Model-Integration with SpecIF

2019-11-20

Oskar von Dungern, Dr.-Ing., adesso AG
Topics Today

1. Basic Assumptions
2. What is SpecIF
3. Model Integration
4. SpecIF Goals and Benefits
5. Resources and Examples
6. Status
Basic Assumptions

- There will be always specialized tools for different purposes
- It is unwise to require collaborators to use certain tools or even a single tool
- There is an interest
  - to search, navigate and audit partial results in a common context
  - to exchange model information between organizations and tools

→ That’s where SpecIF kicks in.
What is SpecIF?

• „Specification Integration Facility“, a GfSE initiative

• SpecIF adds **conventions to convey meaning**
  using known technical formats such as ReqIF or OSLC.

1. **Vocabulary** for Objects, Relations and Attributes
   „Requirement“, „Actor“, „State“, „Event“ ...
   „satisfies“, „reads“, „contains“, „triggers“, ...

2. Logic Assertions
   („First-order predicate logic“)
   „A Component satisfies a Requirement“
   „An Event triggers an Activity“

→ SpecIF carries both the „Visible“ and the „Meaning“
Purpose: Model Integration

- Search, navigate and audit partial engineering results in a common context
- Interrelate information elements of disparate sources to create a common view
- Find inconsistencies and gaps between different views
- Use-Cases:
  - Localize requirements on a BoM: Which components are affected when changing a requirement (or vice versa)?
  - When a use-case mentions a data-object, which system components and interfaces are involved?
  - Interrelate system structures with process models: Which activities are affected when modifying a system component (or vice versa)?
  - Associate a FMI simulation-routine with a SysML system component (block)
  - Collect and compare information about an element from different sources.
Add partial models step-by-step ...

1. Sichten und Modell trennen.
2. Modellelemente abstrahieren,
3. ... konsolidieren und
4. ... logisch vernetzen.

BPMN Processes

UML/SysML/FMC System models

Requirements
User Stories

Processes
System integration
Business objects
Annotations

http://specif.de
Five Principles of Model Integration

1. Separate View and Model
2. Abstract Model Element Types
3. Use a Vocabulary
4. Share Model Elements between Views
5. Interrelate Model Elements to build a semantic net
Find a Useful Abstraction Level

- **Model-Integration**: Cannot match original model-element types, because there are too many.

- **Configuration Management**: Should only handle a few artefact types.

→ Map to 3 fundamental model-element types ■ **Actor**, ● **State** and ♦ **Event** being common to *all* notations.

(see Fundamental Modelling Concepts by S.Wendt)
Use a Vocabulary

• Add meaning to terms
• Agree on terms and meaning
• Meaning is conveyed with the terms
• Can be translated to
  • national languages
  • special terminology in a given field

→ Use and contribute to the SpecIF System Engineering Vocabulary

Resource Class Names
• FMC:Actor
• IREB:Requirement

Statement Class Names
• IREB:satisfiedBy
• oslc_rm:refinedBy

Property Names
• dcterms:title
• SpecIF:FitCriteria

Property Values
• SpecIF:priorityHigh
## Use a Vocabulary

### Example: Requirements in a spread-sheet

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Benefits for the author</td>
<td>- Volle Prozesskontrolle und Nachvollziehbarkeit: kein Datenaustausch, kein e-mail &lt;br&gt;- Kein Informationsverlust durch unterschiedliche Datenmodelle verschiedener Tools &lt;br&gt;- Daten werden nicht verteilt, „disclosure“ wird erschwert &lt;br&gt;- Standardisierung durch HIS Austauschprozess und ReqIF Datenformat &lt;br&gt;- Einfachere Konsolidierung von Lieferanten-Feedback</td>
<td>30_submitted</td>
<td>1_high</td>
<td><a href="mailto:oskar.dungern@adesso.de">oskar.dungern@adesso.de</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benefits for the reviewer</td>
<td>- Kein Datenaustausch: kein Aufwand und keine Probleme &lt;br&gt;- Kosten: Keine Lizenzen und keine Schulung für ein Anforderungs-Management-System, um an einer Lastenheft-Abstimmung teilnehmen zu können &lt;br&gt;- Nachvollziehbarkeit durch zentrale Datenhaltung: kein e-mail</td>
<td>30_submitted</td>
<td>1_high</td>
<td><a href="mailto:oskar.dungern@adesso.de">oskar.dungern@adesso.de</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dialog layout and flow are process-like</td>
<td>The layout of the dialogs as well as the flow of dialogs follow the principal use cases. They are well organized and easy to understand.</td>
<td>60_completed</td>
<td>1_high</td>
<td><a href="mailto:oskar.dungern@adesso.de">oskar.dungern@adesso.de</a></td>
<td>70% of new users (knowing the nature of his task) are able to use the application without studying a manual or having a training.</td>
<td>nonsense</td>
<td></td>
</tr>
</tbody>
</table>
Simple Model-Integration „Adopt“

- Adopt existing model-element types, if they are equivalent
- Add new diagrams (views)
- Adopt existing model-elements having the same title
- Add new statements
- Build new glossary of model-elements, sorted by fundamental type
Example: Search, navigate and audit in a common context
What’s Needed to Integrate System-Models with Requirements

- Import FMC (ARCWAY Cockpit) available
- Import SysML → O. Alt (EA), O. Eichmann (Cameo)
- Import BPMN-XML available
- Import ReqIF
- Import XLSX available
- Model-Integration per „Adopt“ available
- SpecIF Server → O. Alt (.Net Core), O. v. Dungern (Node.js)
- Export ReqIF available
- Export ePub available
- Export OOXML (MS Word) available (early version)
Who's behind?

- **GfSE Arbeitskreis PLM4MBSE**
- **BMBF Forschungsantrag UrbanMovers**
- prostep ivip SysML Workflow Forum

**Development of Prototypes and Reference Solutions**

**Plan**
- SpecIF Program Management
- Agile Produkt-Entwicklung

**GitHub**
- Open Source Repository

**SpecIF Collaboration Platform**

**SpecIF**
- SpecIF Editor
- SpecIF Server
- SpecIF Viewer
- SpecIF Integrity
- SpecIF Database
- SpecIF Word
- SpecIF Confluence

**Model-Integration with SpecIF**
- TOGAF
- SysML
- BPMN
- ReqIF

**Development Team**
- Development
- Development

**ReqIF**
- SpecIF

**FMC**
- SpecIF

**BPMN**
- SpecIF

**SpecIF**
- FMC
- SpecIF
- SpecIF
- SpecIF
- SpecIF
Who's behind?

- Schaeffler
- Daimler
- VPE
- BMBF Forschungsantrag UrbanMovers
- ZF
- GfSE Arbeitskreis PLM4MBSE
- prostep ivip SysML Workflow Forum
- TOGAF → SpecIF
- BPMN → SpecIF
- FMC → SpecIF
- ReqIF ↔ SpecIF
- SpecIF Server
- SpecIF Editor
- SpecIF Database
- SpecIF Viewer
- SpecIF Confluence
- SpecIF Word
- SpecIF Integrity
- SpecIF → CIM Database
- SpecIF → Integrity

Plan

- SpecIF Program Management
- GitHub: Open Source Repository
- Agile Produkt-Entwicklung
- Development Team
- MDD4All.de
- Institute for Flight Cabin Systems
- Fraunhofer IPK
- adesso
- Object Management Group

Development of Prototypes and Reference Solutions
SpecIF Goals and Benefits

- Exchange model-based specifications between organizations and tools.
- Combine texts and models from different tools.
- Search, navigate, and audit in a common context.
- Manage the product lifecycle from birth to death ("end-to-end"):
  - Reference for all engineering-disciplines
  - Combining methods
  - Technology-neutral
  - Vendor-neutral
  - Schema-conforming
  - Standard-conforming
  - Open and cooperative
Resources

- SpecIF Home

- SpecIF Schema and Tools on GitHub (Open Source, Apache License)
  → Use the GitHub Ticket System for questions, proposals and requests

- Hosted SpecIF Schema and Consistency-check (CORS-enabled, watch for new releases)

- SpecIF Viewer

- Examples:
  - Fahrtbeginn (BPMN)
  - Dimmer (FMC)
  - SmAV (SysML)

- SpecIF Vocabulary
Literature


Details for further discussion
Purpose: SpecIF integrates information from different sources
Use Case: Exchange model information between tools

1. Change a state diagram in SysML
2. Check consistency of the specification
3. Validate the behavior in a simulation tool
4. Generate code for an embedded controller
5. Store in a Product Lifecycle Management System
The eye sees the same – SpecIF adds interrelated model data

Creating the „Visible“
- Text editing and image „drawing”
- Needs brain and discipline to build and keep it consistent

Partial Modelling
- Text editing and modelling per method
- Tool support within the methods

Model Integration
- Text editing and modelling per method
- Elements in all views are interrelated by a semantic net

BPMN
SysML
SpecIF carries both the Visible and the Meaning

Medium/Format
(The Presentation)

original, loss-less

Notation
(The Visible)

abstracted

Integrated Model
(The Meaning)

Objects and Relations
(The Persistence)

Interrelates all elements of all model diagrams and all notations
Status ...

see [github.com/GfSE](https://github.com/GfSE)
Syntax and Semantics

Semantics
(Meaning)

No vocabulary:
Import and export by the same vendor
(Extessy → Atego)

2004

Syntax
(Format)

RIF
Syntax and Semantics

Semantics
(Meaning)

- No vocabulary:
  - Import and export by the same vendor (Extessy → Atego)

Semantics
(Meaning)

- ReqIF Implementation Guide,
  - oslc_rm Vocabulary: property names
  - Store/exchange Requirements vendor-independently

Syntax
(Format)

- RIF
- ReqIF
- oslc_rm

distributed
Syntax and Semantics

Semantics

(Meaning)

No vocabulary:
Import and export by the same vendor (Extessy → Atego)

ReqIF Implementation Guide,
oslc_rm Vocabulary:
→ property names
Store/exchange Requirements vendor-independently

SpecIF Vocabulary:
→ resource names, statement names, property names property values
Store/exchange Requirements and Models vendor-independently

Syntax

(Format)

2004
2011
2016

RIF

ReqIF

oslc_rm

SpecIF

distributed

simplified polymorph

20.11.2019

28
Syntax and Semantics

Semantics
(Meaning)

- No vocabulary: Import and export by the same vendor (Extessy → Atego)
- ReqIF Implementation Guide, oslc_rm Vocabulary: property names
  Store/exchange Requirements vendor-independently
- SpecIF Vocabulary: resource names, statement names, property names property values
  Store/exchange Requirements and Models vendor-independently

Syntax
(Format)

- RIF
- ReqlF
- oslc_rm
  distributed
  simplified polymorph
- SpecIF
  Lossless Transformation possible

Applicable to all formats below
A SpecIF data set contains both the types („model“) and the instances („data“ = „payload“)

SpecIF model with Resource- and Statement-C classes

Hierarchical ordering of Resources (e.g. for convenient reading)

SpecIF data (payload) with Resources and Statements

The classes can be defined at runtime („dynamic model“)
The SpecIF Integration Model with 5 Fundamental Model-element Types

- **SpecIF resourceClass**
  - **SpecIF:Diagram**
  - *Model Diagram*

- **SpecIF statementClass**
  - **SpecIF:Diagram**
  - *Model Diagram*
Consolidate model elements from different diagrams

- Key to success is the abstraction using 5 fundamental model element types
- Impossible with 162 model element types in SysML and almost as many in BPMN
- Even within SysML the current tools fail to properly consolidate model elements from different model views
Abstraction: All model diagram types (notations) consist of three fundamental model-element types*

<table>
<thead>
<tr>
<th>Model-element Type</th>
<th>BPMN Business-process</th>
<th>State-machine</th>
<th>System-composition</th>
<th>Organisation Chart</th>
<th>UML Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>X</td>
<td>(X)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>State</td>
<td>(X)</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Event</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Prof. Dr. Siegfried Wendt, Founding Director of the Hasso-Plattner-Institute, Potsdam: Fundamental Modelling Concepts
A complete specification needs also 'feature' and 'requirement'.

<table>
<thead>
<tr>
<th>Model-element Type</th>
<th>View Type (Notation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>X</td>
</tr>
<tr>
<td>State</td>
<td>(X)</td>
</tr>
<tr>
<td>Event</td>
<td>X</td>
</tr>
<tr>
<td>Feature</td>
<td></td>
</tr>
<tr>
<td>Requirement</td>
<td></td>
</tr>
</tbody>
</table>
Taking a model diagram, automatically identify model elements and their relations to build a semantic net.

**Process Z [business process]**

- **Start Event !**
- **Process Step 1**
- **Process Step 2**
- **End Event !**

**Model-Integration with SpecIF**

- **User**
  - **Start Event !**
  - **Process Step 1**
  - **Process Step 2**
  - **End Event !**

5 model elements identified

5+4+1+1+1 relations identified in total

**triggers**

**precedes**

**shows**

**contains**

**signals**