Supporting Users to Manage Breaking and Unresolvable Changes in Coupled Evolution

Alfonso Pierantonio
joint work with
Davide Di Ruscio, Juri Di Rocco, Ludovico Iovino
Introduction

Metamodels play a key role in any **metamodelling ecosystem** since they underpin the development of a wide range of modeling artifacts.

- Similarly to any software component metamodels are expected to evolve
- It is necessary to deal with the coupled evolution problem, also called **co-evolution**
- Co-evolution involves all the artifacts co-existing in the ecosystem, e.g. models, model transformations, textual and graphical editors, and code generators.
Introduction

Metamodel changes can be classified as:
- non-breaking (NBC)
- breaking and resolvable (BRC)
- breaking and unresolvable (BUC)

depending on their impact

In the last case, user intervention is important to provide additional information, which cannot be inferred or derived.
Existing approaches (selection)

Over the last years, co-evolution has been largely investigated and different approaches have been proposed:

Model co-evolution:


Model transformations co-evolution:


Other kind of artifacts co-evolution:


Existing approaches

All these approaches aim at automating the management of non-breaking, and breaking and resolvable changes

The management of breaking and unresolvable changes is still a challenging and error-prone activity

– they are usually managed by applying default migration actions according to predefined heuristics
Proposal

In order to overcome the uncertainty or lack of information during the automated consistency restoring process, we propose a human-in-loop approach.

The user is interactively asked to contribute with the missing information in order to complete the adaptation process.
Motivating scenario
Motivating scenario
Motivating scenario
Motivating scenario
Motivating scenario
Motivating scenario

```html
<html>
<head>
  <title>Users Management</title>
</head>
<body>
  <script type="text/javascript">
    window.app = {
      models: {},
      collections: {},
      templates: {},
      views: {},
      controllers: {},
      router: null
    };
    </script>
  ...
  <script type="text/javascript" src="models/user.js"></script>
  ...
  <script type="text/javascript" src="router.js"></script>
</body>
</html>
```

Evolving metamodel

- WebApp Metamodel
- metamodel evolution
- WebApp Metamodel'

Migrating artifacts

- WebApp Model
- model migration
- WebApp Model'

WebApp2HTML5 Transformation
- transformation migration
- WebApp2HTML5 Transformation'
Metamodel Evolution

Applied changes in the motivating scenario:

1. split attribute
2. change reference type
3. extract metaclass
Split attribute co-evolution

The expression \([\text{binding}.\text{url}]\) has to be migrated, however it is uncertainty whether to be replaced by

- \([\text{binding}.\text{requestUrl}], \text{or}\)
- \([\text{binding}.\text{requestCookies}]\)
Change Reference Type co-evolution

The template `generateViewData` cannot be invoked on elements with type `Data` anymore (because of the reference type change)

Corrupted excerpt of the code generator
Extract Metaclass co-evolution

The expression \[aTemplate.style\] cannot be evaluated anymore, consistency can be restored with:

- \[aTemplate.style.src\]
- \[aTemplate.style.href\]
Proposed Adaptation Approach

The adaptation process relies on an extension of EMFMigrate realized by means of the epsilon platform.

Proposed Adaptation Approach

The adaptation flow is formalized by the following sequence diagram.

[Sequence diagram image]
Proposed Adaptation Approach

EMFMigrate is a co-evolution language for EMF-based artifact migration

- It consists of migration rules applied on a given artifact $A$ conforming to a metamodel $MM$
- EMFMigrate has been extended to generate Epsilon programs
Migration Programs

- A (automatically generated) ANT specification orchestrates the control-flow above
- Whenever necessary the user is prompt for necessary information, which permits to resolve uncertainty points
Split attribute co-evolution

The expression \([binding\textunderscore url}\)] has to be migrated, however it is uncertainty whether to be replaced by

- \([binding\textunderscore requestUrl}\), or
- \([binding\textunderscore requestCookies}\)

Metamodel evolution

Corrupted excerpt of the code generator
Change Reference Type co-evolution

Metamodel evolution

Corrupted excerpt of the code generator

The template `generateViewData` cannot be invoked on elements with type `Data` anymore (because of the reference type change).

```
[template public generateViewData(view : View)]
[template public generateAttribute(attribute : Attribute)]
[template public generateModelAttribute(attribute : Attribute)]
```

generateViewData is not compatible anymore with View.data, please select the template to use:

```
Template [name=generateAttribute, ordered=true, unique=true, lowerBound=0, upperBound=1, many=false, required=false]
Template [name=generateModelAttribute, ordered=true, unique=true, lowerBound=0, upperBound=1, many=false, required=false]
```
The expression `[aTemplate.style]` cannot be evaluated anymore, consistency can be restored with:

- `[aTemplate.style.src]`
- `[aTemplate.style.href]`
Conclusions

- An approach has been proposed to pragmatically address the adaptation of artifacts affected by BUCs
- A general adaptation process has been proposed with the ability to interact with the user in case of BUCs with finite resolution options
- Although it has been demonstrated on Acceleo templates, the approach is ready to be applied to any EMF-based artifact

- Future works include a off-the-shelf catalog of migration programs in accordance with the refactorings catalog [http://www.metamodelrefactoring.org/](http://www.metamodelrefactoring.org/)