POLITECNICO DI MILANO

Massimo Tisi
tisi@elet.polimi.it

MSDL

Cross-Language Comparison and Mutation Analysis of Model Transformations

Outline

- The number of model transformation languages is growing but:
  - transformation languages seem to share a common set of features
  - they seem to belong to a limited set of categories
- Need for:
  - Cross-language theoretical results
  - Unified tools

- Outline

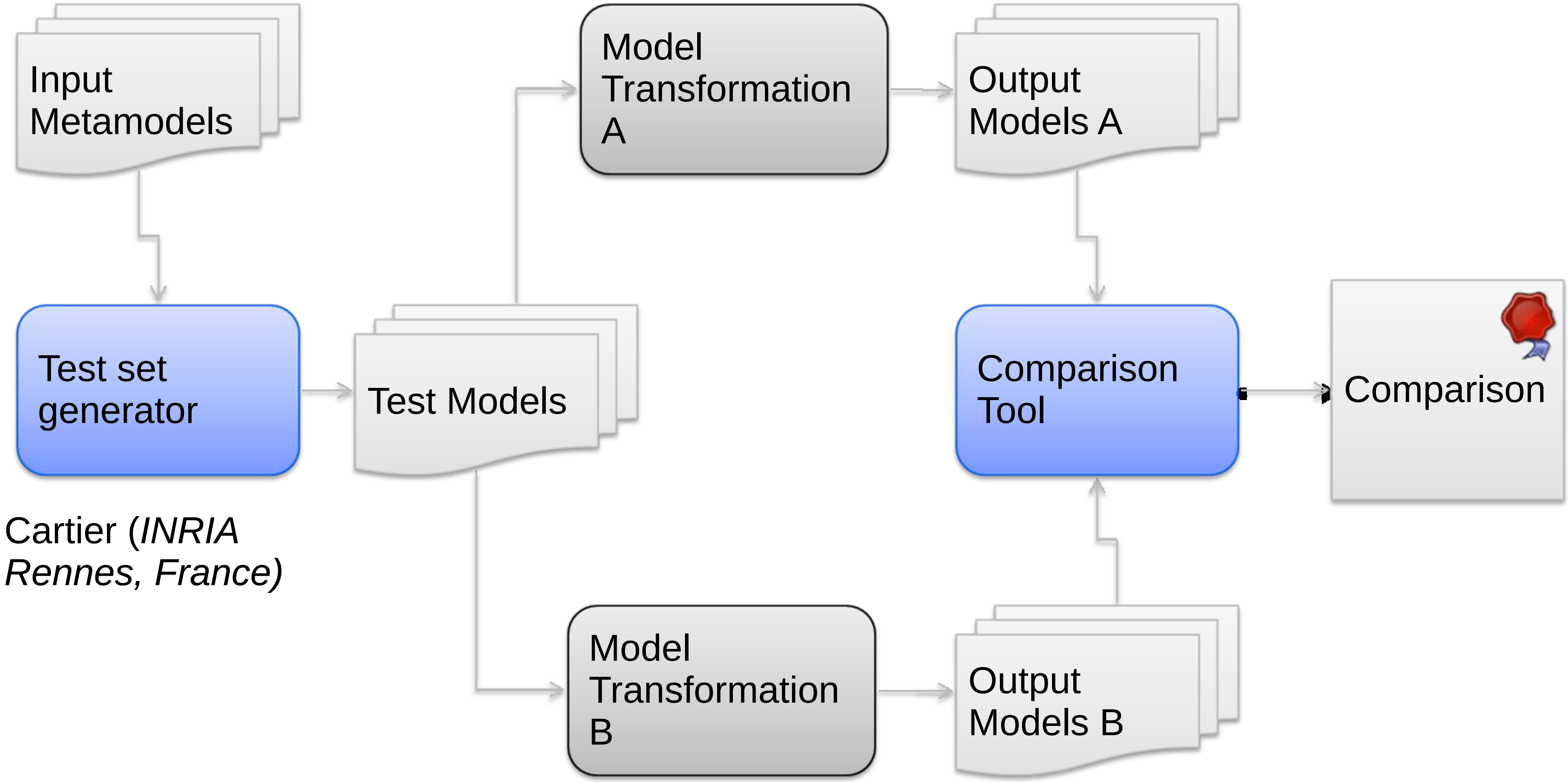
1) A cross-language comparison tool for model transformations
2) A mutation analysis framework for model transformations
3) Language independent vs language specific mutation operators for model transformations.
   - A language independent fault model for transformation languages.

Model transformation languages

This work considers three different model transformation frameworks:

- ATL *(ATLAS Group, France, 2002)*
  - Rule-based language + declarative OCL
  - Imperative user-defined "methods" *(helpers)*
  - It is possible to associate other imperative code to each rule (*do section*)
  - EMF metamodels
- Kermeta (*Triskell team, France, 2005)*
  - Imperative transformation language
  - EMOF metamodels
- AToM^3 *(MSDL, McGill University, 2002)*
  - Graph transformations (+ Python code)
  - ASG metamodels

1) Comparing transformations



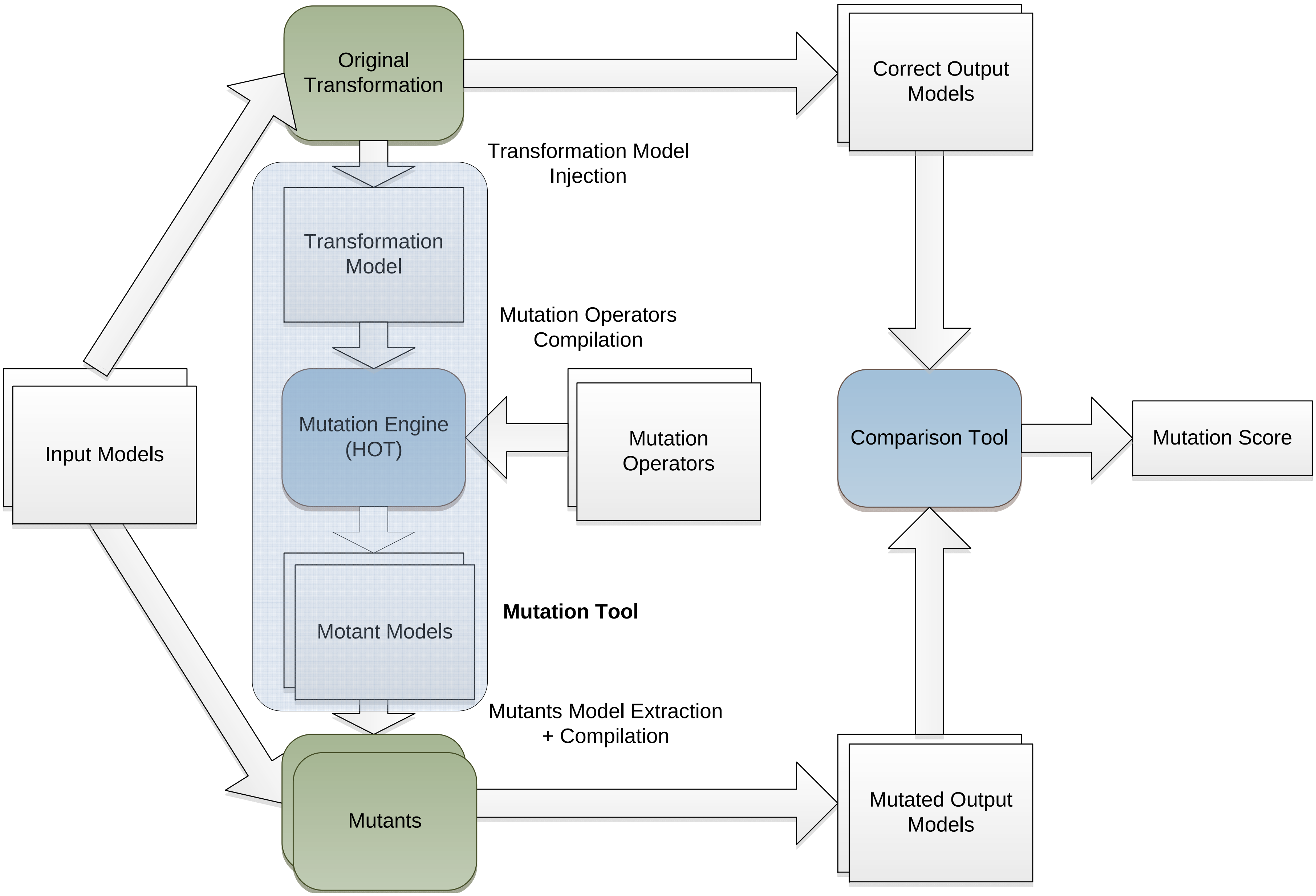
Comparing EMF Models

- A *yes/no* comparison of EMF models can't be performed at text level:
  - EMF models are XML trees where
    - the "physical" ordering of siblings in the EMF file is non-deterministic
    - cross-references between model elements are expressed by "path expressions" that are not unique and order-dependent
    - xmi:id depends on the implementation of the xmi serialization
- Our tool is based on EMF Compare (*Cédric Brun*)
  - Heuristic algorith for matching and difference (graph isomorphism!):
    - Each element of the source model is compared with the elements near the same location in the target model
    - For each comparison 4 strings (representing the elements name, content, type and relations) are calculated and compared.
    - A similarity index between 0 and 1 is computed.
    - If there is a clear winner the match is chosen.
    - The elements that are still not matched are searched in a bigger portion of the target file.

Checklist

- ATL support
  - Complete
- Kermeta support
  - Complete (with a small glitch :-<)
- AToM^3 support
  - Headless transformation execution
  - EMOF - AToM^3 export/import

2) Mutation analysis framework

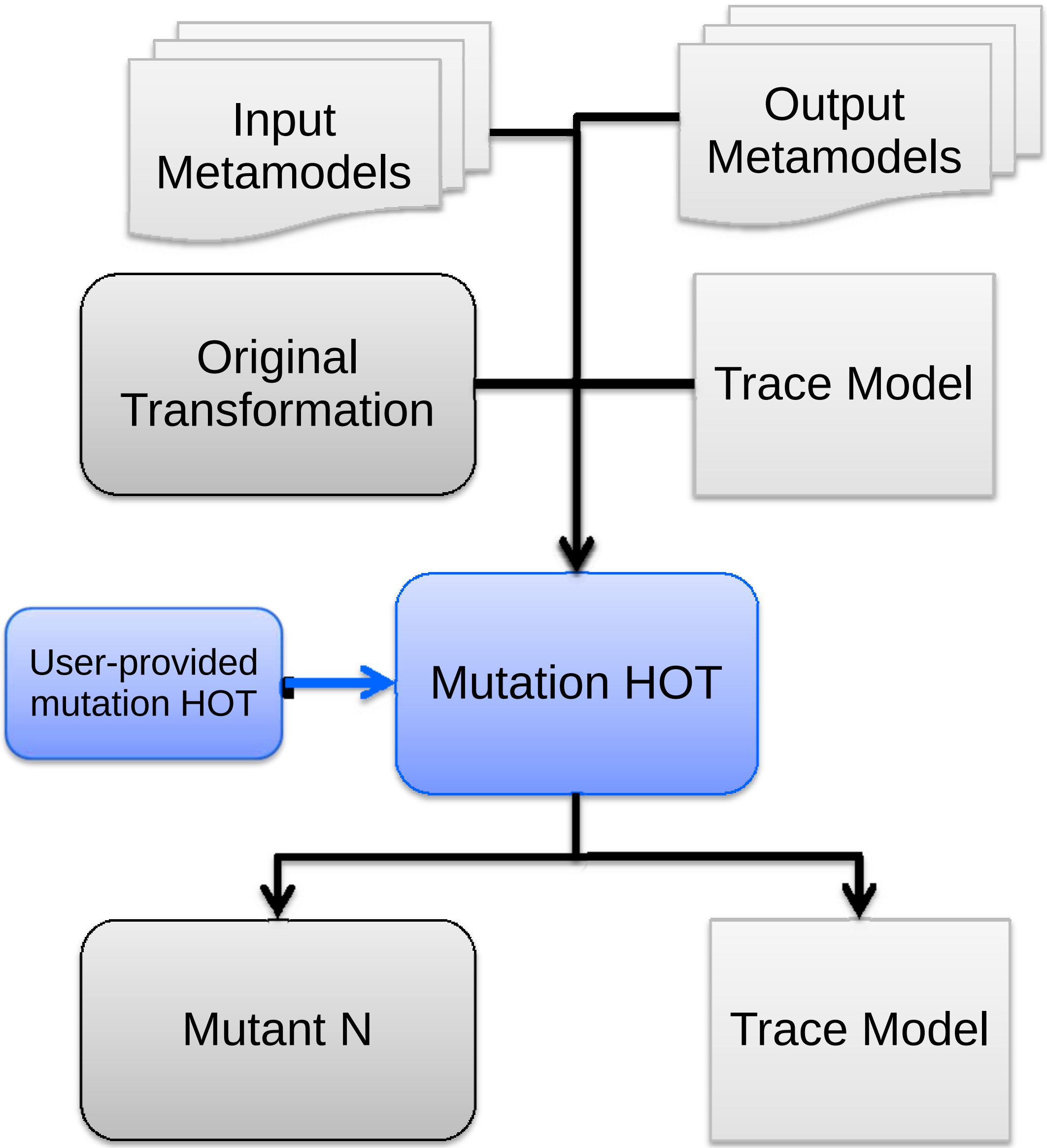


Higher Order Transformations for Mutations

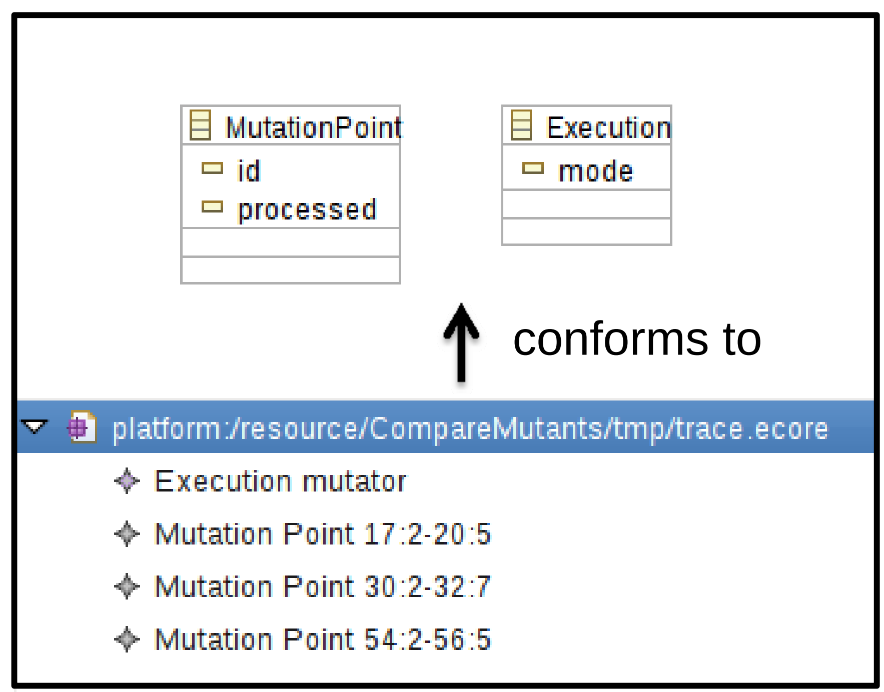
```
rule RemoveFilter {
    from
        m : ATL!InPattern (
            not m.filter.oclIsUndefined()
        )
    to
        pp : ATL!InPattern (
            elements <- m.elements,
            rule <- m.rule,
            location <- m.location,
            commentsAfter <- m.commentsAfter,
            commentsBefore <- m.commentsBefore
        )
}
```

- The user writes a declarative transformation for the mutation
  - Problem: the transformation engine would apply it to all the matches at once.
- Model driven solution:
  - Transparent for the user
  - Minimal computational cost
  - No change to the standard transformation engine
  - (HHOT, H^2OT, SuperHOT ?!?)

Higher Order Transformations for Mutations



Trace Model



Higher Order Transformations for Mutations

Two rules are generated from the user provided HOT.
Helpers control the execution (with constant cost)

1. A negative matching rule

```
rule RemoveFilter {
    from
        m : ATL!InPattern (
            not m.filter.oclIsUndefined()
        )
    to
        pp : ATL!InPattern (
            elements <- m.elements,
            rule <- m.rule,
            location <- m.location,
            commentsAfter <- m.commentsAfter,
            commentsBefore <- m.commentsBefore
        )
}
```

```
rule NotRemoveFilter {
    from
        m : ATL!InPattern (
            thisModule.isNotNextMatch(m.location)
            and ( not m.filter.oclIsUndefined() )
        )
    to
        m1 : ATL!InPattern (
            elements <- m.elements,
            rule <- m.rule,
            filter <- m.filter,
            location <- m.location,
            commentsAfter <- m.commentsAfter,
            commentsBefore <- m.commentsBefore
        )
    do {
        thisModule.notNextMatchingStep(m.location);
    }
}
```

Higher Order Transformations for Mutations

Two rules are generated from the user provided HOT.
Helpers control the execution (with constant cost)

1. A negative matching rule
2. A positive matching rule

```
rule RemoveFilter {
    from
        m : ATL!InPattern (
            not m.filter.oclIsUndefined()
        )
    to
        pp : ATL!InPattern (
            elements <- m.elements,
            rule <- m.rule,
            location <- m.location,
            commentsAfter <- m.commentsAfter,
            commentsBefore <- m.commentsBefore
        )
}
```

```
rule RemoveFilter {
    from
        m : ATL!InPattern (
            thisModule.isNextMatch(m.location)
        )
    to
        m1 : ATL!InPattern (
            elements <- m.elements,
            rule <- m.rule,
            location <- m.location,
            commentsAfter <- m.commentsAfter,
            commentsBefore <- m.commentsBefore
        )
    do {
        thisModule.nextMatchingStep(m.location);
    }
}
```

Checklist

- ATL support
  - ✔ Complete
- Kermeta support
  - ! Higher Order Transformations for Mutations
- AToM^3 support
  - ! Higher Order Transformations support (Rule Metamodel)

3) Cross-language mutation operators

- Future Work:
  - Compare mutation operators of transformation languages:
    - Do transformation languages have a common set of mutation operators ?
    - Is there a set of mutation operators for transformation languages that are inherently language specific ?
  - Howto:
    - Develop mutation operators for transformation languages
    - Compare the mutation scores relative to different mutation operators applied to the same test set
    - Identify equivalent mutation operators among different transformation languages

Thanks

- ATL: http://www.eclipse.org/m2m/atl/
- Kermeta: http://www.kermeta.org/
- AToM^3 : http://atom3.cs.mcgill.ca/
- EMF Compare: http://wiki.eclipse.org/index.php/EMF_Compare