Requirements Exchange: From Specification Documents to Models

Morayo ADEDJOUMA, Hubert DUBOIS, François TERRIER
Ansgar RADERMACHER
Agenda

Big picture
Challenge
Technologies
Experiment Approach
Future findings
Related works / Conclusion
Big picture
• Requirements Engineering can be divided into discrete chronological steps:
  • Requirements elicitation,
  • Requirements analysis and negotiation,
  • Requirements specification,
  • System modeling,
  • Requirements validation,
  • Requirements management.

• For elicitation, analysis, negotiation & specification activities, requirements are in text format*

  ![Diagram of system architecture](image)

• For system modeling activity, a system architecture is defined in graphical environment*

* General practices in automotive industry
• To allocate requirements on system architecture, requirements engineers **manually** define the requirements model from system requirements.

• Problem: Each time the system requirements are changed, the requirements model must be updated.
Challenge
• **Goal**
  - Document the customer needs from source specifications in the required environment as well as requirements model

![SysML Requirements model]

- System Requirements document
- SysML Requirements model

• **Preconditions**
  - System requirements in MS documents (Excel™, Word™)
  - Requirements model in SysML (UML2 Profile)
  - Requirements in model are identical to requirements in requirements document
Use Case & Scenario

- **Use Case:**
  - Export requirements from documents
  - Import requirements from modeling tool (with Papyrus MDT tool)

- **MS documents allow to export (import) data in XML format**
- **UML2 modeler allow to import (export) data in XML format**
  - Scenario: transferring XML documents compliant to a standardized format

ReqIF: Requirement Interchange Format
Technologies
Graphical editor under the Eclipse Model Development Tool (MDT) Features:

- **UML2.2**
  - Papyrus targets to implement 100% of the OMG specification

- **Domain Specific Language (DSL)**
  - Papyrus provides advanced support for UML profiles enabling support for pure DSL: MARTE, EAST-ADL2, CCM...

- **SysML**
  - Provides complete support to SysML (static profile and graphical editors)

- **Text**
  - Possible to edit model elements using contextual text editors (syntax highlight, completion, content assist...)

- **Code Generation**
  - In Java, C/C++, Ada...

- **Customization**
  - Every part of Papyrus may be customized: model explorer, diagram, property and text editors...

- **And more...**
  - Glue around valuable UML/SysML diagram editors & others MDE tools
We are mainly interested in:

- XML Editor in Papyrus
  - Manage XML node and data following a metamodel (XSD, Ecore ...)

- Model transformation and Editor
  - Model 2 model (M2M) transformation using ATL

- Text editor
  - XML data in text format
  - Model in text format (XMI)
• We choose the Requirement Interchange Format (ReqIF) to link the 2 steps
• ReqIF Properties:
  ▪ Open standardized (Object Management Group)
  ▪ Generic, non-proprietary and tool-independent format
  ▪ Based on XML standard
  ▪ Developed to exchange requirements between authoring requirement management tools
  ▪ Driven by car industry

• ReqIF features:
  ▪ Uniquely identify requirements
  ▪ Associate attributes with the requirements, e.g. a requirements text, an attribute value,
  ▪ Establish relations between requirements…
  ▪ Structure specifications hierarchically

• ReqIF structure (main XML tag):
  ▪ ReqIFHeader (comment, creationTime, identifier… Title)
  ▪ ReqIFContent (DataTypeDefinition, SpecType, SpecObject, SpecRelation…)
We use Excel™ and Word™

- Microsoft provides an XML tools add-in to build XML-enabled applications
  - Import /Export data in XML (.xml)
  - Manage XML schema (.XSD) from a document
  - Transformation and style (.XSL, .CSS)
Experiment approach
• Mapping Table ReqIF – MS document – UML

<table>
<thead>
<tr>
<th>ReqIF Term</th>
<th>MS Excel™</th>
<th>MS Word™</th>
<th>UML element</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpecObject</td>
<td>Row</td>
<td>Table Row*</td>
<td>Requirement</td>
</tr>
<tr>
<td>Attribute</td>
<td>Cell</td>
<td>Cell</td>
<td>Attribute</td>
</tr>
<tr>
<td>Datatype</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>SpecRelation</td>
<td>Specific Row</td>
<td>Specific Row</td>
<td>Trace links</td>
</tr>
</tbody>
</table>

• **SpecObject**: In MS Word™, a requirement is defined following a table template*

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Description</th>
<th>ID references</th>
</tr>
</thead>
<tbody>
<tr>
<td>The requirement ID</td>
<td>The text requirement</td>
<td>All requirement IDs related to the requirement</td>
</tr>
</tbody>
</table>

• **Attribute**: Attributes element are id and text in the SysML context

• **Datatype**: In MS document, all data are string. Then, there are no specific datatype defined.

• **SpecRelation**: relation between 2 SpecObject:
  - In MS document
    - source : The current id requirement
    - target : one of ID reference
  - In UML
    - Source : the current id requirement
    - Target: the content of a traceability link

\[\text{Requirement (from SysML::Requirements)}\]
\[\text{id: String [1..1] = R1}\]
\[\text{/derived: Requirement [0..*] = [Requirement3]}\]
\[\text{/derivedFrom: Requirement [0..*] = [Requirement2]}\]
Steps to export data (requirements) from MS documents to XML file with ReqIF format

1. Open a requirements specification document (Word™ or Excel™)
2. Import ReqIF metamodel (ReqIF XML Schema from OMG) in Excel™
   • Integrated plug-in/ feature of MS documents
3. Map ReqIF element from XML Schema to Cell by drag and drop
4. Export mapping data in ReqIF XML format
Steps to export data (requirements) from MS documents to XML file with ReqIF format

1. **Open a requirements specification document (Word™ or Excel™)**
   - Install Developer and Complements add-in to access to XML tools features

2. **Integrate the ReqIF metamodel in MS document**
   - Use the OMG ReqIF metamodel in XML schema format
   - Modify the OMG XML schema to have only element which interest us (SpecObject, SpecRelation, Header)

   ```xml
   <xsd:element name="REQ-IF" type="REQIF:REQ-IF"/>
   <xsd:complexType name="REQ-IF">
   <xsd:sequence>
     <xsd:element maxOccurs="1" minOccurs="1" name="THE-HEADER">
     ....
   </xsd:element>
     <xsd:element maxOccurs="1" minOccurs="1" name="CORE-CONTENT">
     ....
   </xsd:element>
     <xsd:element maxOccurs="1" minOccurs="0" name="TOOL-EXTENSIONS">
     ....
   </xsd:element>
   </xsd:sequence>
   </xsd:complexType>
   </xsd:attribute ref="xml:lang" use="optional"/>
   </xsd:complexType>
   ```

3. **Map the XML schema element on data in MS document by drag and drop**
   - Tags appear in the document

4. **Export the data mapped in XML file**
From MS documents to ReqIF

XML File
(ReqIF Format)

From Word

From Excel
Steps to import data (requirements) from XML file with ReqIF format to UML modeler:

1. Import the XML file in ReqIF format in a new SysML project.
2. Import the ReqIF metamodel (ReqIF CMOF file from OMG) and convert it to Ecore metamodel (EMF).
   - Integrated plug-in/feature of MDT Papyrus.
3. Launch an ATL transformation.
   - Transform data in XML files as UML elements.
4. Generate a requirement diagram.
   - Integrated feature of MDT Papyrus.
1. Import the XML file in ReqIF format in a new SysML project
2. Import the ReqIF metamodel (ReqIF CMOF schema from OMG) and convert it to Ecore metamodel (EMF) MDT Papyrus feature
3. Launch an ATL transformation Transform data in XML files as UML elements
4. Generate a requirement diagram MDT Papyrus feature
• From MS documents to ReqIF, mapping is proceeded tag by tag
  ▪ A generic macro compliant to ReqIF will be easier and faster
• Poor traçability link capture
  ▪ In SpecRelation, attributes are source and target ids requirements, no type (satisfy, derive…)
• Difficult to manage hierarchical structure of requirements
  ▪ IDs reference as a list in MS documents: which is upper or derived?
• Difficult to manage classification of requirements
  ▪ The requirements type (functional, non-functional…) is not set in MS documents
• Datatypes
  ▪ All data are string
  ▪ No distinction between Enumeration, Integer or Boolean attribute value
Future findings
Next steps

- Manage others ReqIF term
  - Handle the requirements types in SpecType term (functional, non-functional…)
  - Improve import/export at Relation level (SpecRelation)
  - Deal with the datatypes (string, integer, enumeration…)

- Requirement have others attributes (risk, status…)
  - Update requirement specification documents (new columns)
  - Extend SysML profile to handle others attributes

- Export from MS documents by macro

- Implement Clause 8 and 9, 10, 11 of the ReqIF standard (OMG specification)

- Implement the roundtrip
  - Export requirements elements from models to XML file (ReqIF format) by the reverse model transformation
  - Generate new MS documents from XML file (ReqIF format) : more interesting than updating to keep the traceability in change management
Related works

- **ProR**, a Eclipse tool to define requirements in RIF* format
  - Definition of XML file in RIF format. Out of our scope because we want to avoid manual requirement definition

- Some Requirement Management tools (Reqtify, DOORS) propose a RIF import/export feature
  - Interoperability is established on tool-by-tool basis. Reqtify tool can only import and assess a RIF* file generated with Reqtify and it is the same case for DOORS tool. [Use cases evaluation in CESAR project]

- EAST-ADL2, UML2 profile proposes to import/export requirement using RIF* format
  - Same conclusion about interoperability that with Reqtify and DOORS tools

=> All implement a RIF version and not a ReqIF version (Actual state)
  - RIF is a first proposal by car manufacturers to exchange requirements between stakeholders. The standardized ReqIF is born of RIF.

- Some others entities work on an implementation of the RIF/ReqIF* specification.
  - Tool vendors (ArtiSAN studio, TopCased, MKS integrity…)
  - EU Projects (Verde, Deploy…)
  - Car industry
Conclusion

- Use modeling and text based approaches to specify requirements and integrate them in a standardized way
- Avoiding redundant working for requirement engineers
  ++ Automatic generation of requirement model
- Providing MDT Papyrus with requirement exchange
- Target to implement ReqIF standard
  -- No consideration of all ReqIF features
Thank you!

???

Morayo.Adedjouma@cea.fr