8</td>
<td>24.2</td>
<td>0.586</td>
</tr>
<tr>
<td>R09</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>6.5</td>
<td>4</td>
<td>9.5</td>
<td>2</td>
<td>6.1</td>
<td>0.517</td>
</tr>
<tr>
<td>R10</td>
<td>7</td>
<td>4</td>
<td>18</td>
<td>11.7</td>
<td>9</td>
<td>21.4</td>
<td>7</td>
<td>21.2</td>
<td>0.365</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td>54</td>
<td>46</td>
<td><strong>154</strong></td>
<td>100</td>
<td>42</td>
<td>100</td>
<td>33</td>
<td>100</td>
<td>--</td>
</tr>
</tbody>
</table>

[Source: Wiegers, 1999]
REQUIREMENTS MANAGEMENT

Traceability of Requirements
Definition of “Traceability”

The traceability of a requirement is the ability to trace the requirements over the course of the entire life cycle of the system.

- Establishing the relationships of a requirement with other requirements or other development artifacts
- Purpose-driven approach: choose the information to be recorded with respect to the purpose that it will serve
  - Focuses the recording of information
  - Helps structuring the information
Using Traceability

- **Verify** if a requirement has been implemented in the system
- Identify **gold-plated solutions in the system**
  - Does each (functional or qualitative) system property contribute to the implementation of a requirement?
- Identify **gold-plated solutions in the requirements**
  - Does each requirement contribute to a system goal?
- **Impact analysis** of the effects of changes
- Opportunities to **reuse** requirements in other projects
- **Accountability**: retroactive assignment of development efforts
- Simplifies **maintenance** because the relations have already been drawn
Classification of Traceability Relations

- **Vertical traceability**
  - Artifacts from before or after the requirements specification (RS)
  - Pre-RS traceability: links between requirements and artifacts that are the basis for the requirements (origin)
  - Post-RS traceability: links between requirements and artifacts of subsequent development activities (realization)
    - E.g.: component, implementation, test cases

- **Horizontal traceability**
  - Traceability between requirements
    - Mapping dependencies between requirements
    - Refinement, exclusion, alternative, ...
Representations of Requirements Traceability (1/2)

- Individual traceability
  - **Text-based references** (static)
    - Implicit: a business process refers to an activity as “A5”. A following chapter provides an explanation of A5.
    - Explicit; “This business process is refined through activities A3, A4, A5, and A6.”
  - **Hyperlinks** provide the additional possibility to jump to the appropriate section by clicking

- Disadvantage
  - No general overview of all relations
Representations of Requirements Traceability (2/2)

- General traceability
  - **Trace matrices** (tables)
    - Initial artifact IDs in rows, target artifact IDs in columns
    - The table cells visualize the relations, either by marking that a relation exists (“X”), or indicating the type of traceability link
  - **Trace graphs**
    - Requirements are nodes in a graph
    - Edges represent relationships (showing the type of relation)
    - Helps understand transitive relations
- Disadvantage
  - Become difficult to maintain as the number of requirements increase
Example: BTB Trace Matrix

<table>
<thead>
<tr>
<th>derived</th>
<th>F-01</th>
<th>F-02</th>
<th>F-03</th>
<th>F-04</th>
<th>F-05</th>
<th>F-06</th>
<th>F-07</th>
<th>F-08</th>
<th>F-09</th>
<th>F-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-01</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-02</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-03</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-04</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-06</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-07</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>F-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>
Example: Trace Graph

- Information About the System Context
- Requirements
- Components

- arrow realized through
- arrow is origin
- arrow refines
REQUIREMENTS MANAGEMENT

Versioning of Requirements
Versioning Management

- Important distinction
  - Versions of single requirements
  - Versions of requirements documents (i.e., requirement configurations)
    - Exceptional versions: requirements baselines

Note: the version of single requirements is difficult to separate from the version of the requirements document when using a word processor
Versioning of Single Requirements

- Every time a requirement is altered, its version number changes
  - E.g., “0.1, ... 1.0, ... 2.3”
- Whole version numbers (e.g., “1.0, 2.0, 3.0”) indicate validated, inspected and accepted statuses of a requirement
- Increments (e.g., “0.5, 1.1”) indicate unvalidated statuses
- The version number does not need to be continuous
  - E.g., it is possible to jump directly from 0.1 to 1.0
Versioning of Requirements Documents (Requirements Configuration)

- Versioning the requirements document is done in the same way single requirements are versioned.
- The requirements document with a particular version number is called a requirements configuration.
- Typically, the versions of single requirements within the same requirements document differ.
  - E.g., requirement 1 has version 0.5 and requirement 2 has version 1.3.
- New versions of the requirements document do not have to contain the latest versions of the single requirements.
Properties of a Requirements Configuration

- **Logical connections** between the requirements in the configuration
- **Consistency** among the requirements in the configuration
- **Unique identification** of the configuration (ID number)
- **Immutable** (stable) state of the specification and its requirements
- **Basis for rollbacks** if changes of requirements must be undone
Example: BTB Requirements Configurations

Product Dimension (Here: Requirements)

Requirements Configuration 1

Requirements Configuration 2 = Baseline 1

Requirement Req-4 in Version 1.1

Version Dimension

v0.1 v0.2 v0.3 ... v1.0 v1.1 v1.2

[Conradi & Westfechtel, 1998]
Requirements Baselines

- Special configurations of (typically stable) requirements, which:
  - represent the basis for release planning and development
    - marked for the contractor
  - can be used to estimate the effort needed to realize a system release
  - can be used to compare the planned system to competing systems
- Release development is based on the baseline
  - New requirements configurations are usually ignored until a new baseline is defined
REQUIREMENTS MANAGEMENT
Management of Requirements Changes
Requirements Changes (1/2)

- Over the course of a project, requirements *always* change
  - New requirements are added
  - Existing requirements are altered
  - Existing requirements are removed
- Many possible causes, e.g.:
  - Requirements were wrong, misunderstood or incomplete
  - Desires of the stakeholders have changed
  - New insights about functions, qualities or restrictions
  - Changes in law, technologies, market trends, business processes
- Changes can affect single requirements or entire requirements documents
Requirements Changes (2/2)

- Changes are usually good and mostly unavoidable
- The problem of requirements changes is not the changes themselves, but improperly dealing with them
- Too few changes indicates low stakeholder interest
- Too many changes over a short period of time:
  - indicates inadequately performed requirements engineering activities (e.g., elicitation and negotiation techniques)
  - makes it nearly impossible to develop a system that all stakeholders agree to
  - takes up a lot of resources
- **Faulty change management can easily ruin a good requirements development**
Change Management

- Describes a systematic dealing with requirements changes
- Core properties
  - Change request
  - Change Control Board (CCB)
  - Change process
Change Request

- A documented request
- Describes the desired change from the view of the requester

1. What should be changed?
   - Title (summary)
   - Description

2. Why?
   - Justification

3. How important is this change?

4. Who requests the change?

5. General data (identifier, date filed, which system release)
Activities of a Change Control Board

- Receive change requests
- Perform an impact analysis (consequences, effort, costs)
- Review the change request accordingly
- Accept or reject the change request
- Identify the necessary change measures (corrective, adaptive, hotfix)
- Define requirement changes or new requirements
- Prioritize and implement/plans the change
- Control and validate the applied changes
Composition of a Change Control Board

- Change manager
  - Responsible for changes, conflict mediation, negotiation, communicating and documenting decisions
- Contractor
- Architect
- Developer
- Configuration manager
- Product manager
- Project manager
- Quality assurance representative
- Requirements engineer
- Representative of the clients/users
Change Process

1. Requesting a change
2. Assessment of changes
   - E.g., reviewing traceability data
3. Acceptance and planning of changes
4. Realization of changes and validation

Core rules
- *Nobody* changes requirements without approval for that change
- This also goes for the project leader and requirements engineer
- Change processes only apply to validated configurations/baselines