Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Università degli Studi dell'Aquila

Automated chaining of model transformations with incompatible metamodels

Francesco Basciani *

Computer Science Department, University of L'Aquila, Italy

* Join work with: Davide Di Ruscio, Ludovico Iovino, Alfonso Pierantonio.
Special thanks to Juri Di Rocco

Roadmap

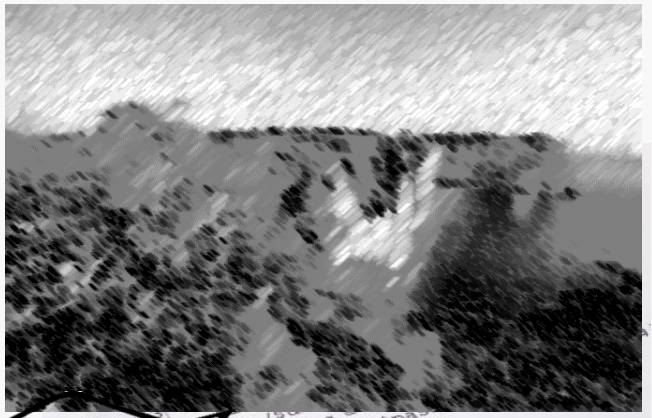
- Background
- Automating Model Transformation Chain
- Adapters
- Implementation - Demo
- Conclusions and future work

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

BACKGROUND

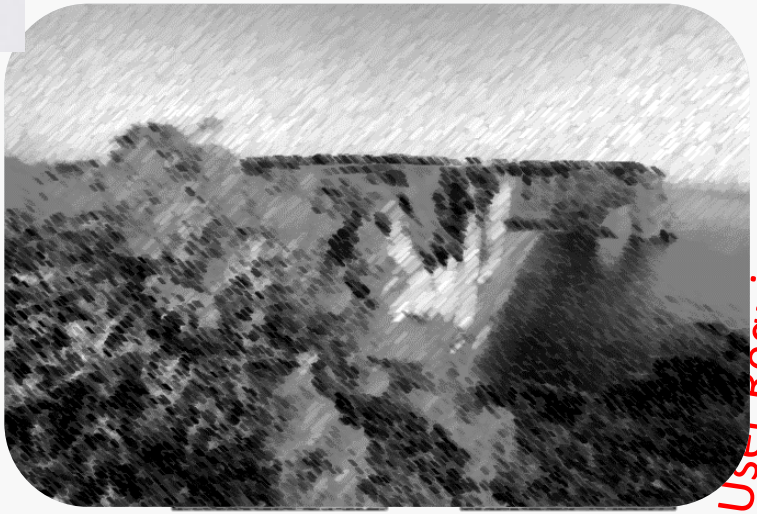
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Overview



Hand write the whole transformation

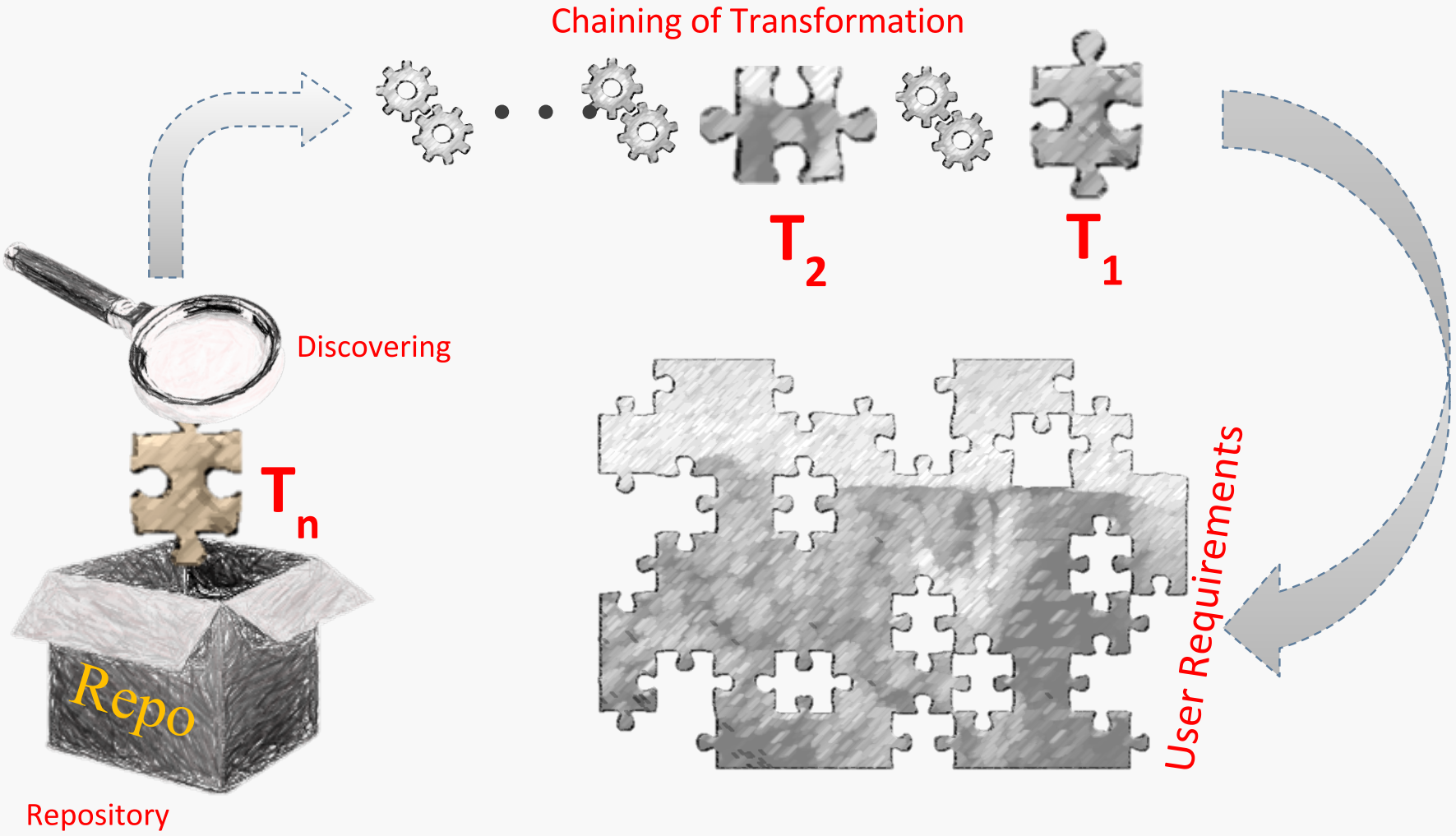
T



User Requirements

USE

Overview


Chaining of Transformation
T2
T1
Discovering
Tn
Repo
Repository
User Requirements
RE-USE

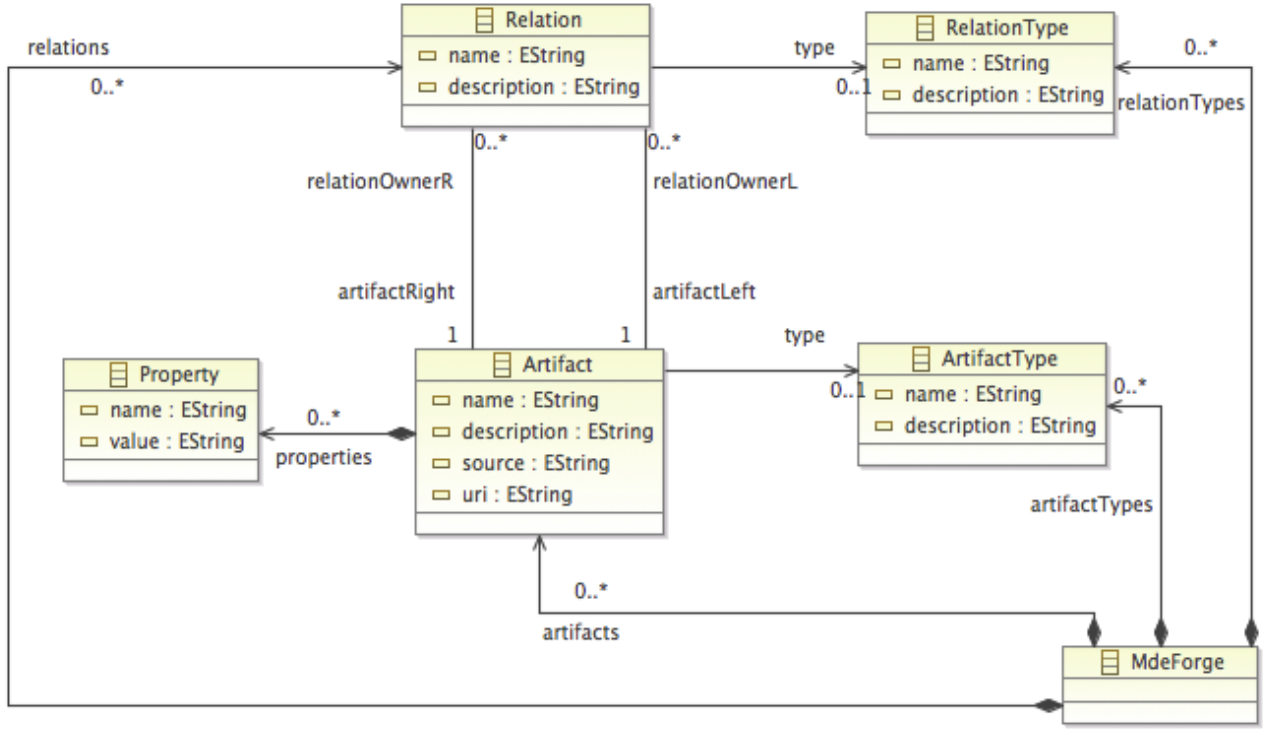
Repository structure

It is a structured repository of artifacts that are properly organized to enable their reuse.

It is constructed from a metamodel, general enough to allow to store artifacts of any type:

- models;
- metamodels;
- transformatios;
- ecc…

Keeping the relationship among them.

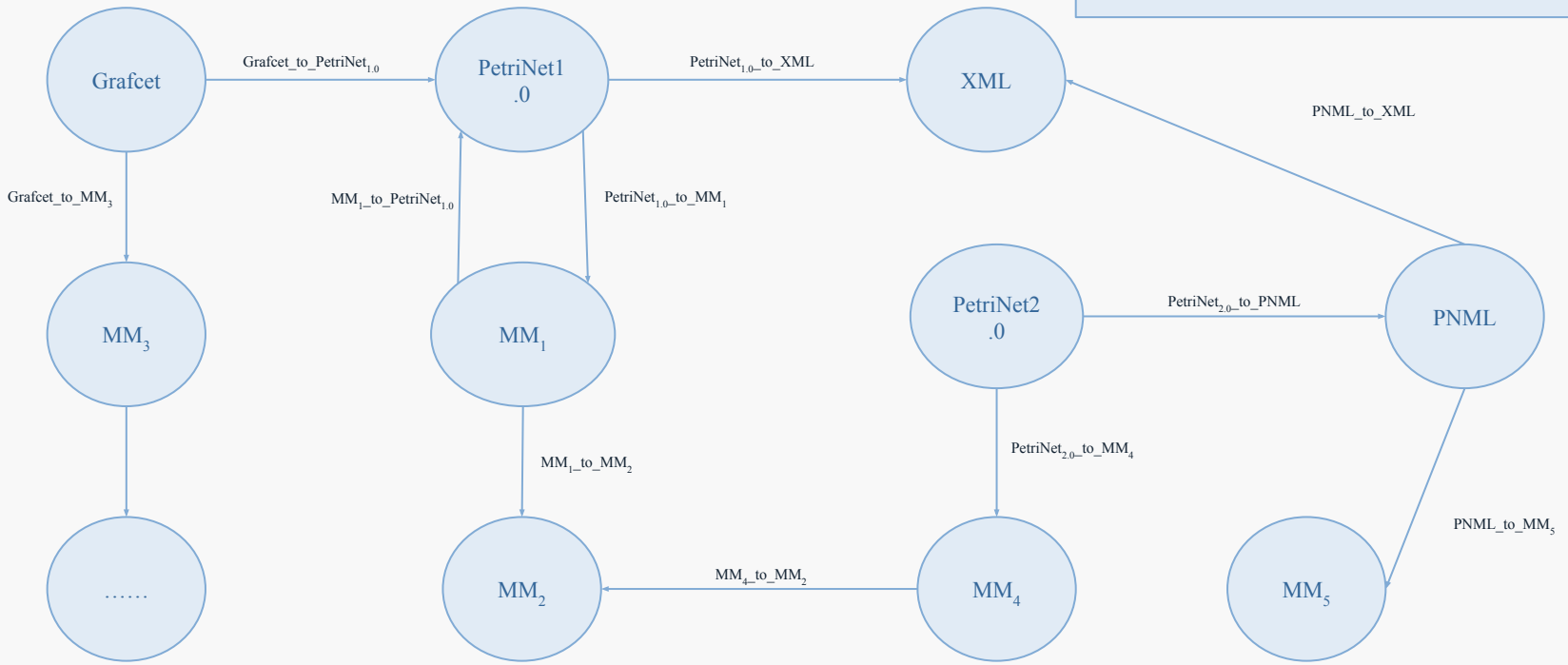


Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Repository graph representation

Example of repository:

It is represented as a graph



- **Nodes** represent metamodels
- **Arcs** represent Transformation

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Type of Composition*

- Internal Composition

  A new transformation is defined as the transformations to be composed.

- External Composition

  Chaining separate model transformations and passing models from one transformation to another

* With the term composition we refer to external composition.
Moreover, the terms composition and chaining are used interchangeably.

Type of Composition*

- Internal Composition

  A new transformation is defined as the transformations to be composed.

- External Composition

  Chaining separate model transformations and passing models from one transformation to another

* With the term composition we refer to external composition.
Moreover, the terms composition and chaining are used interchangeably.

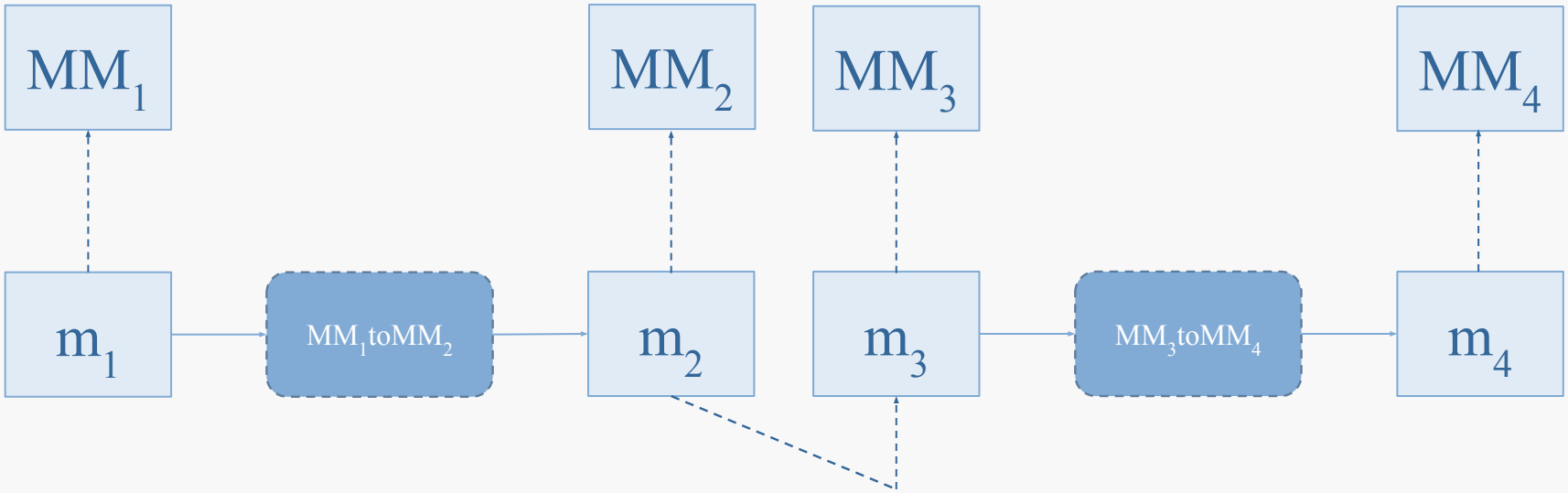
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Chain of Transformation

The common way to compose transformations is to chain them by passing models from one transformation to another.

In order to chain transformations it is necessary to ensure the pre- and post-conditions of the considered transformations and to verify the metamodels compatibility condition.

Chain of Transformation



In order to create a chain, the main condition to ensure is that the metamodel output of a transformation must be conform to the metamodel input of the next.

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

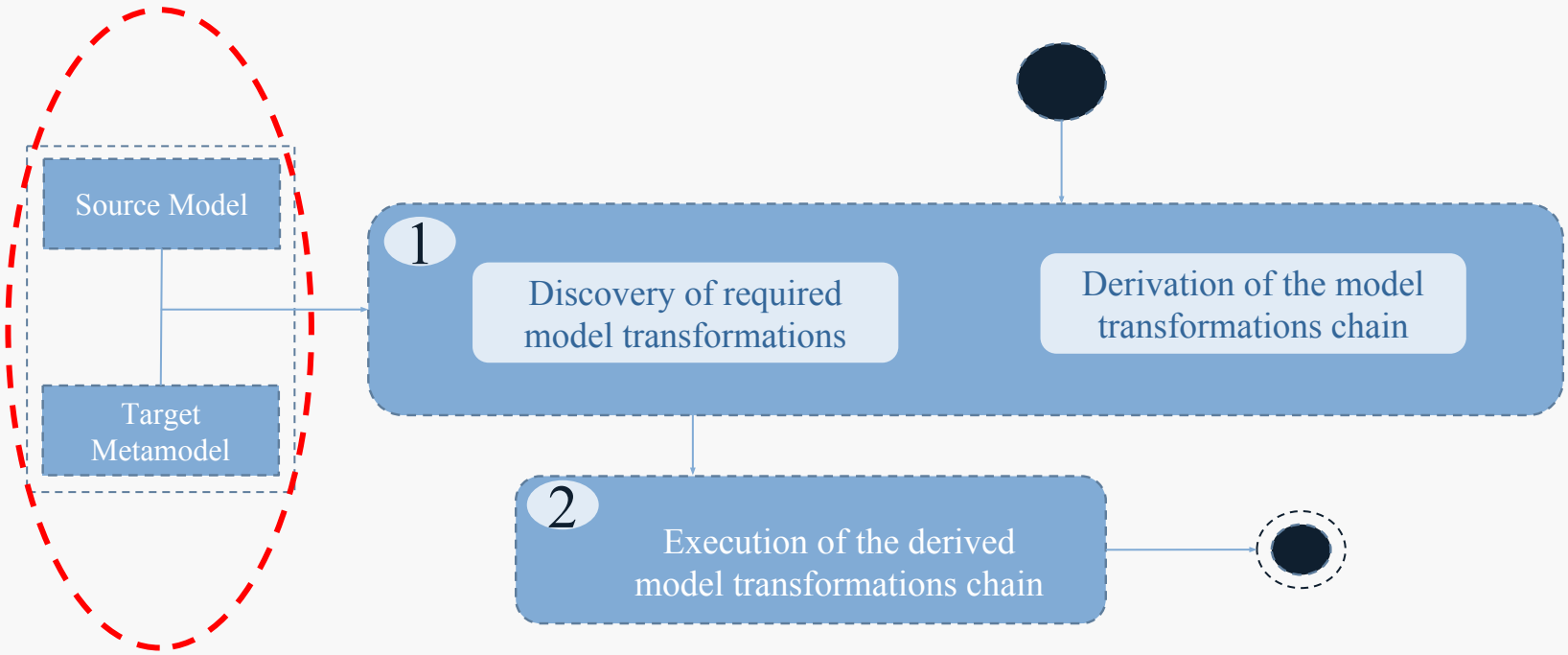
AUTOMATING MODEL TRANSFORMATION CHAINING PROCESS

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Model Transformations Chaining Process

What we propose is to **automate** the process of creating a chain so that the user needs to provide only:

- source model;
- target metamodel.

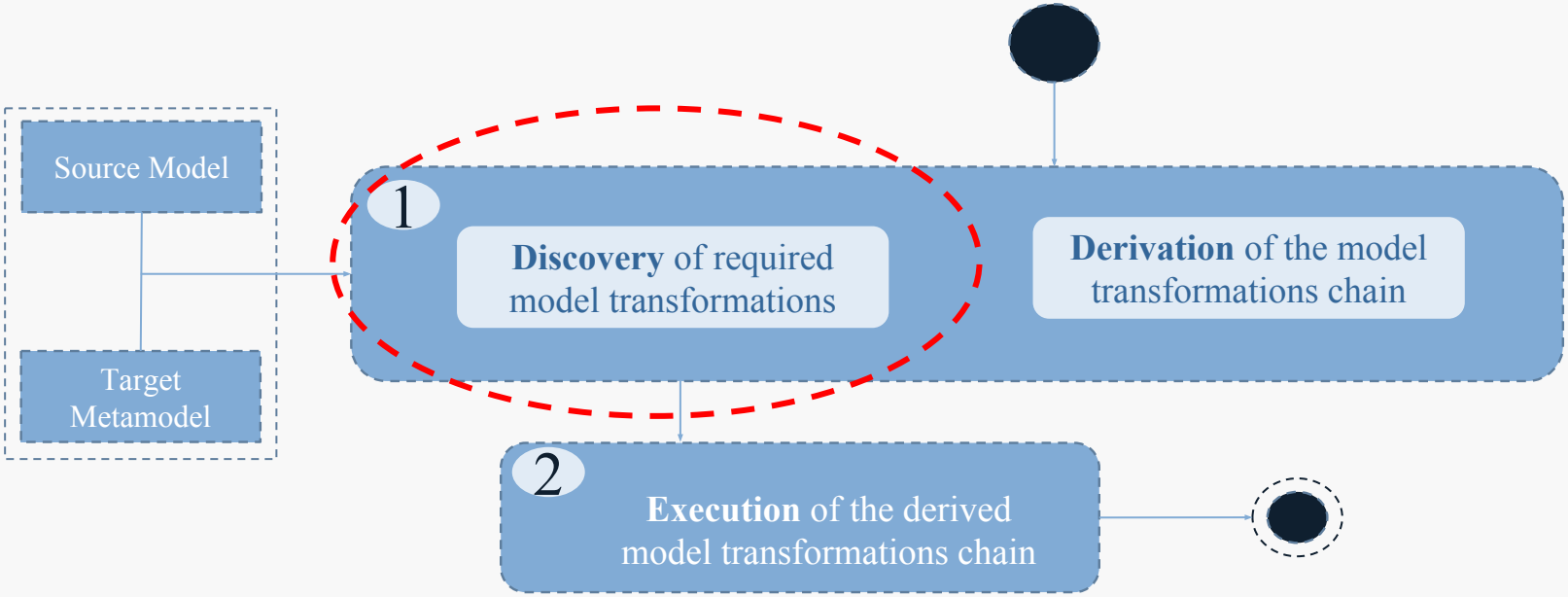


Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Model Transformations Chaining Process

Discovery of required model transformations

refers to the repository and search the transformations that can be compatible for a chain;

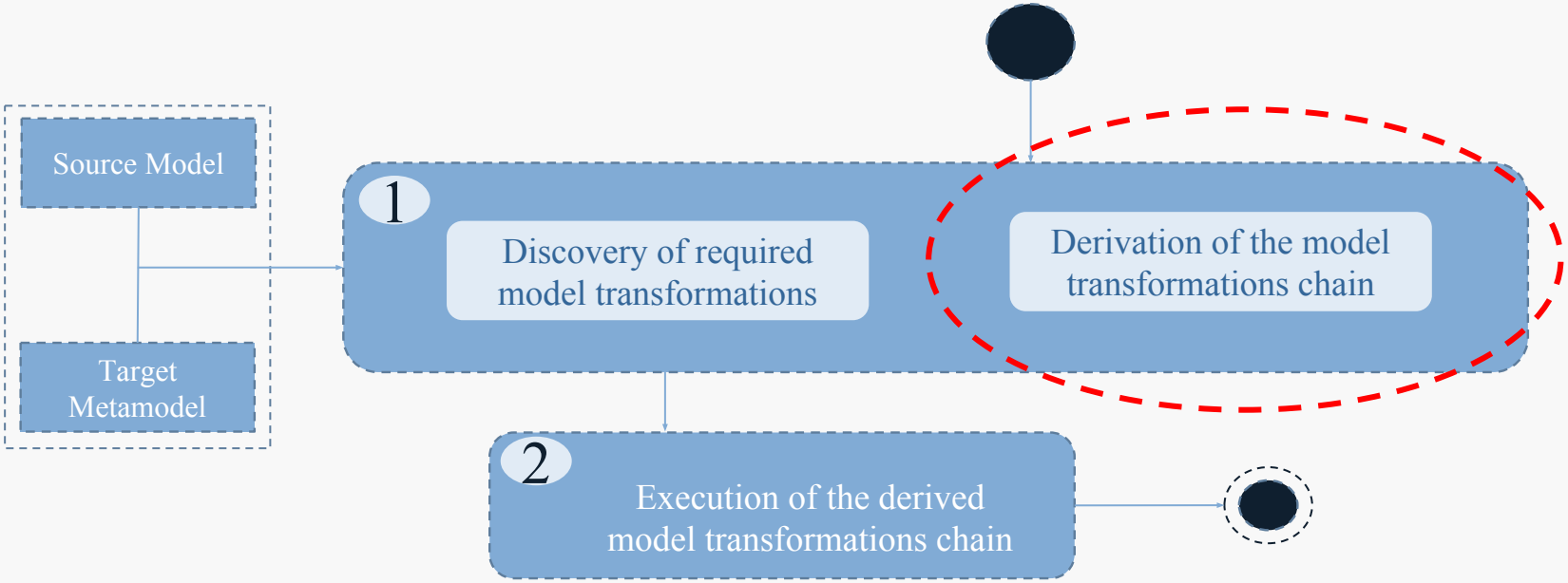


Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Model Transformations Chaining Process

Derivation of the model transformations chain

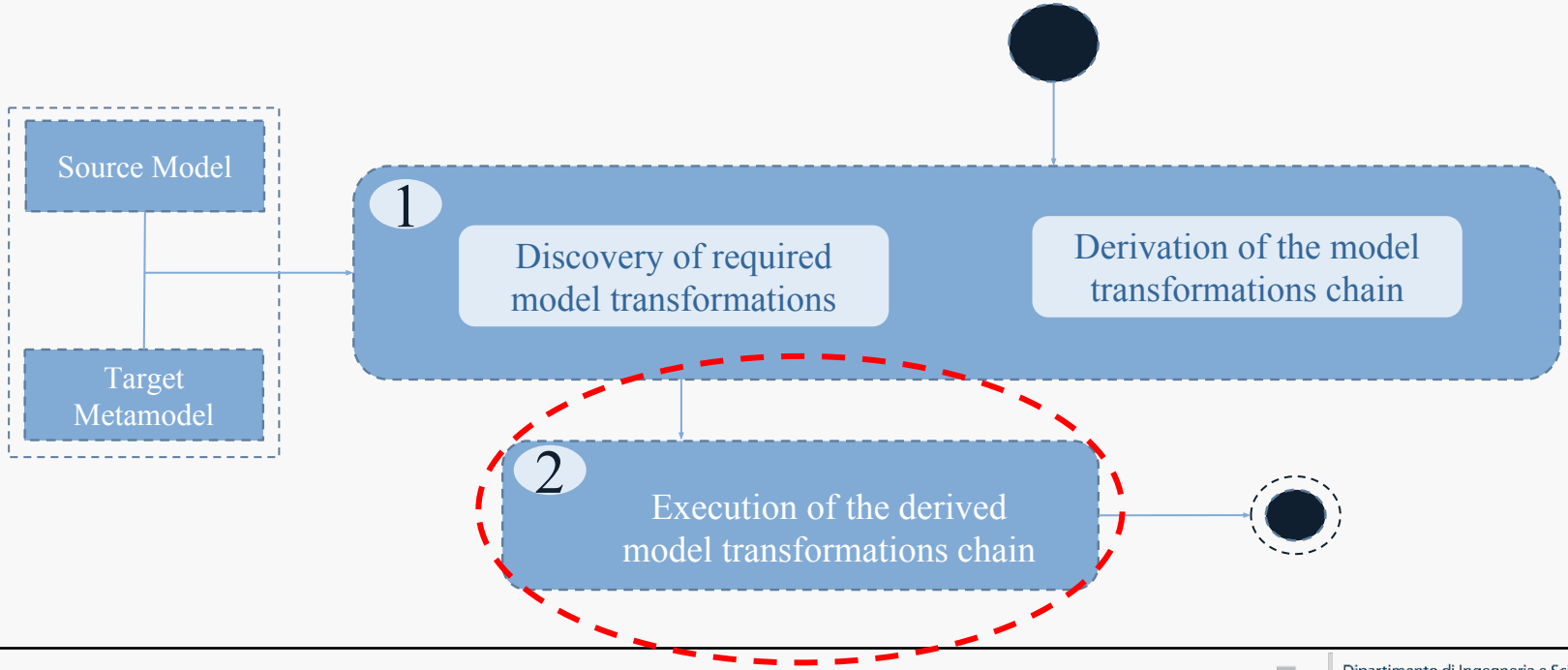
given directed graph representation solve the user's request and thus from the source metamodel **search one or more paths that leave from the source and arrive to the target metamodel** (user supplied).



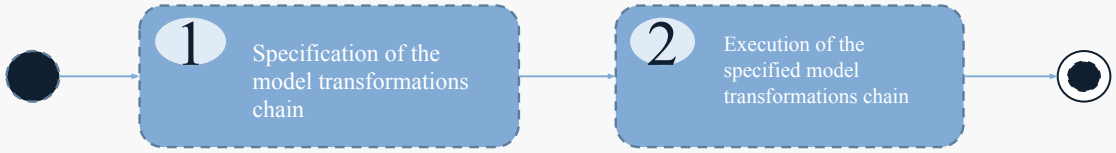
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

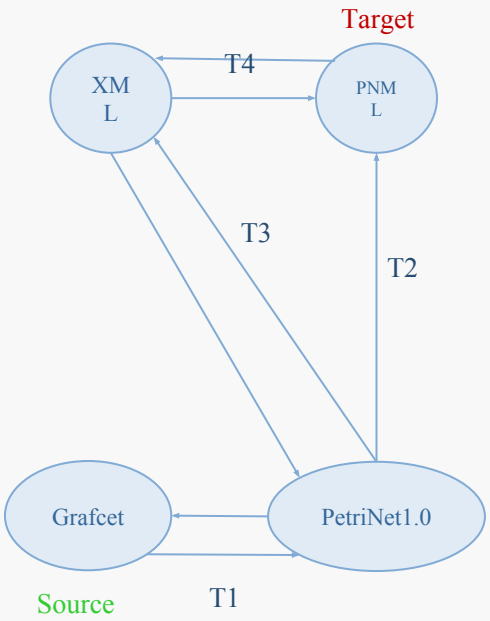
Model Transformations Chaining Process

The transformations present within the chain of transformations are performed in order.



Dealing with Multiple chains





The activity 1 might give place to different possible chains and the users have to select one of them.

If the user gives as input a **Grafcet** model and wants to generate a target **PNML**. The request can be satisfied by two possible chains, i. e.:

- T1 → T2;
- T1 → T3 → T4.

In these cases it is necessary to consider one or more **evaluation criteria** to select one of the derived chains:

- **Coverage of the source metamodels**;
- **Usage;**
- **Number of transformations;**
- **Execution time**.

Compatibility & Composability

Essential to create a chain are the notions of compatibility and composability:

Definition 1 - (Metamodel compatibility):

Let MM_1 and MM_2 be two metamodels, then MM_1 and MM_2 are compatible if

$$MM_1 \subseteq MM_2 \quad or \quad MM_2 \subseteq MM_1$$

Definition 2 - (Transformation composability):

Let $T_1 : MM_1 \rightarrow MM_2$ be a model transformation from the metamodel MM_1 to the metamodel MM_2.

Let $T_2 : MM_3 \rightarrow MM_4$ be a model transformation from the metamodel MM_3 to the metamodel MM_4.

Then, T_1 and T_2 are composable as $T_1 \circ T_2 (T_2 \circ T_1)$ if $MM_4 \subseteq MM_1$ ($MM_2 \subseteq MM_3$).

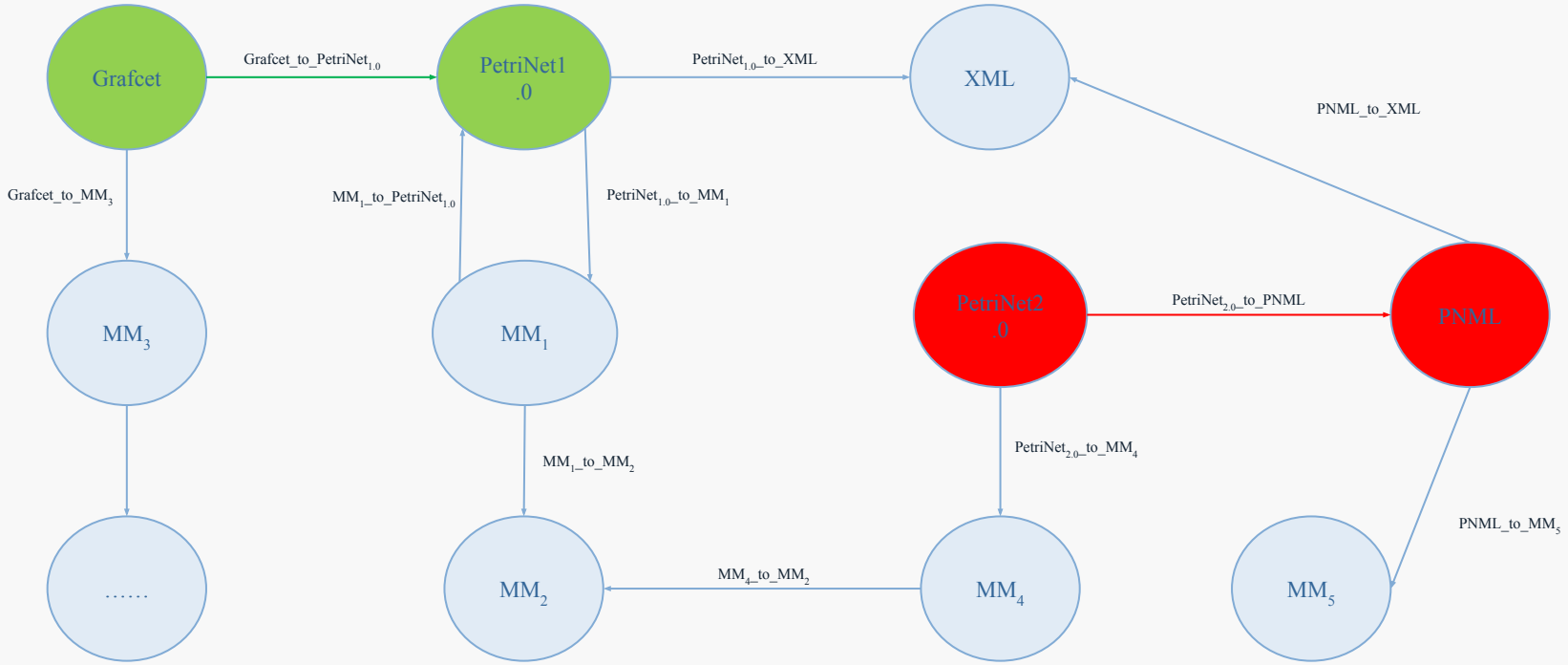
Problem ➡ Restricting the definition of transformation chains only for the cases of compatible metamodels can **reduce the number of chains that might be potentially obtained**.

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Example

Suppose that the user wants to:

- **start** from a model **Grafcet**
- **get** a **PNML** model.



- **Nodes** represent metamodels
- **Arcs** represent Transformation

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Incompatibility

Due to the **definition of transformation composability** it is impossible to have a chain

$$Grafcet \longrightarrow \ldots \longrightarrow PNML$$

Because starting from the transformations $Grafcet\_to\_PetriNet_{1.0}$ and $PetriNet_{2.0}\_to\_PNML$

$$\boldsymbol{PNML \not\subseteq Grafcet}$$

and

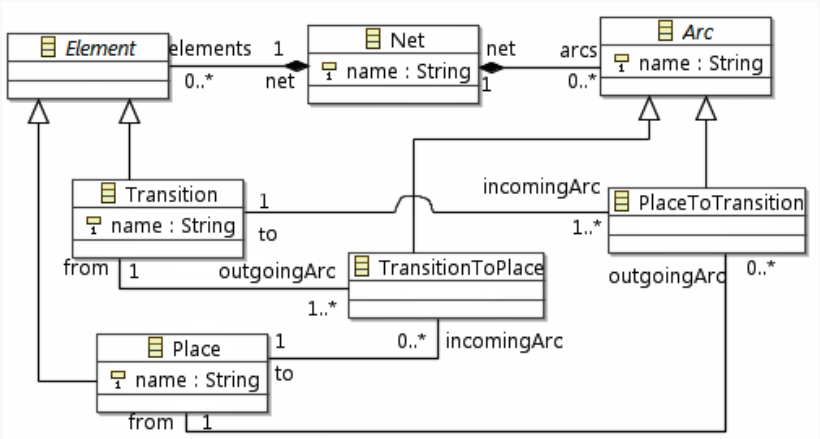
$$\boldsymbol{PetriNet_{1.0} \not\subseteq PetriNet_{2.0}}$$

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

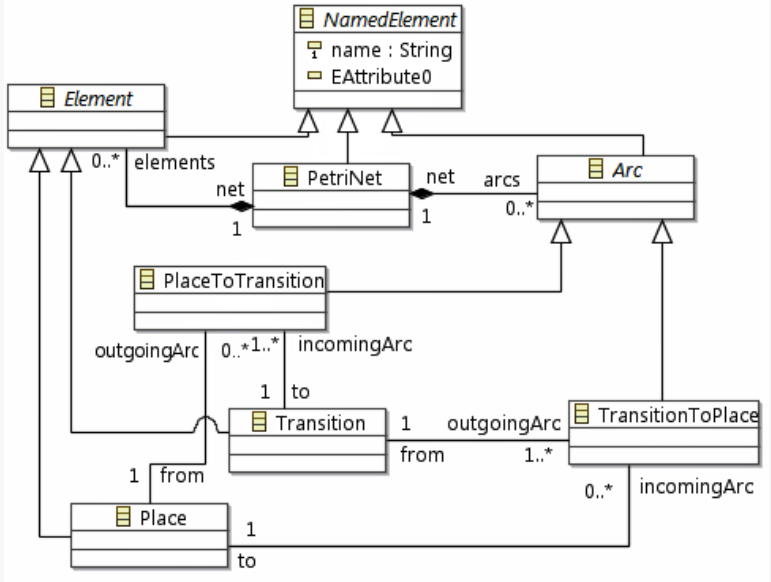
$PetriNet_{1.0}$ vs $PetriNet_{2.0}$

$PetriNet_{2.0}$ is a simple evolution of $PetriNet_{1.0}$. in particular, the differences between the two versions are:

δ_1 - pull up of the attribute *name* to the new abstract metaclass *NamedElement*;
δ_2 - renaming of the metaclass *Net* as *PetriNet*.



$PetriNet_{1.0}$



$PetriNet_{2.0}$

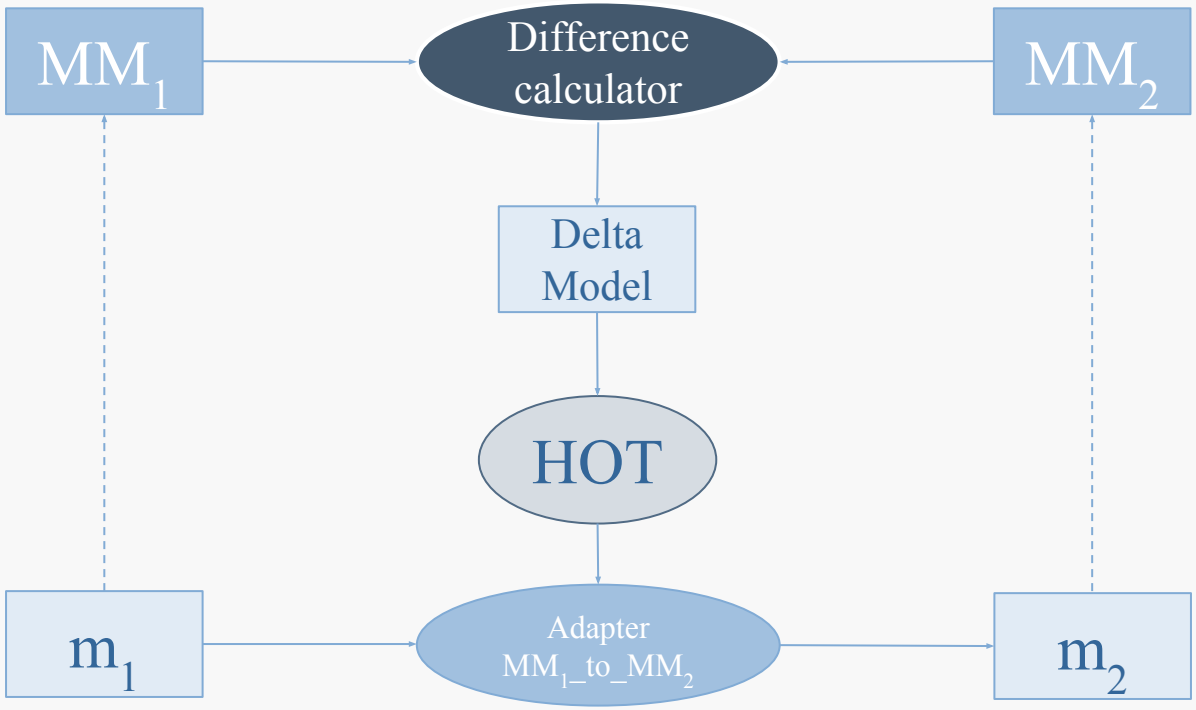
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

ADAPTERS

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Generation of the Adapter transformation

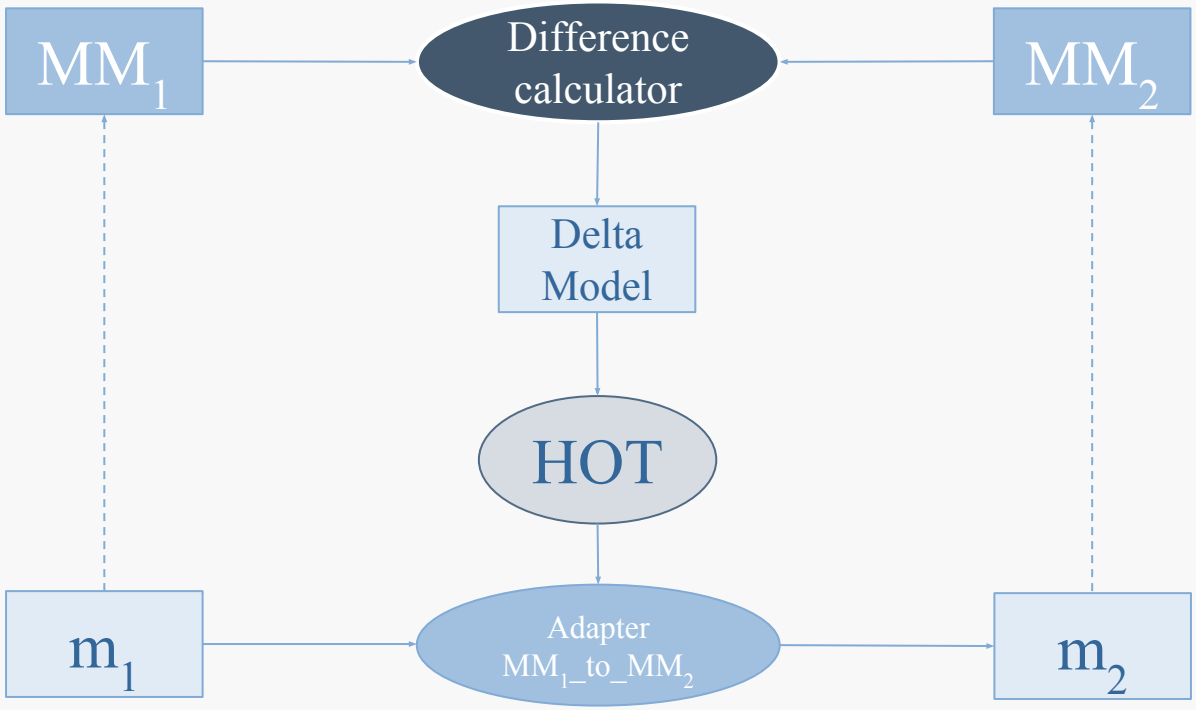
Our main contribution is focused in proposing an approach that (under certain conditions) permits to chain model transformations defined on **incompatible metamodels.**



Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Generation of the Adapter transformation

This is done by means of an adapter transformation that can be automatically synthesized from a **delta model** that represent the differences between the output and input metamodels of the transformations to be chained.



Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Coupled-evolution Problem

This discussion relates to the coupled-evolution problem.

In such contexts metamodel manipulations can be classified by their corrupting or not-corrupting effects on corresponding artifacts as:

- **non-breaking changes**: changes which do not break the conformance of models to the corresponding metamodel;
- **breaking and resolvable changes**: changes which break the conformance of models even though they can be automatically co-adapted;
- **breaking and unresolvable changes**: changes which break the conformance of models which can not automatically co-evolved and user intervention is required.

Similarity function

In order to perform in automatic way an adaptation we have to introduce the **similarity function** used to calculate a similarity value among all the metamodels stored in the repository.

The function is launched every time a new transformation is stored (or deleted, or updated) into the repository.

This function assigns a value of similarity between the input and target metamodels of the transformation respect to all other metamodels already present in the repository.

*For our purposes of tests on the prototype we used the function of similarity of "Falleri" that makes use of the **Similarity Flooding Algorithm** to calculate the similarity between two graphs.*

Similarity function

The similarity values are maintained in a table like this:

	Grafcet	$PetriNet_{1.0}$	$PetriNet_{2.0}$	XML	PNML
Grafcet	1	0,20	0,30	0,29	0,26
$PetriNet_{1.0}$	-	1	0,89	0,20	0,28
$PetriNet_{2.0}$	0,30	0,89	1	0,30	0,30
XML	0,29	0,20	0,30	1	-
PNML	0,26	-	-	0,28	1

If the similarity value between two considered metamodels is higher then a fixed threshold (i.e. **0,80**) such metamodels are further analyzed.

Then, if the resulting **delta model** consists of **non-braking** and **breaking and resolvable changes**, then corresponding adapter transformations can be generated.

Adapter$_{\text{PetriNet1.0,PetriNet2.0}}$

Referring to our example (differences between $\text{PetriNet}_{1.0}$ and $\text{PetriNet}_{2.0}$), that the similarity value exceeds the threshold.

Furthermore, by analyzing the differences between the two metamodels we have:

δ_1 - pull up of the attribute *name* to the new abstract metaclass *NamedElement*;
δ_2 - renaming of the metaclass *Net* as *PetriNet*.

Where:
δ_1 is a **non-breaking change**;
δ_2 is **breaking and resolvable change**.

So our approach can be applied and the adapter generation is completely automated without requiring user intervention.

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Adapter$_{PetriNet1.0,PetriNet2.0}$

All the rules of the ***PetriNet$_{2.0}$_to_PNML*** transf. that refer to the pulled up attribute *name* are still valid and do not require any adaptation.

```
rule PetriNet {
    from
        g : Grafcet!Grafcet
    to
        p : PetriNet!Net
        (
            location <- g.location,
            name <- g.name,
            elements <- g.elements,
            arcs <- g.connections
        )
}
```

Grafcet_to_PetriNet$_{1.0}$

```
rule PNMLDocument {
    from
        e : PetriNet!PetriNet
    to
        n : PNML!PNMLDocument
        (
            location <- e.location,
            xmlns <- uri,
            nets <- net
        ),
        uri : PNML!URI
        (
            value <- 'http://www.informatik.hu-be
        ),
        net : PNML!NetElement
        (
            name <- name,
            location <- e.location,
            id <- e.location,
            type <- type_uri,
            contents <- e.elements.union(e.arcs)
        ),
```

PetriNet$_{2.0}$_to_PNML

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Adapter$_{PetriNet1.0,PetriNet2.0}$

The rules of ***PetriNet$_{2.0}$_to_PNML*** whose source patterns refer to *PetriNet* elements can be still applied on *Net* elements after an adaptation phase that simply copies all the *Net* elements to target *PetriNet* ones (Rename).

Thus the rule PNMLDocument can be applied on *Net* elements generated by ***Grafcet_to_PetriNet$_{1.0}$*** after an adaptation step that generates *PetriNet* elements from the generated *Net* ones.

```
8  rule PetriNet {
9      from
10         g : Grafcet!Grafcet
11     to
12         p : PetriNet!Net
13         (
14             location <- g.location,
15             name <- g.name,
16             elements <- g.elements,
17             arcs <- g.connections
18         )
19 }
```

Grafcet_to_PetriNet$_{1.0}$

```
7  rule PNMLDocument {
8      from
9          e : PetriNet!PetriNet
10     to
11         n : PNML!PNMLDocument
12         (
13             location <- e.location,
14             xmlns <- uri,
15             nets <- net
16         ),
17         uri : PNML!URI
18         (
19             value <- 'http://www.informatik.hu-be
20         ),
21         net : PNML!NetElement
22         (
23             name <- name,
24             location <- e.location,
25             id <- e.location,
26             type <- type_uri,
27             contents <- e.elements.union(e.arcs)
28         ),
```

PetriNet$_{2.0}$_to_PNML

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Adapter$_{PetriNet1.0,PetriNet2.0}$

More precisely, the adaptation step can be performed by means of the Adapter transformation

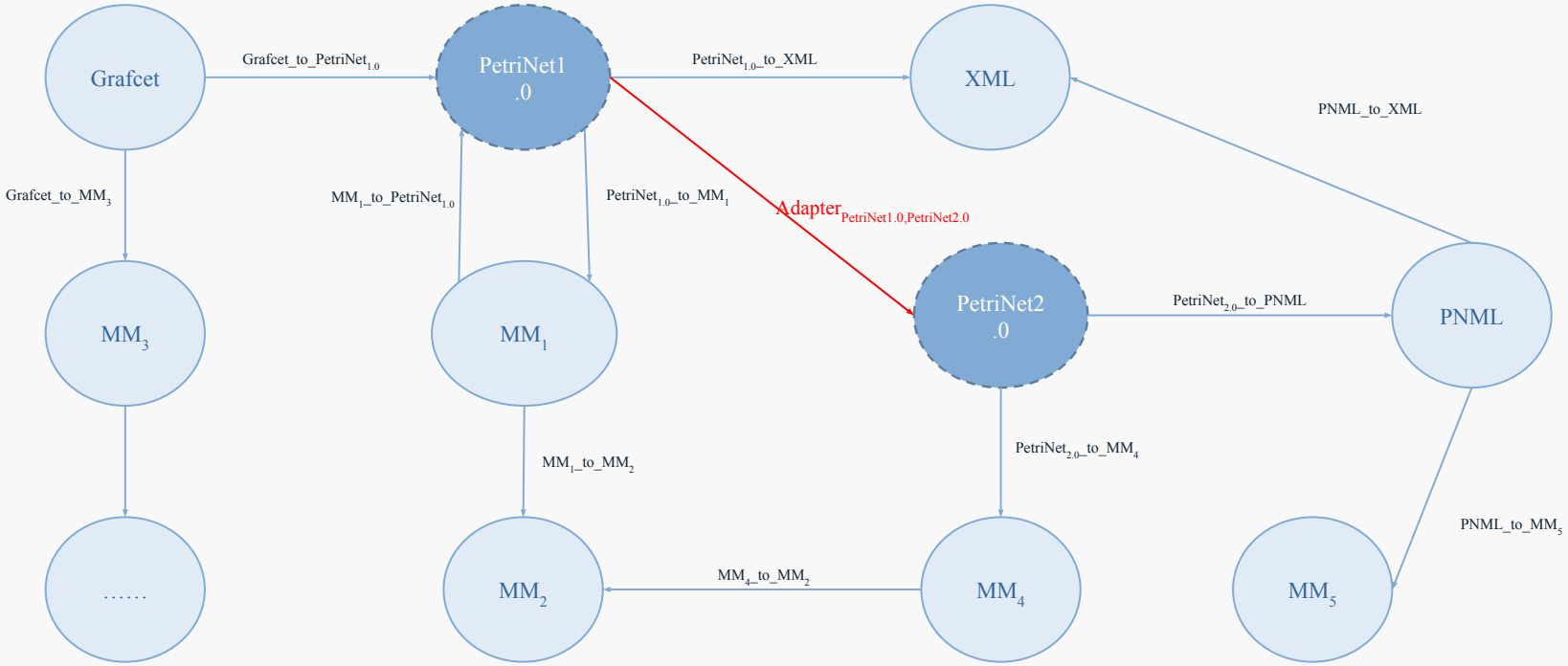
```
21 rule Net2Petrinet {
22     from
23         s : PetriNetv1!Net
24     to
25         t : PetriNetv2!PetriNet
26         (
27             ...
28             name <- s.name
29         )
30 }
31 ...
```

(Fragment of the adapter transformation *Adapter*$_{PetriNet1.0,PetriNet2.0}$)

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Adapter$_{PetriNet1.0,PetriNet2.0}$

With this adapter the repository is update:



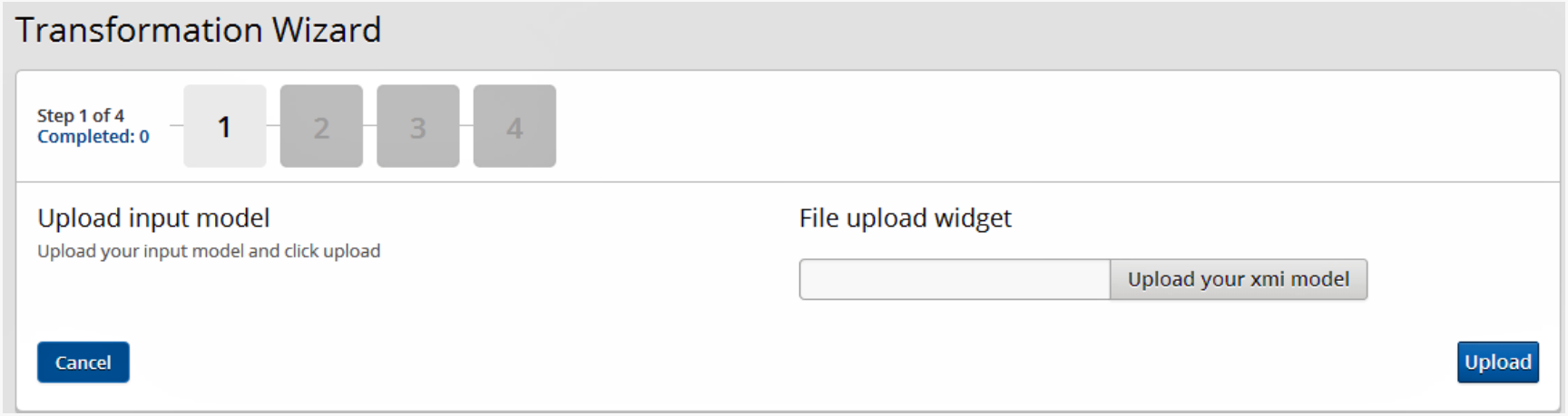
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

IMPLEMENTATION - DEMO

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Screen – 1/ Upload of the source model

In the first step the user has to upload his model

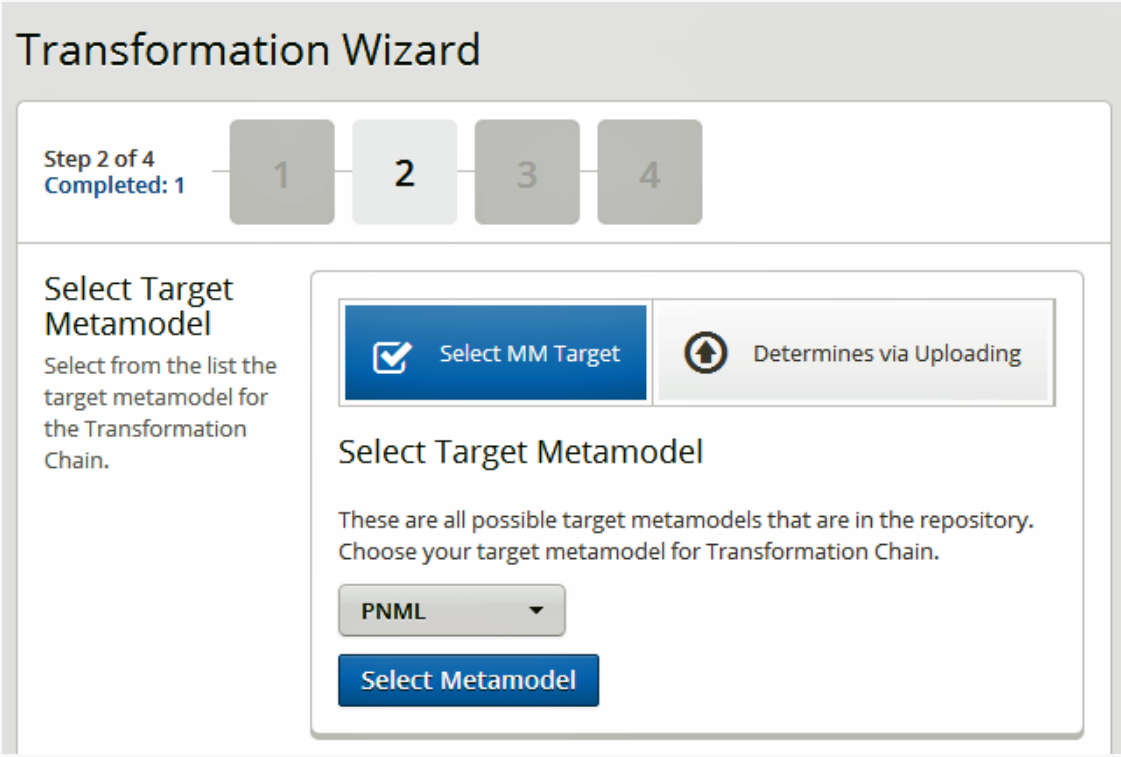


Link to the online web application: www.mdeforge.org

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Screen – 2/ Selection of target metamodel

Once the model has been uploaded, the system is able to detect the metamodel the model conforms to. Afterwards, the user has to select the target metamodel as shown in this screenshot:

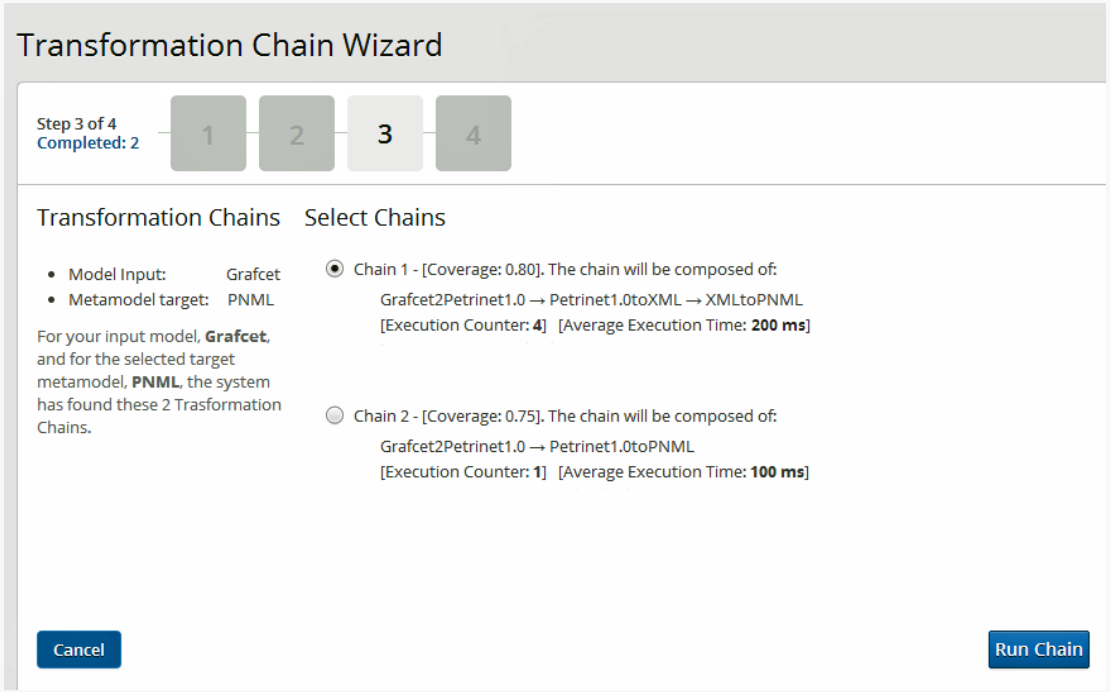


The list of metamodels shown to the user is retrieved from the repository used to store transformations and all the metamodels required to execute them.
The system will only search in the repository the metamodels target of all these transformations.
Once the target metamodel is selected, the whole activity 1 of the process is executed.

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Screen – 3/ Transformation chain selection

In the third step, all the possible chains that satisfy the user request are shown.

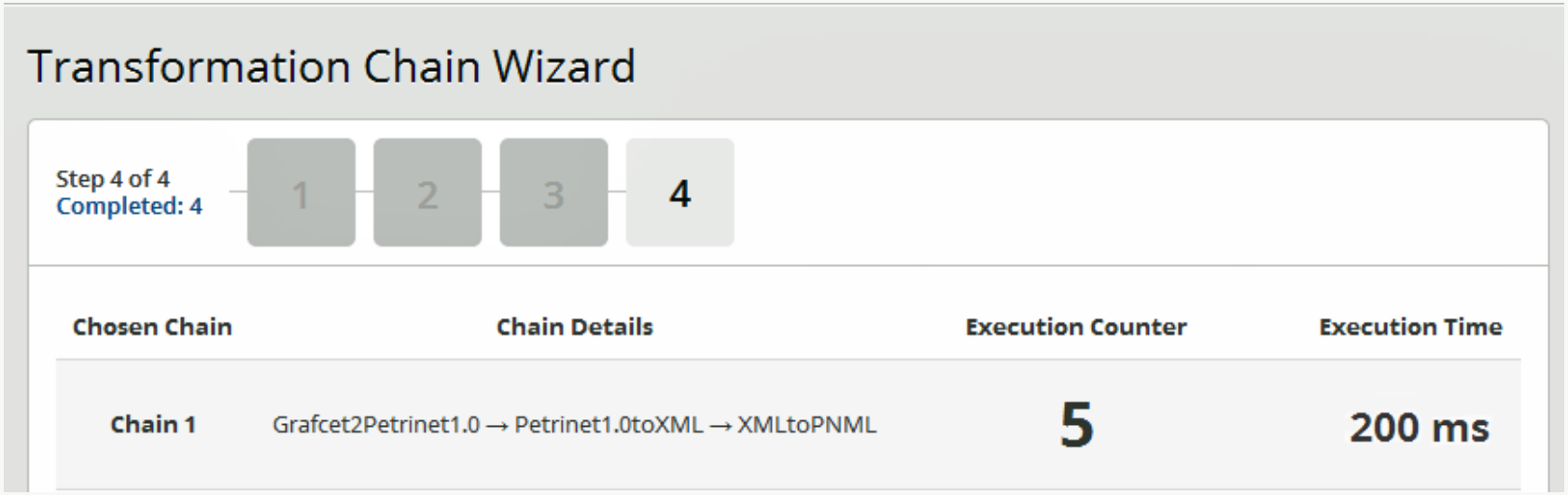


To help users in the selection, each chain is characterized by different attributes, like the number of single transformations that will be executed, the **coverage** with respect to the source metamodel**, how many times** the chain has been already executed, and its average **execution time**. Once the chain is selected, then the user can trigger its execution and wait for the generated model.

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Screen – 4/ Report about the executed chain

Once the chain has been executed some statistical data are stored in the system and shown to the user as report.



Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

CONCLUSION AND FUTURE WORK

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila

Discussion and Future work 1/

The main strengths of the approach are related to the possibility of chaining transformations that would be discarded if is considered only the notion of metamodel compatibility.

However, the approach we proposed can be enhanced in different directions:

Management of breaking and unresolvable changes:

with a **breaking and unresolvable change** (BUR) we might exclude some feasible transformation chains.

Managing BUR is complex and require user intervention also in “coupled-evolution” area.

(this is why we decided to do not manage BUR changes at a first stage and investigate them as a future work)

Dipartimento di Ingegneria e Scienze dell’Informazione e Matematica
Università degli Studi dell’Aquila

Discussion and Future work 2/

Metamodel similarity functions:

Our implementation is independent from the similarity function in the sense that, when a new similarity function is considered to be more appropriate, then we have to refining only one method of a dedicated Java class.

Missing the management of semantic aspects of the transformations:

We are based only on structural aspect of transformations and metamodels.

Dealing with semantics of model transformations and in MDE in general is complex and represents an interesting and challenging research direction to pursue for extending the proposed approach.

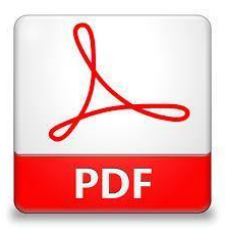
Experimentation and evaluation:

The approach has been validated by considering the transformations available in the ATL Zoo.

To better assess the validity it is necessary to consider an extended set of data (whose definition is a complex task per se).

Applications

This work was been accepted to
MODELS 2014 Conference



Basciani F., Di Ruscio D., Iovino L., Pierantonio A.: Automated chaining of model transformations with incompatible metamodels. In: MODELS 2014, Valencia, Spain.

References

1. Etien, A., Aranega, V., Blanc, X., Paige, R.F.: Chaining Model Transformations. In: Proceedings of the First Workshop on the Analysis of Model Transformations. AMT '12, New York, NY, USA, ACM (2012) 9–14

2. Etien, A., Muller, A., Legrand, T., Blanc, X.: Combining Independent Model Transformations. In: Proceedings of the 2010 ACM Symposium on Applied Computing. SAC '10, New York, NY, USA, ACM (2010) 2237–2243

3. Vanhooff, B., Baelen, S.V., Hovsepyan, A., Joosen, W.: Towards a Transformation Chain Modeling Language (2006)

4. Etien, A., Muller, A., Legrand, T., Paige, R.F.: Localized model transformations for building large-scale transformations. Software Systems Modeling (2013) 1–25

5. Wagelaar, D.: Composition Techniques for Rule-Based Model Transformation Languages. In Vallecillo, A., Gray, J., Pierantonio, A., eds.: Theory and Practice of Model Transformations. Volume 5063 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2008) 152–167

References

1. Di Ruscio, D., Iovino, L., Pierantonio, A.: Coupled Evolution in Model-Driven Engineering. IEEE Software 29 (2012) 78–84

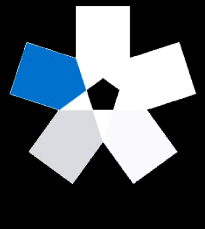
2. Cicchetti, A., Di Ruscio, D., Pierantonio, A.: A Metamodel Independent Approach to Difference Representation. Journal of Object Technology 6 (2007) 165–185

3. Voigt, K.: Structural Graph-based Metamodel Matching. PhD thesis (2011)

4. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proceedings. 18th International Conference on Data Engineering, 2002. (2002) 117–128

5. Falleri, J.R., Huchard, M., Lafourcade, M., Nebut, C.: Metamodel Matching for Automatic Model Transformation Generation. In Czarnecki, K., Ober, I., Bruel, J.M., Uhl, A., Vlter, M., eds.: Model Driven Engineering Languages and Systems. Volume 5301 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2008) 326–340

Link to the online web application: www.mdeforge.org



Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Università degli Studi dell'Aquila