Modelling Language Evolution

Hans Vangheluwe (and Bart Meyers)

17 April 2012
Assume DSLs are Useful

Industrial Adoption?

Major Challenge: Language Evolution!
(because DSLs evolve more rapidly than GPLs)
and artifact co-evolution
<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 references</td>
</tr>
<tr>
<td>2 running example</td>
</tr>
<tr>
<td>3 de-constructing the co-evolution problem</td>
</tr>
<tr>
<td>4 it’s not so simple . . .</td>
</tr>
<tr>
<td>5 proposed workflow</td>
</tr>
</tbody>
</table>
Cicchetti, A., Di Ruscio, D., Eramo, R., Pierantonio, A.  
Automating co-evolution in model-driven engineering.  

Herrmannsdoerfer, M., Benz, S., Juergens, E., 2009.  
Cope - automating coupled evolution of metamodels and models.  

Bart Meyers and Hans Vangheluwe.  
A framework for evolution of modelling languages.  
Recap: Modelling Languages/Formalisms terminology
Modelling Language/Formalism Engineering
Meta-Model of TrainSim Language
Instance Model in TrainSim Language
Meta-Model of (simplified Place/Transition) Petri Net Language
Instance Model in Petri Net Language
Model of TrainSim to Petri Net Transformation
Transformation with Input and Output
A Typical Formalism Transformation Graph (FTG)
The Essential Pattern

\[ \mathbf{MM}_D \quad \text{T} \quad \mathbf{MM}_I \]

\[ m \quad \text{T} \quad T(m) \]
Delta operations with their migration and inverse operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Type</th>
<th>Migration operation</th>
<th>Inverse Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-breaking operations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalize meta-property</td>
<td>Additive</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Add non-optional meta-class</td>
<td>Additive</td>
<td>None</td>
<td>Eliminate meta-class</td>
</tr>
<tr>
<td>Add non-optional meta-property</td>
<td>Additive</td>
<td>None</td>
<td>Eliminate meta-class</td>
</tr>
<tr>
<td>Extract superclass</td>
<td>Additive</td>
<td>None</td>
<td>Flatten hierarchy</td>
</tr>
<tr>
<td><strong>Breaking and resolvable operations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eliminate meta-class</td>
<td>Subtractive</td>
<td>Eliminate instances</td>
<td>Add non-optional meta-class</td>
</tr>
<tr>
<td>Eliminate meta-property</td>
<td>Subtractive</td>
<td>Eliminate instances</td>
<td>Add non-optional meta-property</td>
</tr>
<tr>
<td>Push meta-property</td>
<td>Subtractive</td>
<td>Eliminate properties from superclass instances</td>
<td>Pull meta-property</td>
</tr>
<tr>
<td>Flatten hierarchy</td>
<td>Subtractive</td>
<td>Eliminate superclass instances</td>
<td>Extract superclass</td>
</tr>
<tr>
<td>Rename meta-class</td>
<td>Updative</td>
<td>Change instances</td>
<td>Rename meta-element</td>
</tr>
<tr>
<td>Rename meta-property</td>
<td>Updative</td>
<td>Change instances</td>
<td>Rename meta-element</td>
</tr>
<tr>
<td><strong>Breaking and unresolvable operations</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add obligatory meta-class</td>
<td>Additive</td>
<td>Add default instances</td>
<td>Eliminate meta-class</td>
</tr>
<tr>
<td>Add obligatory meta-property</td>
<td>Additive</td>
<td>Add default instances</td>
<td>Eliminate meta-class</td>
</tr>
<tr>
<td>Pull meta-property</td>
<td>Additive</td>
<td>Add default properties for superclass instances</td>
<td>Push meta-property</td>
</tr>
<tr>
<td>Restrict meta-property</td>
<td>Subtractive</td>
<td>Remove instance if non-compliant</td>
<td>Generalize meta-property</td>
</tr>
</tbody>
</table>

Note: in practice, majority (empirical: 86%) of co-evolutions can be automated (e.g., with Cope tool)
Original TrainSim Meta-Model
Evolved TrainSim Meta-Model
The Essential Pattern

\( \text{MM}_D \)

\( m \) \( T \) \( T(m) \)

\( \text{MM}_I \)
Co-evolution: Model Evolves

\[ m' = m_E \]

\[ T(m') \]

\[ T(m) \]

\[ \Delta m \]

\[ E \]

\[ M_{M_D} \]

\[ M_{M_I} \]
Co-evolution: Image Evolves

\[ \text{Modelling Language Evolution} \]
Co-evolution: Domain Evolves

\[ \Delta \text{MM}_D \]

\[ \text{MM}_D' \]

\[ \text{MM}_D \]

\[ \text{T}(m) \]

\[ \text{T'} \]

\[ \text{E} \]

\[ \text{E}^{-1} \]

\[ m \]

\[ m_E \]
Co-evolution: Transformation Evolves

\[ \Delta T' \]

\[ T'(m) \]

\[ m \]

\[ T \]

\[ T(m) \]

\[ \Delta T \]

\[ \Delta \text{MMI}_I \]

\[ \text{MMI} \]

\[ \text{MMI}' \]

\[ \text{MMD} \]
It’s not so simple ... look more closely at Languages as Sets.
Add Meta-Property
Image of Transformation Evolves

\[ m \xrightarrow{T} T(m) \xrightarrow{E} E(I) \]
Image of Transformation Evolves (example)
Domain of Transformation Evolves

\[ T'(m) = T \circ E^{-1} \]
Domain of Transformation Evolves (example)

\[ E = \text{id} \]
\[ E(D) = D' \]
\[ T(m) \rightarrow T'(m) = T \circ E^{-1} \]
Proposed Workflow

- **Additive**
- **Updative**
- **Subtractive**

v0 artifact → intermediate → v1 artifact

Migrate v0 artifact → Additive Update v1 artifact → Migrate v1 artifact
TrainSim Meta-model before evolution

Diagram:
- Train
- TrainPlace
- Rail
- Split
- Arrows indicating relationships:
  - Train to TrainPlace: 0..1
  - TrainPlace to Train: 0..1
  - TrainPlace to Rail: 1
  - TrainPlace to Split: 2

TrainSim Meta-model after evolution
Proposed Workflow: “virtual model”
“Merged and Weakened” TrainSim Meta-model

Diagram showing the relationships between Train, TrainPlace, Rail, and Junction classes. The diagram includes association multiplicities and directions.
Example of Workflow Use

- RenameMetaElement(Split, "Junction")
- RestrictMetaProperty(TrainOnTrainPlace, "left", 1, 1, 0, -1, False)
- EliminateMetaProperty(Junction.to)
- AddNonObligatoryMetaProperty(Junction, TrainPlace, "right", 1, 1, 0, -1, False)
- AddNonObligatoryMetaProperty(Junction, TrainPlace, "left", 1, 1, 0, -1, False)
- AddObligatoryMetaProperty(Rail, "length", Integer, 1, 1, 1)
- AddNonObligatoryMetaClass("RailStation", [Rail], False)

LEGEND:
- generated migration operation (L0)
- edited migration operation (L1)
- manual pass needed (L2)
Facets of Co-Evolution

- Modelling Language Evolution
- Continuity
- Consistency
- Co-evolve
- Diff
- Calculation
- Representation
  - Structural (L0)
  - Operational (L0)
  - Intermediate MM (L0)
  - Diff analysis (L0)
  - Change recording (L0)
- Source is MM
- Free-hand edit (MM edit)
- Instance edit
  - Instance consistency
- Operation selection (L1)
  - Optimize
  - Manually (L1)
  - Automatically (L0)
- Image evolution
  - Merge with E (L0)
  - Optimize
  - Manually (L1)
  - Automatically (L0)
- Domain evolution
  - Merge with E (L0)
  - Optimize
  - Manually (L1)
  - Automatically (L0)
  - Optimize
  - Manually (L1)
  - Automatically (L0)
- Instance edit
  - Instance consistency
- Pipeline part
- Updata part
- Add/sub part
- Update pass (L1)
- Manual pass (L2)
- Undirected
- Directed
- Manual part (L1)
  - Optimize
  - Manually (L1)
  - Automatically (L0)
- Instance edit
  - Instance consistency
- Convert to MM diff (L0)
  - Evolve MM
- Concrete syntaxes
- Instance co-evolution
- Mapping co-evolution
- MMk co-evolution
- Semantic similarities
- Semantic changes

- Manual modelling (L1)
- Change recording (L0)
- Diff analysis (L0)
- Structural (L0)
- Operational (L0)
- Intermediate MM (L0)
- Pipeline part
- Evential part
- Add/Sub part
- Update pass (L1)
- Manual pass (L2)
- Undirected
- Directed
- Manual part (L1)
  - Optimize
  - Manually (L1)
  - Automatically (L0)
- Instance edit
  - Instance consistency
- Convert to MM diff (L0)
  - Evolve MM
- Concrete syntaxes
- Instance co-evolution
- Mapping co-evolution
- MMk co-evolution
- Semantic similarities
- Semantic changes

36
Conclusions

- TrainSim running example
- de-constructing the co-evolution problem
- it’s not so simple . . .
- proposed workflow
- no magic
- need Higher Order Transformations (HOT)
- active area ⇒ awareness, contribution
Modelling Language Evolution

LHS

: speaker

: participant

: question

RHS

: speaker

: answer

: participant