Exploring Extensibility of Architectural Design Decisions

Salvador Trujillo¹, Maider Azanza¹, Oscar Díaz¹, Rafael Capilla²

¹: University of the Basque Country (Spain)
²: Universidad Rey Juan Carlos (Spain)

SHARK/ADI 2007
Minneapolis, MN (USA)
May 20th, 2007
General Problem

- Design decisions are an important part of the architecting process that must be documented.

- But, can this architectural knowledge be reused?

- This work
  - explores extensibility ideas from software product lines to show how synthesis architectures can be extended on the basis of design decisions.
  - introduces design decision documentation in such synthesis architectures.
Outline

- Introduction
- Background
- Problem Statement
- Architecture Extensibility
  - Representing Synthesis Architectures
  - Extending Synthesis Architectures
  - Composing Synthesis Architectures
  - Documenting Design Decisions
- Conclusions
Software Product Lines

- **Precursors:** McIlroy 60s, Parnas 70s

- **Definition**
  
  "A software product line is a set of software-intensive systems, sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [Clements 2001]

- "A Software Product Line…
  
  - (1) Set of **products** “SPL is a set of software-intensive systems..”
  - (2) **Features** “..sharing a common, managed set of features..”
  - (3) **Domain** “..that satisfy the specific needs of a particular market segment or mission..”
  - (4) **Core Assets** “..are developed from a common set of core assets..”
  - (5) **Production Plan** “..in a prescribed way.”
FOP and MDD

- Feature Oriented Programming (FOP)
  - is a paradigm for software product lines
  - programs are synthesized by composing features
  - a feature is an increment in program functionality

- Model Driven Development (MDD)
  - is a paradigm for software creation
  - programs are the result of model transformations
  - a model represents a part of the reality (prog. design)

- FOP and MDD
  - are productivity paradigms
  - are mostly independent paradigms
FOMDD

- FOMDD = FOP • MDD
  - is a blend of FOP and MDD
  - products in a SPL are synthesized
    - by composing features to create models
    - transforming models into executables

- Synthesis in FOMDD
  - initially using scripting
    - repetitive, time-consuming and cumbersome
  - recently, new approach
    - scripts generated from abstractions

- FOMDD ideas further detailed at ICSE2007 paper:
  - Feature Oriented Model Driven Development: A Case Study for Portlets. S. Trujillo, D. Batory, O. Diaz.
Synthesis Architectures

- **Synthesis architecture**: Synthesis abstraction that represents the parts that form a synthesis metaprogram.

- Comprises:
  - software models
  - relationships between them
  - structural properties
Outline

- Introduction
- Background
- Problem Statement
- Architecture Extensibility
  - Representing Synthesis Architectures
  - Extending Synthesis Architectures
  - Composing Synthesis Architectures
  - Documenting Design Decisions
- Conclusions
Problem Statement

Synthesis architectures involve a set of design decisions.

Our aim:
- document the design decisions that happen along synthesis.
- use step-wise refinement to extend synthesis architectures and their design decisions.

Benefits:
- the relationship between design decisions and their impact on architecture is documented
- it is possible to extend the architecture with new decisions
Outline

- Introduction
- Background
- Problem Statement
- Architecture Extensibility
  - 1) Representing Synthesis Architectures
  - 2) Extending Synthesis Architectures
  - 3) Composing Synthesis Architectures
  - 4) Documenting Design Decisions
- Conclusions

S. Trujillo, M. Azanza, O. Díaz, R. Capilla
1) Representing Synthesis Architectures

**Synthesis Architecture example**

**GXL representation**

```xml
<gxl x:artifact="aPKSynthesisArchitecture">
  <graph id="PKBaseGraph"
    edgeids="true" edgemode="directed" hypergraph="false">
    <!- content omitted -->
    <node id="ViewObject">
      <!- content omitted -->
    </node>
    <node id="JspObject">
      <!- content omitted -->
    </node>
    <edge id="d_View2Jsp" to="JspObject" from="ViewObject">
      <!- content omitted -->
      <dd:decision id="j2eeJsp">
        <dd:issue>Generate JSP for a J2EE container</dd:issue>
        <dd:description>Extend transformation to ...</dd:description>
        <dd:status>Approved</dd:status>
        <!- remaining attributes omitted -->
      </dd:decision>
    </edge>
    <!- content omitted -->
  </graph>
</gxl>
```
1) Representing Synthesis Architectures

- Design decisions shape the synthesis architecture.

- GXL was selected to depict such architectures because:
  - is XML based
  - provides a language and a metamodel
  - has tool support

- Currently there are tools available to generate a script starting from this synthesis abstraction.
2) Extending Synthesis Architectures

- An architecture can be extended to obtain customized architecture using **refinements**.

- To attain this we need to define:
  - a set of design decisions, common to the base architecture.
  - extension points to make the architecture extensible for new requirements.
3) Composing Synthesis Architectures

**Base Architecture**

**Extended Architecture**

- **Refinement**
  - Original conception to deal with mobile
  - Remaining attributes omitted
  - Status: Approved
3) Composing Synthesis Architectures

- XAK is used to specify the architecture extensions
  - supports the refinement of XML documents for SPL extensibility.

- Each extension encompasses:
  - the impact of synthesis-specific design decisions
  - the rationale behind them

- Immediate traceability between extensions and design decisions
  - we can trace how each decision extends the architecture
4) Documenting Design Decisions

- Composition of architectural extensions is not enough.
- Document the design decisions in order to
  - detail traceability between requirements and products.
  - foster reuse by specifying the attributes of the decisions that can be useful in the future
- We introduce an XML-based language to describe the attributes of a decision

<table>
<thead>
<tr>
<th>Decision Attribute</th>
<th>Description</th>
<th>XML Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision ID</td>
<td>Identifier for the decision</td>
<td><code>&lt;cd:decision id=&quot;&quot;/&gt;</code></td>
</tr>
<tr>
<td>Decision Issue</td>
<td>The architectural issue being addressed</td>
<td><code>&lt;cd:issue&gt;</code></td>
</tr>
<tr>
<td>Decision Name</td>
<td>Name of the decision</td>
<td><code>&lt;cd:name&gt;</code></td>
</tr>
<tr>
<td>Decision Description</td>
<td>Description of the decision</td>
<td><code>&lt;cd:description&gt;</code></td>
</tr>
<tr>
<td>Status</td>
<td>Denotes the status of the decision</td>
<td>`&lt;cd:status value=&quot;pending</td>
</tr>
<tr>
<td>Constraints</td>
<td>Restrictions that apply to a decision</td>
<td><code>&lt;cd:constraint&gt;@list</code></td>
</tr>
<tr>
<td>Alternatives</td>
<td>List of alternative decisions considered during the decision making activity</td>
<td><code>&lt;cd:alternatives&gt;@list</code></td>
</tr>
<tr>
<td>Related Decisions</td>
<td>Show related decisions but complex dependencies between decisions can be defined here</td>
<td><code>&lt;cd:dependencies&gt;@list</code></td>
</tr>
</tbody>
</table>
4) Documenting Design Decisions

Documentation Example

Drawback: Documentation is included within synthesis abstraction

A XSL transformation can extract user-friendly HTML documentation
Outline

- Introduction
- Background
- Problem Statement
- Architecture Extensibility
  - 1) Representing Synthesis Architectures
  - 2) Extending Synthesis Architectures
  - 3) Composing Synthesis Architectures
  - 4) Documenting Design Decisions
- Conclusions
Conclusions

- Initial attempt towards reuse of synthesis architectural knowledge and design rationale.

- We promote traceability between architectural design decisions and architectural solutions.

- Future:
  - Explore the relationship between features and design decisions. (e.g. to handle dependencies between design decisions and their impact of change).
  - See if evolution of architectural knowledge could be handled by the described extensibility approach.
Thank you!

**Rafael Capilla**
rafael.capilla@urjc.es

Salvador Trujillo
struji@ehu.es

Maider Azanza
maider_azanza@ehu.es

Oscar Díaz
oscar.diaz@ehu.es

http://www.onekin.org