The Common Variablility Language (CVL)

Franck Fleurey
SINTEF
franck.fleurey@sintef.no
About SINTEF

- Largest independent research organisation in Scandinavia

- About 2000 employees
  - 1200 in Trondheim
  - 800 in Oslo

- 7 Divisions
  - ICT about 280 researchers
Motivation

- For the common variability language
  - More and more modeling DSLs
  - Need to express variability
  - Make a generic language to express variability
    - Based on capturing the differences between variants
  - Make generic tools to integrate with DSL tools

- For presenting it here
  - Similarity with AOM techniques
  - The core concepts of CVL
    - Fragments <-> Aspects ?
    - Substitution <-> Weaving ?
Outline

- The Common Variability Language
  - Overview of CVL
  - Basic concepts: Substitutions
  - A simple example

- Building generic tools

- Demo

- Discussion
Big Picture

- Generic & Standardized
  - CVL
    - Variation model
      - Description of possible variations in the system
      - Resolution models
        - Selection of a set of options in the variation model
    - Execute CVL Transformations
      - Resolved domain models

- Focused on a domain
  - DSL
    - Base domain model
      - Domain model of a particular family of system
      - Family of systems fully described in the domain specific language. All regular DSL tools can be applied to these models
CVL Concepts

- Substitutions
  - Value substitution
  - Reference Substitution
  - Fragment substitution

- Variability
  - Choices
  - Similar to Feature models
  - Constraints

- Choices
Arithmetic Expressions
Meta-Model
ARI Base Model

\[ 3x^2 - 4x + 1 \]
ARI Variation (1)

Variations on the coefficients

$3x^2 - 4x + 1 \rightarrow 3x^2 - 4x + 1$

3 value substitutions

$3x^2 - 4x + 1$
ARI Resolution (1)

Providing the new values

\[ 3x^2 - 4x + 1 \rightarrow 5x^2 - 1x + 0 \]

3 value resolutions

Resolution Model: \(5x^2 - x\)

- Value Resolution a = 5
- Value Resolution b = 1
- Value Resolution c = 0
ARI Variation (2)

Variations on the variables

\[ 3x^2 - 4x + 1 \rightarrow 3x^2 - 4x + 1 \]

2 reference substitutions

Variation Model: \( 3?^2 - 4? + 1 \)
Reference Substitution: \( xa \rightarrow ? \)
Reference Substitution: \( xb \rightarrow ? \)

\[ V \]
\[ 2 \]
\[ V \]

\[ \text{var x} \text{ var y} \text{ var z} \]
ARI Resolution (2)

Providing replacement variables

\[ 3x^2 - 4x + 1 \iff 3y^2 - 4z + 1 \]

2 reference resolutions

\[ 3x^2 - 4x + 1 \]

\[ 3y^2 - 4z + 1 \]

\[ \text{Resolution Model } 3y^2 - 4z + 1 \]

\[ \text{Reference Resolution } x_a \rightarrow y \]

\[ \text{Reference Resolution } x_b \rightarrow z \]
ARI Variation (3) 1/2

Variation on an operator

\[ 3x^2 - 4x + 1 \Rightarrow 3x^2 - 4x + 1 \]

- 1 Fragment substitution
- 1 Placement Fragment
  - 2 From Placement BE
  - 1 To Placement BE
Replacement operator specification

- 1 Replacement Fragment
  - 2 From Placement BE
  - 1 To Placement BE
ARI Resolution (3) 1/2

Fragment binding

\[ 3x^2 - 4x + 1 \quad x + y \]
ARI Resolution (3) 2/2

Fragment substitution

$3x^2 - 4x + 1 \Rightarrow 3x^2 + 4x + 1$

- 1 Fragment resolution

$5 \wedge 1 \rightarrow x$
Outline

- The Common Variability Language
  - Overview of CVL
  - Basic concepts: Substitutions
  - A simple example

- **Building generic tools**
  - Demo

- Discussion
Variability Tool /Editor

- Specify Variation Models
  - Model Fragments
  - Substitutions
  - Constraints

- Specify Resolution Models
  - Choices
  - Fragment Bindings

- Trigger Transformations

- Generic Tools – DSL independent
  - Transformations
  - Editors
Relations with DSL / DSL Tools

- Reuse exiting DSL tools
  - Keep base-model and variability model separated
  - Only references from the variability model to the base model
  - Generic transformations

- Concrete syntax
  - Generic variability editor
  - Integrate with exiting DSL editors
    - Object editors
    - Text editors
    - Graphical editors
  - Low requirements on the DSL editors
  - Customizable visualization of variability
Integration with DSL Editor

Solution

- A required interface for selection and highlighting
- Many editors already include these functionalities
- Supported by any editor (tree, text, graphical)

Interaction with DSL editor

- Provide object selection
  - Make references to objects
  - Create fragments / boundary elements
- Provide object highlighting
  - Display referenced objects
  - Display fragments
  - Display substitutions
Eclipse Implementation

- Meta-model defined in EMF
  - EMF based variability editor
  - Integration with any EMF meta-model
- Editor interface for integration

```java
/**
 * Highlight in the editor the object identified by xml_id with the color corresponding to type.
 * The ID of object is provided by the static operation
 * ICVMLEnabledEditor.IDProvider.getObjectId(EObject obj). Only this operation
 * should be used to compute the ID of EObjects.
 * @param xml_id The ID of the object.
 * @param type The type of highlighting to use.
 */
public void highlightObject(String xml_id, int type);

/**
 * Remove highlighting for all object in the editor
 */
public void clearHighlighting();

/**
 * Get the editor selection.
 * @return The set of domain objects (instances of the DSL meta-model) selected in the editor.
 */
public ArrayList<Object> getSelectedObjects();

/**
 * Set the selection of the editor.
 * @param objects The IDs of the set of domain object which should be selected.
 */
public void selectObjects(ArrayList<String> objects);
```
Tool integration

MOF (ECore)

CVL
- CVL model
- CVL Generic Object Editor

DSL
- domain model

Editors
- DSL Editor Interface
- DSL Editor Interface
- Editors

Transformations
- Variability Editor
- Resolution Editor
- Transformations
- ... 

Code Generators
- Analyzer
- Transformations
- Code Generators
- ...
Tools integration

- 3 editors:
  - Generic EMF Tree editor
  - Train Control Language Editor (GMF)
  - Papyrus UML Editor (GEF)
Outline

- The Common Variability Language
  - Overview of CVL
  - Basic concepts: Substitutions
  - A simple example

- Building generic tools

- Demo

- Discussion
Outline

- The Common Variability Language
  - Overview of CVL
  - Basic concepts: Substitutions
  - A simple example

- Building generic tools

- Demo

- Discussion
Discussion

- Low level variability meta-model
  - Graph based
  - Fully generic
- Not user-friendly to instantiate by hand
  - Creating fragments / boundary elements
- But, the tedious tasks can be automated with the integration with DSL editors
  - Choosing objects in the base model
  - Creating the fragment
  - Computing boundary elements
- Very low requirements on the DSL tools and editor
  - A simple interface to implement
Thank you for your attention !