Transforming Aspectual Requirements to Architecture using Natural Language Semantics: Experiences from AOSD-Europe

Awais Rashid, Ruzanna Chitchyan, Monica Pinto, Lidia Fuentes
Need for Separation of Concerns

Software is becoming increasingly large ... complex ... distributed

Development requires focusing on one concern at a time
Aspect-Oriented Software Development (AOSD)
AOSD-Europe

• A large-scale academia-industry collaboration funded by the European Commission FP6 Programme for
  – Large-scale AOSD research
  – Shared investment in development of tools and methodologies
  – Building bridges between AOSD research and practice

• 11 organisations investing approx. 1000 pm of effort
An Infrastructure for Large-Scale Collaboration

National
- MULDRE (UK)
- VERA (UK)
- Isis (UK)
- FeasiPLE (Germany)
- AspectLab (Belgium)
- SOFTAS (Portugal)

European
- AMPLE (FP6)
- DiVA (FP7)
- EA-Analyzer (MC)
- DISCS (MC)

AOSD-Europe
Modelling Research in AOSD-Europe

- MDE not MDA
- AOSD-Europe Atelier
  - A toolset to systematically refine and transform aspect-oriented requirements to implementation
  - A methodology to provide guidance on the use of associated methods
- Key focus in a number of spin-off projects
  - AMPLE, DiVA, MULDRE, VERA, SOFTAS
AOSD-Europe and AMPLE

**SPLE**
- Variations and commonalities of features & artifacts

**AOSD**
- Modularise features and their concerns

**MDD**
- Domain-specific abstractions (DSL) to configure generation and assembly of artefacts

---

**Industrial Case Studies**

- Align metamodels along the SW development chain
- Improve traceability techniques, repositories, and tooling
- AOSD > AOP
  - Start mining aspects from requirements
  - Try to evolve aspects separately through SW lifecycle
  - Model Weaving
- Iterative assessment against real use cases
Bridging the gap between unstructured and structured models
Focus

Mapping and Refinement of Requirements level Aspects to Architecture and down to the Detailed Design

Architecture

Requirements

Integration

UML Design Models
Challenges

- Natural Language requirements are unstructured
- Different metamodels at different development stages
  - Requirements, Architecture and Design
  - Elements at different abstraction levels
- Metamodel elements refer to the same concerns
  - How do these concepts relate?
- No one-to-one mapping between the elements
  - We need guidelines and heuristics
Concern **Security**

R₁: **Users must enrol with the system**

- **Subject**
- **Degree**
- **Relationship**
- **Object**
AOSD-Europe Requirements Description Language (RDL)

- Make natural language requirement semantics more explicit and base composition on these semantics.
- How?
  - Use grammatical information
  - Use semantic grouping from linguistics
  - Use natural language processing to help automation
RDL Concern Example

Concern Security

R₁: Users must enrol with the system

<Concern name="Security">
  <Requirement id="1">
    <Subject> Users </Subject>
    <Degree type="Modal" semantics="Modal" level="high"> must </Degree>
    <Object> system </Object>
    <Relationship type="Move" semantics="Group"> enrol with the </Relationship>
  </Requirement>
</Concern>
Semantic Queries

Query:
Select the requirements where user enrols or a system is enrolled to.

RDL Query
all requirements where (Subject="users" and Relationship="enrol")

Also used in Queries for matching:
Synonyms: register, sign up, join...
Lemmas: enrol, enrolled, enrolling..
RDL Composition Example

Composition:
Apply the requirements where user enrols to a system before any buyer bids.

<Composition name="SecureBidding">
  <Constraint operator="apply"> subject= "user" and relationship= "enrol" and object= "system" </Constraint>
  <Base operator="before"> subject= "buyer" and relationship= "bid" </Base>
  <Outcome operator="satisfied"/>
</Composition>
Composition Sequencing Operators
AO Requirements Specification

• RDL Representation

(a) RDL SOR Pattern

```xml
<Concern name="Security">
 <Requirement id="1">
   <Subject> User</Subject>
   <Degree type="modal" semantics="obligation" level="very high">have to
   <Degree>
   <Relationship type="Move" semantics="Group">log on
   </Relationship> to the
   <Object>auction system</Object> for each session.
   </Requirement>
 </Concern>
```

(b) RDL Composition

```xml
<Composition name="LoggingComposition">
 <Constraint operator="affiliate">
   concern="Security" </Constraint>
 <Base operator="before">all concerns</Base>
 <Outcome operator="ensure" />
</Composition>
```

Concern & Composition Refinement
- Use of “Use Cases”
Case 2: Log on to the System

Brief Description
This use case describes how an enrolled user logs on to the auction system.

Basic Flows

B1: Logon:
The use case begins when a buyer wants to place a new bid.

1. The user selects the Log On option.
2. The system provides the log on window.
3. The user inputs his/her email address and the system password.
4. The system successfully verifies that the email address and password are correct. The user is authenticated.
5. The system notifies the user that she/he is successfully logged in and is authorised to use her/his system account.
6. The use case terminates.

Alternative Flows

A1: Incorrect login details:
In step 4 the system verifies that the provided email address and password are registered with the system. The password/email are not registered. The system prompts the user to input a registered email/password:

1. If the user provides a correct email address and password, the use case resumes at step 5.
2. If the user provides an incorrect email address/password, the system prompts that the input is incorrect and the use case terminates.

Sub-flows

Pre-conditions
The user must be enrolled in order to be able to log in.

Post-conditions

Special Requirements
Multiple users may be logging in at the same time and should not interfere with one another.

This affects the composition specification of "Log on"

This identifies a conflict between "Security" and "Enrollment"
Enrol
1. All potential users of the system must first enrol with the system.
   1.1 To enrol the user must input his/her name, surname, email address, and postal address.

Security
1. Users have to log on to the system for each session.

2. The system emails the password for the system access to the newly enrolled users’ email address.

Guidelines for concern refinements

Guidelines for composition refinements

(a)

(b)
Requirements Refinement Guidelines

- Guidelines on Concern Refinement
  - **Guideline 1.** New Concern Formation
    - A new concern is identified
  - **Guideline 2.** Sub-Concern Formation
    - A concern is divided into sub-concerns or new concern identification
  - **Guideline 3.** Requirements Update
    - More information about requirements is obtained

- Guidelines on Composition Refinement
  - **Guideline 1:** Review due to new concern formation
  - **Guideline 2:** Review due to sub-concern formation
  - **Guideline 3:** Review due to requirements update
  - **Guideline 4:** Review due to conflict or inconsistency identification
  - **Guideline 5:** Review in case of ALL wildcard use
Mapping of SRO Patterns

• Pattern identified for each requirement in a concern
• Several identified mapping alternatives for the SRO elements to AO-ADL
### Mapping from Requirements to Architecture Metamodells

<table>
<thead>
<tr>
<th>RDL Meta-model Elements</th>
<th>Corresponding AO-ADL Meta-model Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Concern</strong> → Composite component, component, interface, or role</td>
<td><strong>Requirement</strong> → SRO patterns</td>
</tr>
<tr>
<td>Subject &amp; Object</td>
<td>Component, state attribute, interface, or role</td>
</tr>
<tr>
<td>Relationship</td>
<td>Operation of an interface (provided by the component representing the object in the SRO pattern and required by the component representing the subject in the SRO pattern)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Composition</strong> → <strong>Connector</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>See Base Operator and Base Query Expression mappings.</td>
</tr>
<tr>
<td>Base Operator</td>
<td>Binding type in AO-ADL</td>
</tr>
<tr>
<td>Base Query Expression</td>
<td>Port specification and Pointcut specification</td>
</tr>
<tr>
<td>Constraint</td>
<td>Binding section in aspectual-bindings. Also see Constraint Operator and Constraint Query Expression mappings.</td>
</tr>
<tr>
<td>Constraint Operator</td>
<td>No direct mapping, but these are mapped via semantic roles realised by sub-elements from the constraint and base pointcuts.</td>
</tr>
<tr>
<td>Constraint Query Expression</td>
<td>Aspect, Role, and Advice in AO-ADL</td>
</tr>
<tr>
<td>Outcome</td>
<td>Constraints (post-conditions), critical execution paths</td>
</tr>
</tbody>
</table>
Mapping of SRO Patterns - Example

(a) RDL SRO Pattern

```
<Concern name="Security">
  <Requirement id="1">
    <Subject>User</Subject>
    has to <Relationship type="Move" Semantics="Group">log on</Relationship> to the <Object>auction system</Object> for each session.
  </Requirement>
</Concern>
```

(b) ADL Mapping

```
<component name="User"> …
  <required-interface role="Log-on">
    <operation name="log on">…
    </operation>
  </required-interface>…
</component>

<component name="AuctionSystem"> …
  <provided-interface role="Log-on">
    <operation name="log on">…
    </operation>
  </provided-interface>…
</component>

<connector name="UserAuctionSystem"> …
  <provided-port name="UserPort">
    <component name="User"/>
  </provided-port>
  <required-port name="AuctionPort">
    <component name="Auction"/>
  </required-port>
  <component-binding name="…">
    <binding provided-role="UserPort" required-role="AuctionPort"/>
  </component-binding>
</connector>
```

The User and Auction System components will interact among them throughout a connector.
Mapping of Compositions

• RDL composition = Base + Constraints + Outcome

• **Base**: Concerns affected by crosscutting concerns
  – Mapped to Specification of Pointcuts in the Aspect Binding section in Connectors in AO-ADL

• **Constraints**: Crosscutting concerns affecting other concerns
  – Mapped to Aspectual Components specified inside the Aspect Binding section in Connectors in AO-ADL

• **Outcome**:
  – **Ensure**: Mapped to Critical Execution Paths governed by aspectual components in AO-ADL
  – **Satisfied**: Mapped to AO-ADL Constraints.
<Composition name="LoginComposition">
  <Constraint operator = “affiliate”> concern = “Security” </Constraint>
  <Base operator = “before”> all concerns where subject = “user” and 
  (relationship = “buy” or relationship = “sell”) </Base>
  <Outcome operator = “ensure”/>
</Composition>

1 <connector name="…”>
2  <required-port name="User”>
3   <port-specification>
4    component name="User” and
5    (interface-operation = “buy” or interface-operation = “sell”) 
6   </port-specification>
7  </required-port>
8  <required-port name="Security”>
9   <port-specification>
10    component name="Security” and interface-operation = “login”  
11   </port-specification>
12  </required-port>
13  <aspectBinding name = “LoginComposition“>
14   <pointcut-specification>
15    required-port-name="User”
16   </pointcut-specification>
17   <binding operator="before*“>
18     <aspectual-component name="Security” criticality="critical”  
19      role="log-on" advice-name = "login”/>  
20   </binding>
21  </aspectBinding>
22 </connector>
Mapping Guidelines

- **Guideline 1**: RDL concern mapping
  - The specific corresponding element is defined by the semantics of the concern
  - RDL allows specification of concern representing a viewpoint, feature, use case, etc.

- **Guideline 2**: RDL requirement mapping
  - Application of SRO pattern mappings

- **Guideline 3**: Mapping of Compositions
  - Very close link between RDL’s composition and AO-ADL’s aspect bindings in connectors

- **Guideline 4**: Review provided interfaces
  - Identification of crosscutting influences

- **Guideline 5**: Checking and refinement
  - Basic Refinement to be completed by next section
Conclusions

• Systematically supporting MDE across the lifecycle is non-trivial
• Natural language is not as ad-hoc and unstructured as it seems
• Utilising natural language semantics can help traceability of stakeholder intentions and goals through to implementation
• Scalable tool support can help automate the mappings
Further Details


For More Details About AOSD-Europe and its Activities

Contact me: awais@comp.lancs.ac.uk

Visit: http://www.aosd-europe.net