Towards Evolving Configuration Models

Thorsten Krebs, Lothar Hotz

Christoph Ranze, Guido Vehring
Introduction
- Application Context
- Basic Approach
- General View on Evolution
Consistency of a Model
Basic Operations for Modifications
Degrees of Modifications
Support for Evolution
Summary
Configuration of Industrial Product Families (ConIPF):
- Combining *Product Lines* and *Structure-oriented Configuration*.

Software-based products are rarely without relatives.

Products evolve to accommodate specialized demands by:
- Becoming *variants and versions* of the same product, or
- Becoming *separate but similar products*.

This set of products represents a *product family*.
Defining a methodology for deriving products starting with features / customer requirements.

- Use **feature and artifact models**, 
- Develop **intermediate knowledge representations**, and 
- Define a **configuration notation** with the expressiveness to handle the variability in industrial product lines.
Evolution in the Application domain

- Reasons for evolution:
  - New requirements
  - New technical abilities
  - Bug fixes

- Preventative evolution during domain engineering
- Adaptive evolution during application engineering
- Corrective evolution during maintenance

Evolution of the existing artifacts vs. evolution of the models
Configuration Knowledge Modeling Language (CKML):

- **Conceptual model:**
  - Describing objects by types and properties
  - Relations: Taxonomic (is-a), Compositional (has-parts), and Constraints

- **Procedural model**

- **Goal Specification**

- A priori known facts about the product-to-be
General View

Product Domain

Modeled Domain

Possible Solutions

Modeled Solutions

Innovative Solutions

Evolution/Innovative Configuration
Aspects of Innovative Configuration

- Changing the domain model
- Changing already (partly) configured configurations
- Changing the real artifacts

Evolution and the configuration process:
- Interchanging approach vs.
- Separate processes

Preventative evolution does not have impacts on the configuration process!
- „Only“ new real artifacts have to be produced

Change the domain model, don‘t change the configuration goal.
Aspects of Innovative Configuration

- Changing the domain model
- Changing already (partly) configured configurations
- Changing the real artifacts
- Evolution and the configuration process:
  - Interchanging approach vs.
  - Separate processes
- Preventative evolution does not have impacts on the configuration process!
  - „Only“ new real artifacts have to be produced

Change the domain model, don’t change the configuration goal.
General Questions

- How can the configuration model be changed?
- What are pre-requisites?
- What is allowed to be changed?
- What has to be true after the change?
- What is the relation between innovative configuration and knowledge acquisition?

Pre-requisite for reasoning: A consistent model!
Consistency of a Model

- Specialization conditions:
  - Subsets between each property of a sub- and super-concept

- Compositional conditions:
  - Common super-concept of the parts
  - Disjunctive parts
  - Correct number restrictions
  - One part can be related to several wholes (non-exclusiveness assumption on parts)

- Constraint conditions:
  - Constraint must be related to concept properties
  - Constraint may only compute subsets of value ranges
Basic Operations for Modifications

- Add, delete, modify (= delete plus add)
- Extending/reducing parameter value ranges
- Adding/deleting parameters
- Adding/deleting specializations
- Adding/deleting decompositions
- Adding/deleting constraints

- Changing procedural knowledge has no impact on the result!
<table>
<thead>
<tr>
<th>Operation</th>
<th>Impacts on old Configurations</th>
<th>Impacts on the Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extending parameter values</td>
<td>Check the subset relation and corresponding constraints</td>
<td>Perform inheritance, check concept types</td>
</tr>
<tr>
<td>Deleting parameter values</td>
<td>Performing inheritation and checking the corresponding constraints</td>
<td>Checking the specialization subtree and the corresponding constraints</td>
</tr>
<tr>
<td>Adding new parameter values</td>
<td>Adding new specializations</td>
<td>Deleting specializations</td>
</tr>
<tr>
<td>Deleting parameter values</td>
<td>Deleting parameters</td>
<td>Modify/delete the sub-concepts</td>
</tr>
</tbody>
</table>

**Modification Impacts**

- **Modify/delete the sub-concepts**

- **Check the subset relation and corresponding constraints**

- **Check the specialization subtree and the corresponding constraints**

- **Perform inheritance, check concept types**

- **Deleting specializations**

- **May be inconsistent**

- **Existing configurations are still consistent**

- **May miss the new parameter**

- **Instances may be of different type**

- **May be inconsistent**

- **Adding new specializations**

- **Deleting parameters**

- **Adding new parameter values**

- **Deleting parameter values**
### Modification Impacts

<table>
<thead>
<tr>
<th>Operation</th>
<th>Model impacts</th>
<th>Configuration impacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding new decompositions</td>
<td>Check the compositional conditions</td>
<td>May contain too less instances</td>
</tr>
<tr>
<td>Deleting decompositions</td>
<td>Check the compositional conditions</td>
<td>May contain too many instances</td>
</tr>
<tr>
<td>Adding constraints</td>
<td>Check the constraint conditions for the new constraint</td>
<td>May be inconsistent</td>
</tr>
<tr>
<td>Deleting constraints</td>
<td>No impacts</td>
<td>May be too narrow</td>
</tr>
</tbody>
</table>

*Often only mixed operations make sense.*
Degress of Modifications

- Easy to handle (e.g. adding parameters)
  - Modifications do not cause inconsistencies
  - Modifications do not entail further modifications

- Unproblematic
  - Impacts on several (sub)concepts (e.g. parameter changes)

- Problematic
  - Involving Constraints
  - Multiple Concepts
  - Support through dependency-analysis

- Critical
  - Complete restructuring of taxonomies and partonomies
  - E.g. Movement of subtrees
  - Configuration solutions are substantially changing
Degrees of Modifications

- Parameter affected
- Specialization affected
- Comp. Relation affected
- Constraint involved
- Re-Structuring
Support through the Model

- Declarative representations can be used for computing dependencies between knowledge entities
  - Impact analysis
  - Estimate costs for evolution

- Existing configurations can give hints for relevant modifications

- Support for the evolution of artifacts
  - Compute focused views on related concepts and relations
  - Automatic classification of new concepts

- Specific, previously known *evolution points* can be specified in the domain model
Evolution initiation:
- User starts the evolution process
- A conflict occurs which is not wanted to be solved by changing the goal specification

Change the model, possibly by being supported through the model

Add the changes in the current configuration process

Perform a new configuration with the old goal specification (reuse of user interactions)
Summary

- Configuration in a changing world
- Integrate domain modeling and configuration
- Integrate even, modeling, configuration, and manufacturing
- Types of modifications and their impacts
- Support of evolution tasks through declarative models
Thank You for Your Attention
Introduction

EU project with 4 partners:

- Industrial Partners:
  - Robert Bosch GmbH (project leader)
  - Thales Nederland B.V.

- University Partners:
  - Rijksuniversiteit Groningen
  - Universität Hamburg