Transformation approaches between user requirements and analysis models (report on paper)

Viliam Šimko

26.10.2010, D3S Seminar

Department of Distributed and Dependable Systems, Charles University, Prague
My motivation

- State of the art in the area of automated requirements processing
- The Eurocopter case
- GAČR: P103/11/1489 (Metody pro tvorbu a ověřování komponentových systémů ze specifikací v přirozeném jazyce)
- Also the new SW project – “Use Case Editor for Eclipse”

Yue T., Briand L. C., Labiche Y.: **A systematic review of transformation approaches between user requirements and analysis models**, August 2010, Journal: Requirements Engineering (impact factor: 0.931)
Requirements and MDE

- Model-Driven Engineering (MDE) works with formal models
- Requirements are in textual form
- Traceability
  - Traceability is important to understand connections between artefacts (IEEE Standard for Software Requirement Specification 1998)
  - Traceability links can be maintained through transformations
Systematic Review

- Guidelines for performing systematic literature review in software engineering*
  - Find recent approaches
    - Type of requirements (input):
      - Focused on requirements in natural language
      - Formal specification as an input was excluded
    - Type of analysis model (output):
      - UML elements
      - MSC (message sequence charts)
      - ERM (entity relationship model)
  - Derive common concepts and terminology
  - Compare the approaches
  - Define open issues and limitations

Research questions of the paper

- What are the different approaches used for transforming requirements into analysis models?
  - Requirement representations
  - Difficulty of documentation for users
  - Tool support
  - Analysis models that can be generated (structural / behavioural)
  - Intermediate models used during transformation
  - Efficiency of the transformation (how many steps)
  - Level of automation (automatic, semi-automatic, manual)
  - Traceability management support
  - Performed evaluation
- What are the current limitations of these approaches?
- What are the open issues to be further investigated?
**Search strategy**

- **Search in electronic databases**: IEEE Xplore, ACM Digital Library, Compendex, Inspec, SpringerLink
- **Manual search in peer-reviewed journals 1996-2008**:
  - IEEE Transactions on Software Engineering
  - Automated Software Engineering
  - Requirements Engineering Journal
  - Journal of Natural Language Engineering
  - ACM Transactions on Software Engineering and Methodology
  - Journal of Systems and Software, Software and Systems Modeling
  - Information and Software Technology, and Data & Knowledge Engineering
- **Manual search in conference proceedings 1996-2008**:
  - ACM/IEEE International Conference on Software Engineering
  - IEEE International Conference on Software Maintenance
  - IEEE/ACM International Conference on Automated Software Engineering
  - IEEE International Conference on Model Driven Engineering Languages and Systems and the former UML workshops
  - IEEE International Requirements Engineering Conference
- **Manual search in important software engineering textbooks**
After applying inclusion/exclusion criteria:
- **20 relevant studies (16 approaches)**
  - 12/16 derive **structural** elements
  - 9/16 derive **behavioural** features
  - 5/16 generate **complete analysis model**

+ journals
+ transitive closure on references

### Stats

<table>
<thead>
<tr>
<th>Electronic databases</th>
<th>Query results</th>
<th>After removing duplicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE Xplore</td>
<td>179</td>
<td>179</td>
</tr>
<tr>
<td>ACM Digital Library</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>Compendex</td>
<td>86</td>
<td>66</td>
</tr>
<tr>
<td>Inspec</td>
<td>83</td>
<td>34</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>86</td>
<td>67</td>
</tr>
<tr>
<td>Total</td>
<td>451</td>
<td>361</td>
</tr>
</tbody>
</table>

+ journals
+ transitive closure on references
Conceptual framework

- Static model describing common concepts
- Taxonomies:
  - How are the requirements specified (input)
  - Restriction rules imposed on requirements
  - Type of analysis model (output)
  - Pre-processing approaches
  - Transformation approaches
Generic transformation process 
(just for illustration)
Static model
Taxonomy of requirements

Hierarchy of terms
key/value pairs
class diagram

Taxonomy of restriction rules
Taxonomy of restriction rules

e.g. active vs passive voice, singular vs plural ...

Compound Sentences e.g. “only if-then structure is allowed”

Choice of words e.g. “only be or become can represent generalization”
## Restriction Rules

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>The subject of a sentence in basic and alternative flows should be the system or an actor.</td>
<td>Enforce describing flows of events correctly. These rules conform to our use case template (the five interactions).</td>
</tr>
<tr>
<td>R2</td>
<td>Describe the flow of events sequentially.</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>Actor-to-actor interactions are not allowed.</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>Describe one action per sentence. (Avoid compound predicates.)</td>
<td>Otherwise it is hard to decide the sequence of multiple actions in a sentence.</td>
</tr>
<tr>
<td>R5</td>
<td>Use present tense only.</td>
<td>Enforce describing what the system does, rather than what it will do or what it has done.</td>
</tr>
<tr>
<td>R6</td>
<td>Use active voice rather than passive voice.</td>
<td>Enforce explicitly showing the subject and/or object(s) of a sentence.</td>
</tr>
<tr>
<td>R7</td>
<td>Clearly describe the interaction between the system and actors without omitting its sender and receiver.</td>
<td></td>
</tr>
<tr>
<td>R8</td>
<td>Use declarative sentences only. “Is the system idle?” is a non-declarative sentence.</td>
<td>Commonly required for writing UCSs.</td>
</tr>
<tr>
<td>R9</td>
<td>Use words in a consistent way.</td>
<td>Keep one term to describe one thing.</td>
</tr>
<tr>
<td>R10</td>
<td>Don’t use modal verbs (e.g., might)</td>
<td>Modal verbs and adverbs usually indicate uncertainty; therefore metrics should be used if possible.</td>
</tr>
<tr>
<td>R11</td>
<td>Avoid adverbs (e.g., very).</td>
<td>Reduce ambiguity and facilitate automated NL parsing.</td>
</tr>
<tr>
<td>R12</td>
<td>Use simple sentences only. A simple sentence must contain only one subject and one predicate.</td>
<td></td>
</tr>
<tr>
<td>R13</td>
<td>Don’t use negative adverb and adjective (e.g., hardly, never), but it is allowed to use not or no.</td>
<td></td>
</tr>
<tr>
<td>R14</td>
<td>Don’t use pronouns (e.g. he, this).</td>
<td></td>
</tr>
<tr>
<td>R15</td>
<td>Don’t use participle phrases as adverbial modifier. For example, the italic-font part of the sentence “ATM is idle, displaying a Welcome message”, is a participle phrase.</td>
<td></td>
</tr>
<tr>
<td>R16</td>
<td>Use “the system” to refer to the system under design consistently.</td>
<td>Keep one term to describe the system; therefore reduce ambiguity.</td>
</tr>
</tbody>
</table>

### Use of control structures

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R17</td>
<td>INCLUDE USE CASE</td>
</tr>
<tr>
<td>R18</td>
<td>EXTENDED BY USE CASE</td>
</tr>
<tr>
<td>R19</td>
<td>RFS</td>
</tr>
<tr>
<td>R20</td>
<td>IF-THEN-ELSE-ELSEIF</td>
</tr>
<tr>
<td>R21</td>
<td>MEANWHILE</td>
</tr>
</tbody>
</table>


+ Technical Report with the same title TR SCE-09-05
DFG = Data Flow Graphs (data dependencies)
MSC = Message Sequence Charts (~ sequence diagram in UML)
ERM = Entity Relationship Model
Taxonomy of pre-processing approaches

- Tokenization
- Sentence splitting
- POS tagging
- Morphological analysis

Sequences of tokens → Syntactic parse tree

Enriches the parse tree using domain specific information

To eliminate ambiguities and inconsistencies before a new element is added

Pre-processed requirements transformed to analysis model or intermediate model
Taxonomy of transformation approaches

Set of predefined transformation rules

- Transformation Rule
- Transformation
- Rule based
- Ontology based
- Identity Transformation
- Pattern based

An intermediate model (shared vocabulary) is filled during NLP

Same information, different format

Source patterns → target patterns

- List of transformation rules inside the paper
<table>
<thead>
<tr>
<th>Requirements configuration</th>
<th>Analysis model</th>
<th>Pre-processing (Step 1)</th>
<th>Steps (Fig. 7)</th>
<th>Transformation</th>
<th>Automation</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (None, None, No)</td>
<td>Object diagrams</td>
<td>LA, SynP, SemP, PA</td>
<td>(1, 2, 4)</td>
<td>(R, R)</td>
<td>Automated</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Classes, attributes, and associations</td>
<td>LA, SynP, SemP</td>
<td>(1, 2, 4)</td>
<td>(O, R)</td>
<td>Automated</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Domain models, hybrid activity diagrams</td>
<td>LA, SynP</td>
<td>(1, 2, 3, 4)</td>
<td>(R, R, R)</td>
<td>Automatable</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Class, activity, state machine diagrams</td>
<td>LA, SynP, SemP</td>
<td>(1, 2, 4)</td>
<td>(R, R)</td>
<td>Semi-automated</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Architecture concepts (e.g., components and connectors)</td>
<td>Catg</td>
<td>(1, 2, 4, 5)</td>
<td>(R, R)</td>
<td>Manual</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Class diagrams</td>
<td>LA, SynP</td>
<td>(1, 6)</td>
<td>(R)</td>
<td>Semi-automated</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Class diagram, coarse-grained behavioral concept</td>
<td>Catg</td>
<td>(1, 2, 4, 5)</td>
<td>(R, P)</td>
<td>Manual</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Data types, variables, operations, control constructs (e.g., if-then-else and for loop)</td>
<td>None</td>
<td>(6)</td>
<td>(None)</td>
<td>Manual</td>
<td>N/A</td>
</tr>
<tr>
<td>2 (Glossary, Definition, None, No)</td>
<td>ERM, DFG, UML models (not described)</td>
<td>LA, SynP, SemP</td>
<td>(1, 2, 3, 4)</td>
<td>(I, R, R)</td>
<td>Automated</td>
<td>Low</td>
</tr>
<tr>
<td>3 (None, OBFS, Yes)</td>
<td>Class diagrams</td>
<td>LA, SynP, Catg</td>
<td>(1, 6)</td>
<td>(R)</td>
<td>Automatable</td>
<td>Low</td>
</tr>
<tr>
<td>4 (None, Use cases, No)</td>
<td>Sequence diagrams</td>
<td>Unknown</td>
<td>(?, ?)</td>
<td>(?)</td>
<td>Automated</td>
<td>Unknown</td>
</tr>
<tr>
<td></td>
<td>Message sequence charts</td>
<td>Catg, LA, SynP</td>
<td>(1, 6)</td>
<td>(R)</td>
<td>Automated</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Class and sequence diagrams</td>
<td>None</td>
<td>(5, 6)</td>
<td>N/A</td>
<td>Manual</td>
<td>N/A</td>
</tr>
<tr>
<td>5 (None, Use cases, Yes)</td>
<td>Extended sequence and extended activity diagrams</td>
<td>None</td>
<td>(6)</td>
<td>(P)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>6 (Glossary, Use cases, Yes)</td>
<td>Class diagrams</td>
<td>LA</td>
<td>(1, 2, 3, 4)</td>
<td>(R, R, R)</td>
<td>Automated</td>
<td>Low</td>
</tr>
<tr>
<td>7 (Domain model, Use cases, Yes)</td>
<td>State machines</td>
<td>None</td>
<td>(6)</td>
<td>(R)</td>
<td>Automated</td>
<td>High</td>
</tr>
</tbody>
</table>
Conclusion

- Practical automated solution still not available
- Observations:
  - Too much effort is required from users to document requirements (e.g. DSI provided manually)
  - Too many transformation steps
  - Incomplete analysis models generated
  - Most approaches did not address traceability
Suggestions from the review

- Use fully automated pre-processing techniques
- Use reasonably restricted natural language
  - Rationale should be provided for each restriction rule
  - Templates are good
- When available, use glossary (domain-specific information)
  - Derive initial version automatically
  - Provide support tools for DSI specification
- Derive complete, correct and consistent model (structure + behaviour)
- Use an intermediate model
- Maintain traceability links throughout transformations
- Also address scalability, usability, extensibility, interoperability of used transformations
Thank you