From Aspect-oriented Requirements Models to Aspect-oriented Business Process Design Models

Sébastien Mosser\textsuperscript{1}, Gunter Mussbacher\textsuperscript{2}, Mireille Blay–Fornarino\textsuperscript{1}, Daniel Amyot\textsuperscript{2}

\textsuperscript{1}CNRS – I3S Lab (UMR 6070), University of Nice, France \quad \textsuperscript{2}SITE, University of Ottawa, Canada

March 23, 2011
Motivation

- From individual aspect-oriented techniques to a **concern-driven** approach for software engineering
  - Continued encapsulation of requirements-level concerns in design-level artifacts (and eventually through all development phases)
  - Transformation process
  - Feedback loops support iterative development

- In the context of business process modeling / service-oriented architecture (SOA)
Motivation: PicWeb Example

• PicWeb
  • Allows a set of pictures with a given tag and up to a given threshold number to be retrieved from existing partner services
    • Flickr (Yahoo!), Picasa (Google)
  • Part of SOA-based information system called jSeduite
    • Supports information broadcasting for academic institution
    • Deployed in three institutions, used daily
  • Implementation follows SOA methodological guidelines

• Concerns
  • GetPictures, Truncate, Picasa, Caching, Payment, Randomizer
Background: ADORE (A meta–moDel suppOrting oRchestration Evolution)

• Specifies complete business processes as an orchestration of services
• Aspects are modeled with fragments which are integrated into processes through the usage of several composition algorithms
• Built on many-sorted first order logic
  • Consistency rules (no concurrent access, no dead path…)
• Shared join point handling
  • Default merge algorithm
  • Fragment weaving to order aspects
Transformation from AoURN to ADORE (1)

- Generate **ADORE** process skeleton automatically

**Activity**
- **act1** receive()
- **act2** flickr::getPicturesWithTag()
- **act3** truncate::truncate()
- **act4** reply()

**Orchestration**
- picweb::getPictures
- truncate::truncate

**Guard**
- condition := gt(nrPhotos, threshold)

**Feedback**
- Incomplete Req.
- Lessons Learned
- Conclusion & Future Work
Transformation from AoURN to ADORE (2)

- **Generate ADORE process skeleton automatically for aspect**

  - ![Diagram 1](image1.png)

- **Generate composition directive from matched join points as composition techniques of AoURN and ADORE differ greatly**

  - ![Diagram 2](image2.png)

### Motivation

- AoURN
- ADORE

### Transformation

- Feedback
- Incomplete Req.
- Lessons Learned
- Conclusion & Future Work
Transformation from AoURN to ADORE (3)

- Interactions between concerns and their resolution are modeled on the Concern Interaction Graph (CIG) in AoURN

- Caching is applied before Picasa which is applied before Payment

- Transformation to ADORE takes the CIG into account when creating the composition directives
- Lower precedence fragment applied to hook of other fragment instead of matched join point
Transformation from AoURN to ADORE (4)

- Increased consistency
- Requirements engineers and designer work with their own models
- Req. analyses ensure properties in design model
- ADORE-specific profile for AoURN (e.g., well-nested)
- Reuse concern composition and ordering in design model
- Interactions identified in design model which may not be apparent in the requirements model

<table>
<thead>
<tr>
<th>Mapping:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AoURN</td>
<td>ADORE</td>
</tr>
<tr>
<td>Start Point</td>
<td>receive (orchestration); predecessors (fragment)</td>
</tr>
<tr>
<td>End Point</td>
<td>reply (orchestration); successors (fragment)</td>
</tr>
<tr>
<td>Responsibility</td>
<td>assign (&lt;&lt;business process&gt;&gt;); invoke (&lt;&lt;service&gt;&gt;); boolean assign activity, guard relations</td>
</tr>
<tr>
<td>OR-fork</td>
<td>exclusive waitFor relations entering an activity</td>
</tr>
<tr>
<td>OR-join</td>
<td>non–exclusive waitFor relations entering an activity</td>
</tr>
<tr>
<td>AND-fork</td>
<td>not transformed (same comp.); invoke (different component);</td>
</tr>
<tr>
<td>AND-join</td>
<td>waitFor relation between two activities</td>
</tr>
<tr>
<td>Static Stub</td>
<td>hook activity</td>
</tr>
<tr>
<td>Sequence</td>
<td>module (&lt;&lt;business process&gt;&gt;)</td>
</tr>
<tr>
<td>Pointcut Stub</td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td></td>
</tr>
</tbody>
</table>
Data-Driven Feedback and Requirements

- Absence of data specifications in AoURN limits the analysis of the transformed design models
  - Inputs and outputs for actors are known → captured with metadata
  - Inputs and outputs for a <<service>> or a <<business processes>> may also be known to a certain extent → captured with metadata
  - Added as inputs and outputs to the ADORE activities by transformation

- Incomplete requirements (shown in more detail here)
- Design choice impacts requirements (see paper)
- Verification of user interface (see paper)
Incomplete Requirements

• There may be inconsistencies
  • Addressed by default refactoring techniques in ADORE
  • Ensures that the dataflow makes sense in the design model
    • Variables used but not defined $\rightarrow$ input to the orchestration/fragment
    • Variables not used after definition $\rightarrow$ output as part of the reply

• Designer investigates changes made by default refactoring

• This may lead to changes to the requirements model
Add tags describing input/output data

Default refactoring can be improved by the designer (may even lead to an update in the scenario model)
Lessons Learned

- Mapping of concerns relatively straightforward owing to the similarity of AoURN scenario models and ADORE
- AoURN goal models only mapped indirectly through AoURN scenario models
- More challenging to identify the semantic concepts in the notations for which transformation rules can be established
- AoURN is constrained to a subset by profile
  - Is the loss of expressiveness at the AoURN level a problem?
  - Has to be balanced carefully against ADORE’s ability to prove its composition mechanisms (more concepts, harder to prove)
- Lightweight data-modeling + “fill in” technique
  - Enables feedback-driven data-flow specification at a high level
- To unify or not to unify?