An MDE-based method for bridging different design notations

Tian Zhang[1,2], Frédéric Jouault[3], Jean Bézivin[3], Xuandong Li[1,2]

[1] State Key Laboratory for Novel Software Technology

[2] Nanjing University

[3] AtlanMod team, INRIA and EMN

Outline

- Motivation
- Background on MDE tools
- MDE based method for bridge construction
- Case study
- Conclusion and future work

Motivation

- There are plenty of different design notations
  - Notations in support of *practical* approaches
    - UML
  - Notations in support of *rigorous* approaches
    - Formal methods
- Hence the problem of bridging these notations rises
- MDE is helpful in bridging heterogeneous platform

Background on MDE tools

- Model-driven Engineering (MDE)
  - The term MDE is typically used to describe software development approaches in which abstract models of software systems are created and systematically transformed to concrete implementations – Robort F, Rumpe B (2007)
- The AMMA platform
  - Kernel MetaMetaMode (KM3)
  - ATLAS Transformation Language (ATL)
  - Textual concrete syntax (TCS)
  - ATLAS Model Weaver (AMW)
  - ATLAS MegaModel Management (AM3)

The AMMA platform

- AMMA consists of two main sets of tools
  - One set is for modeling on a small scale
    - ATL, AMW, TCS
  - The other is for modeling on a large scale
    - AM3

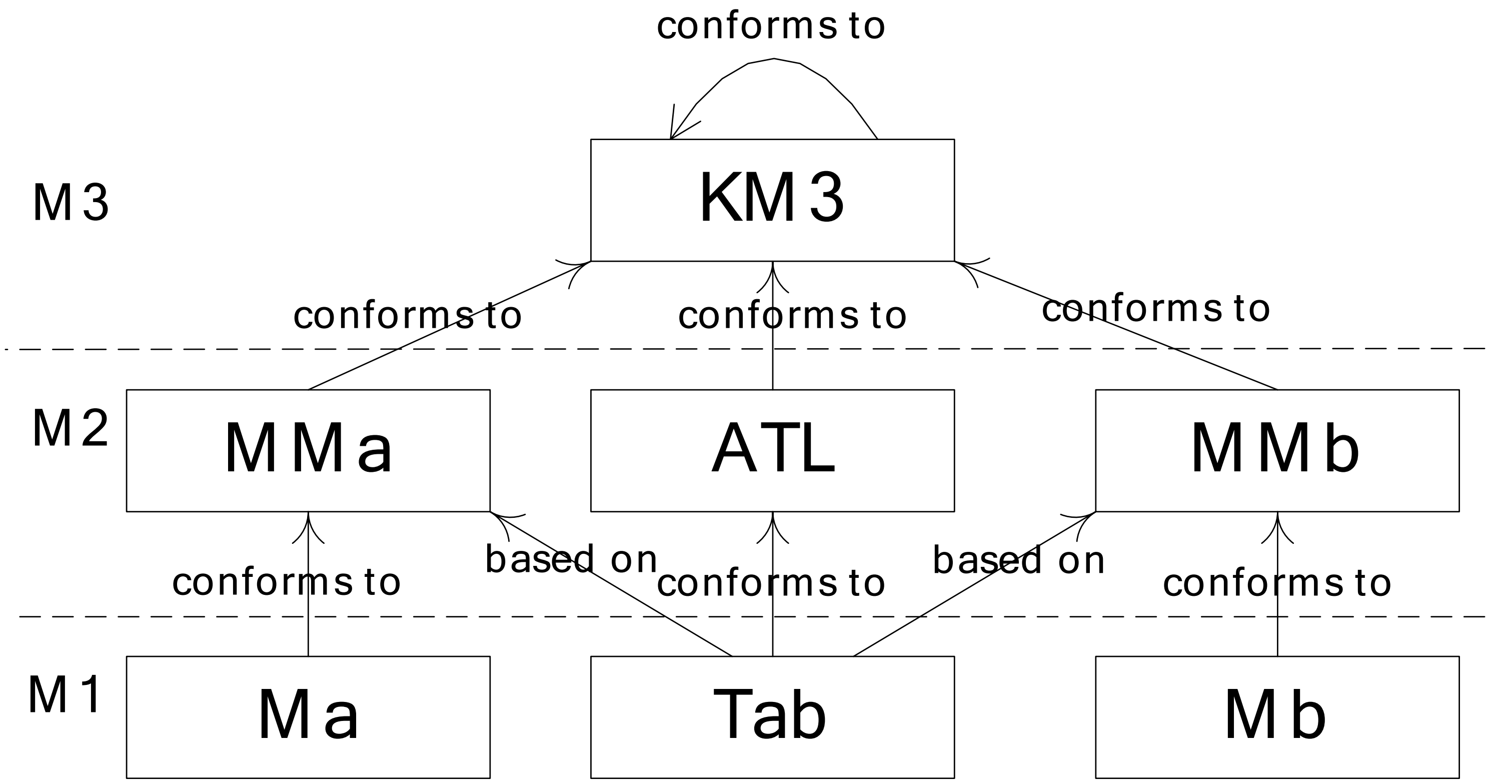
KM3

- The main purpose of KM3 is to define a root for meta–meta model hierarchy of the AMMA platform
- Another important purpose of KM3 is to define metamodels for DSLs
- KM3 is very similar to MOF but present more practical usability

ATL

- ATL is a hybrid of declarative and imperative constructs
  - ATL has been designed as an answer to the QVT RFP
  - The recommended style to write transformations is declarative
- ATL provides the basis upon which the other parts, such as AMW, TCS and AM3, are implemented

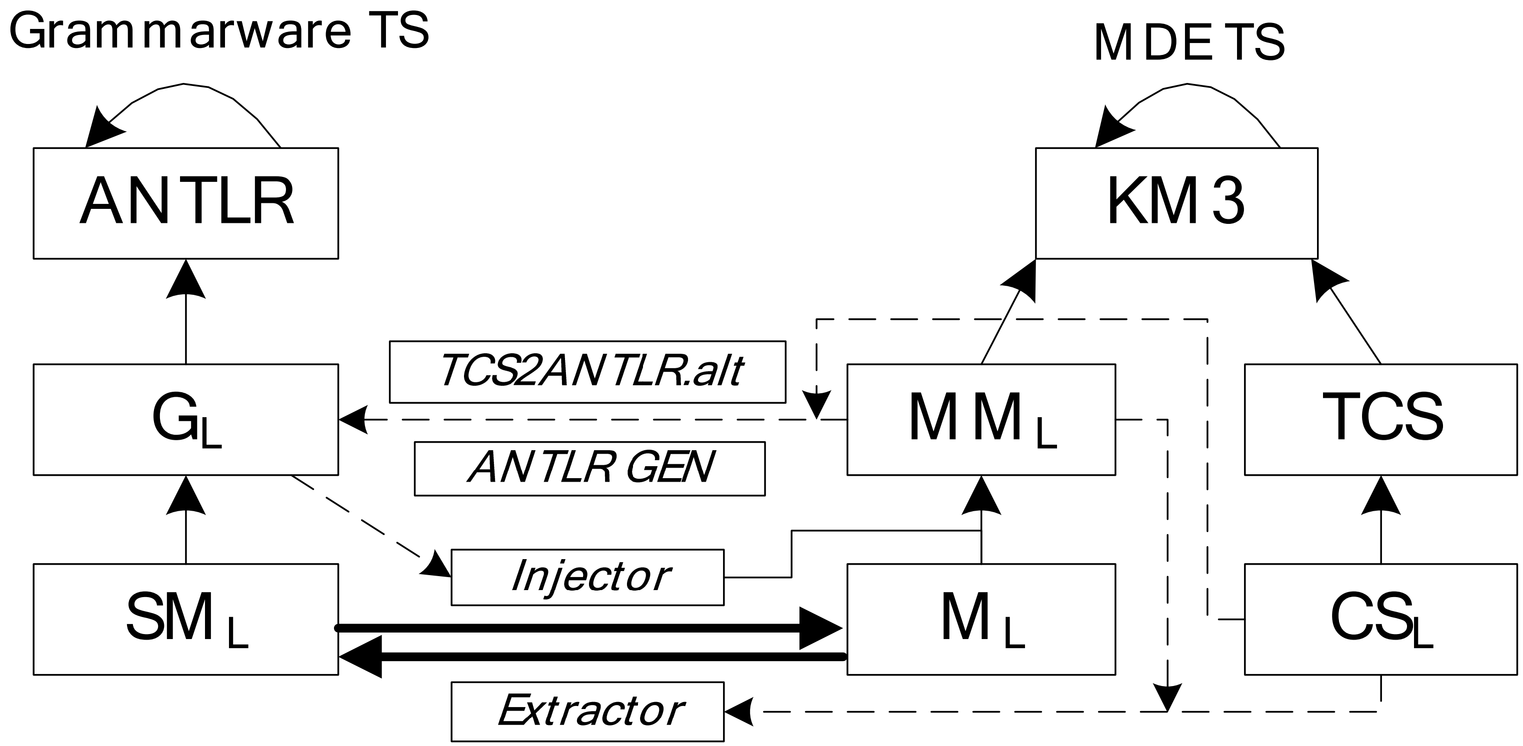
Overview of ATL usage



Textual concrete syntax

- TCS provides the capability of bridging Modelware and Grammarware
- TCS is implemented by reusing ANother Tool for Language Recognition (ANTLR)

Overview of TCS usage



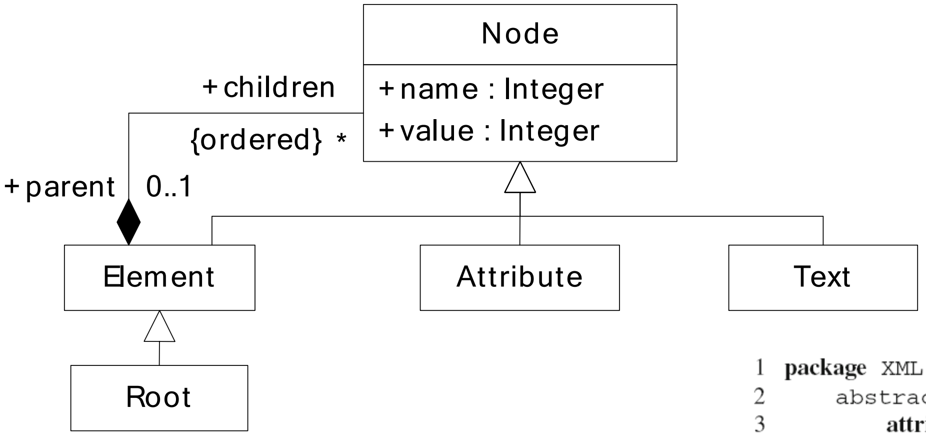
MDE-based approach for bridging different design notations

- Two main sub-problems:
  - Semantic mapping
  - Syntactical transformation
- How to solve these two problems based on AMMA
  - **Metamodeling**
  - **Semantic mapping**
  - **Syntactical transformation**

KM3 metamodeling

- Firstly, concepts of a notation are analyzed so as to abstract constructs as well as their corresponding structures
- Secondly, a metamodel is defined using KM3 notations

KM3 metamodeling for XML



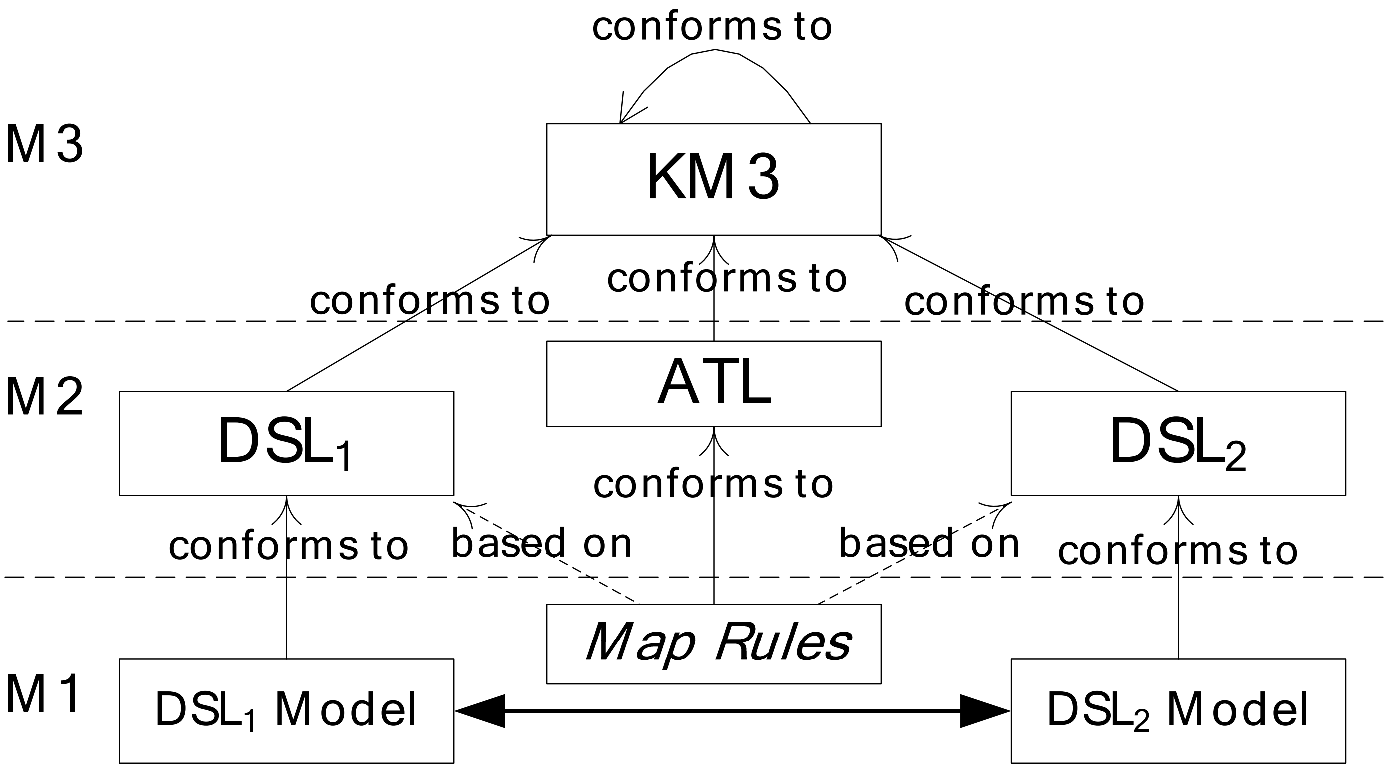

```
package XML {
    abstract class Node {
        attribute startLine[0-1] : Integer;
        attribute startColumn[0-1] : Integer;
        attribute endLine[0-1] : Integer;
        attribute endColumn[0-1] : Integer;
        attribute name : String;
        attribute value : String;
        reference parent[0-1] : Element oppositeOf children;
    }
    class Attribute extends Node {}
    class Text extends Node {}
    class Element extends Node {
        reference children[*] ordered container : Node oppositeOf parent;
    }
    class Root extends Element {}
}

package PrimitiveTypes {
    datatype String;
    datatype Integer;
    datatype Boolean;
}
```

Semantic mapping

- Semantic mapping relations can be defined as model transformation rules on corresponding metamodels
  - ATL transformation rules
    - Every single ATL rule supports the unidirectional mapping
    - The bidirectional mapping can be obtained by constructing ATL rules in both directions

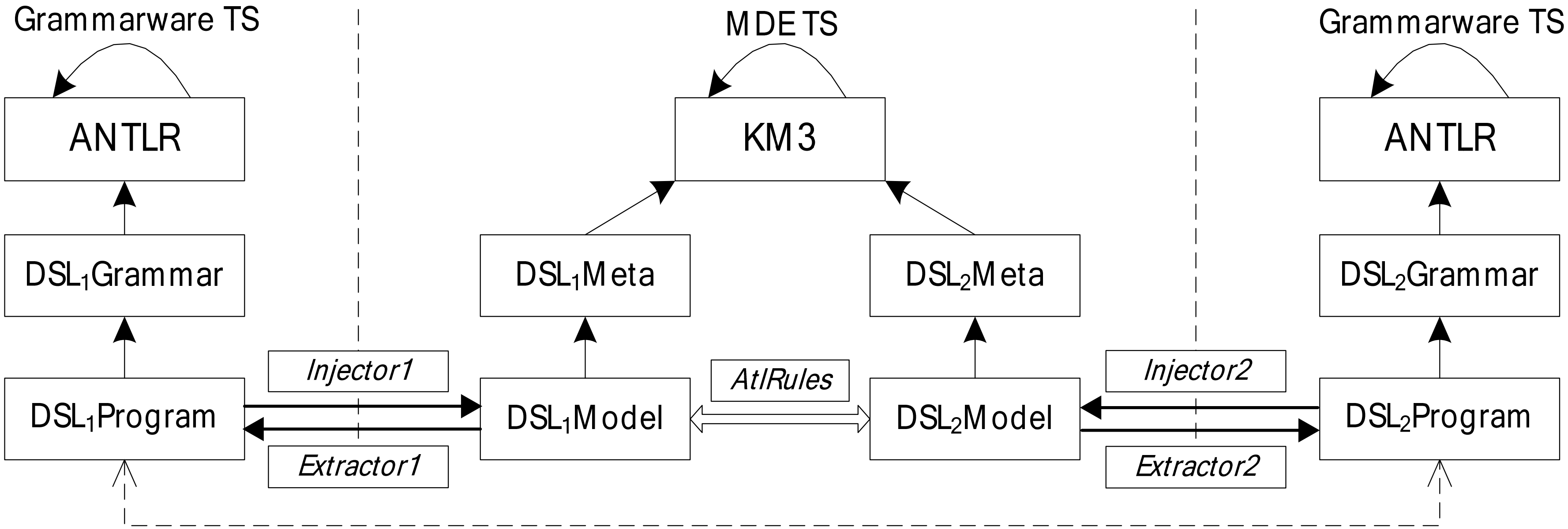
ATL based semantic mapping



Syntactical transformation

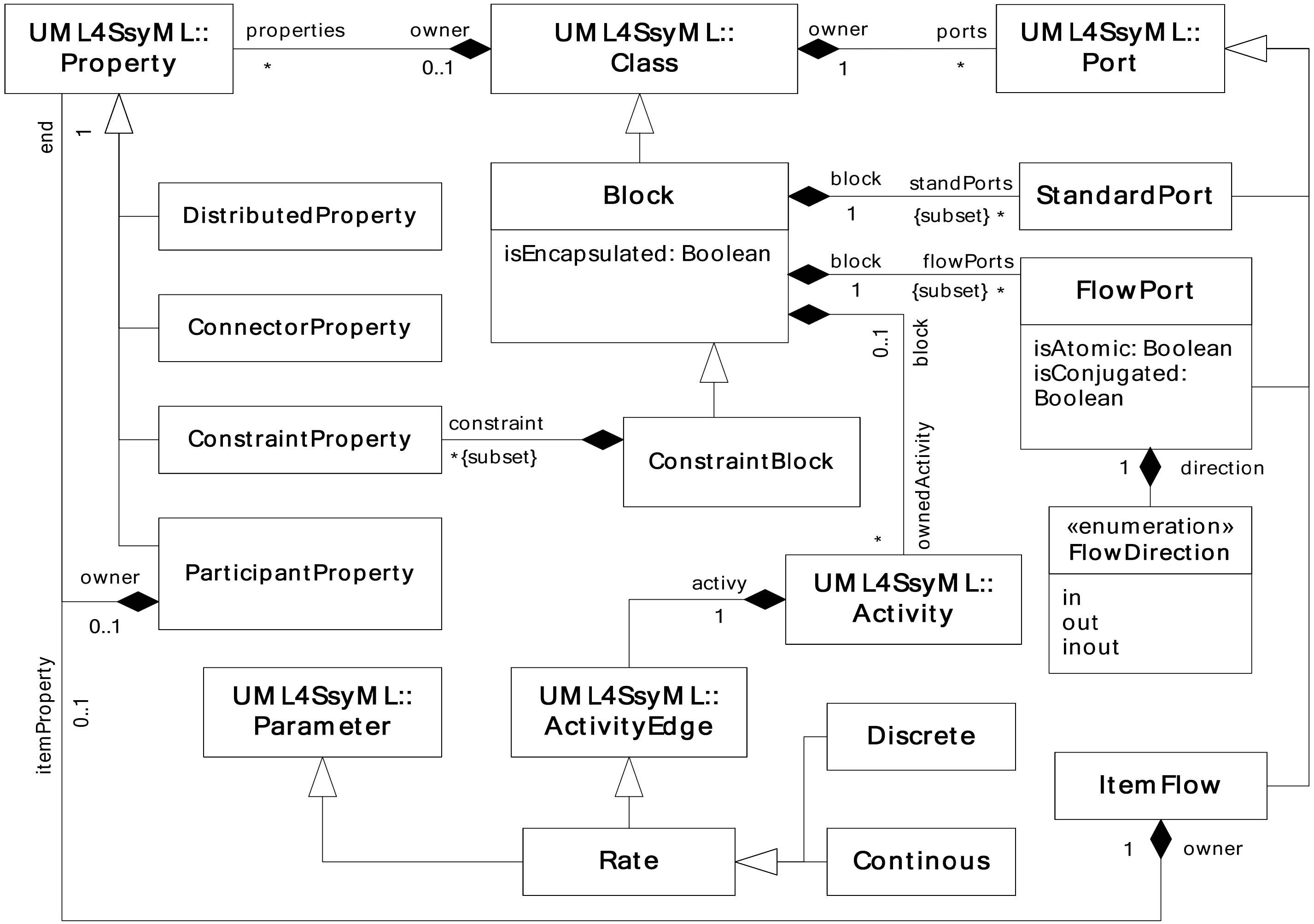
- Tow main field are considered
  - Modelware & Grammarware
- To transform graphical notations within Modelware
  - UML
    - To serialize the models into the XMI
- To transform the design notations between Modelware and Grammarware
  - Textual syntax rules are defined using TCS

TCS syntactical transformation

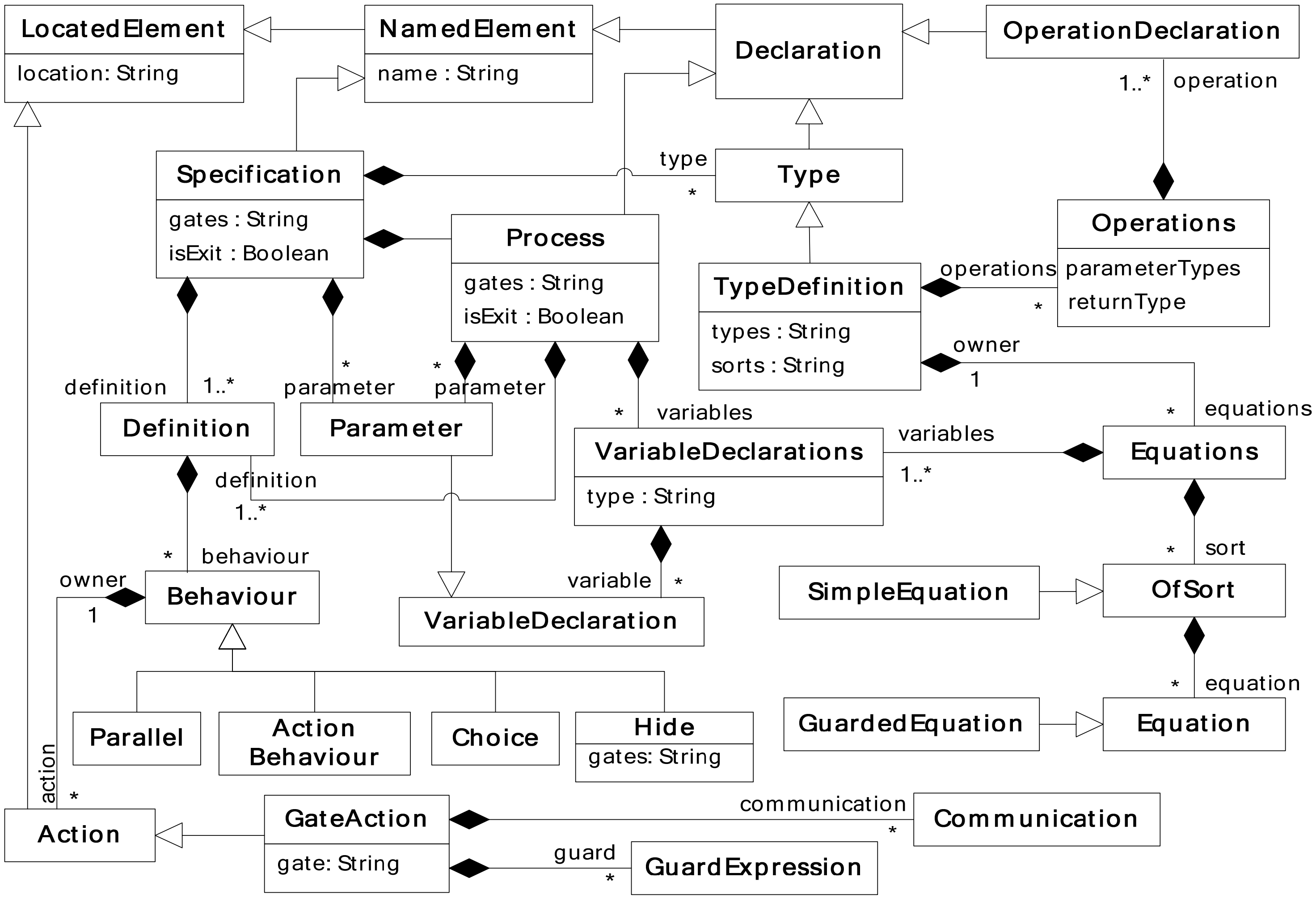


A case study: bridge SysML to LOTOS

The simplified metamodel of SysML



The simplified metamodel of LOTOS



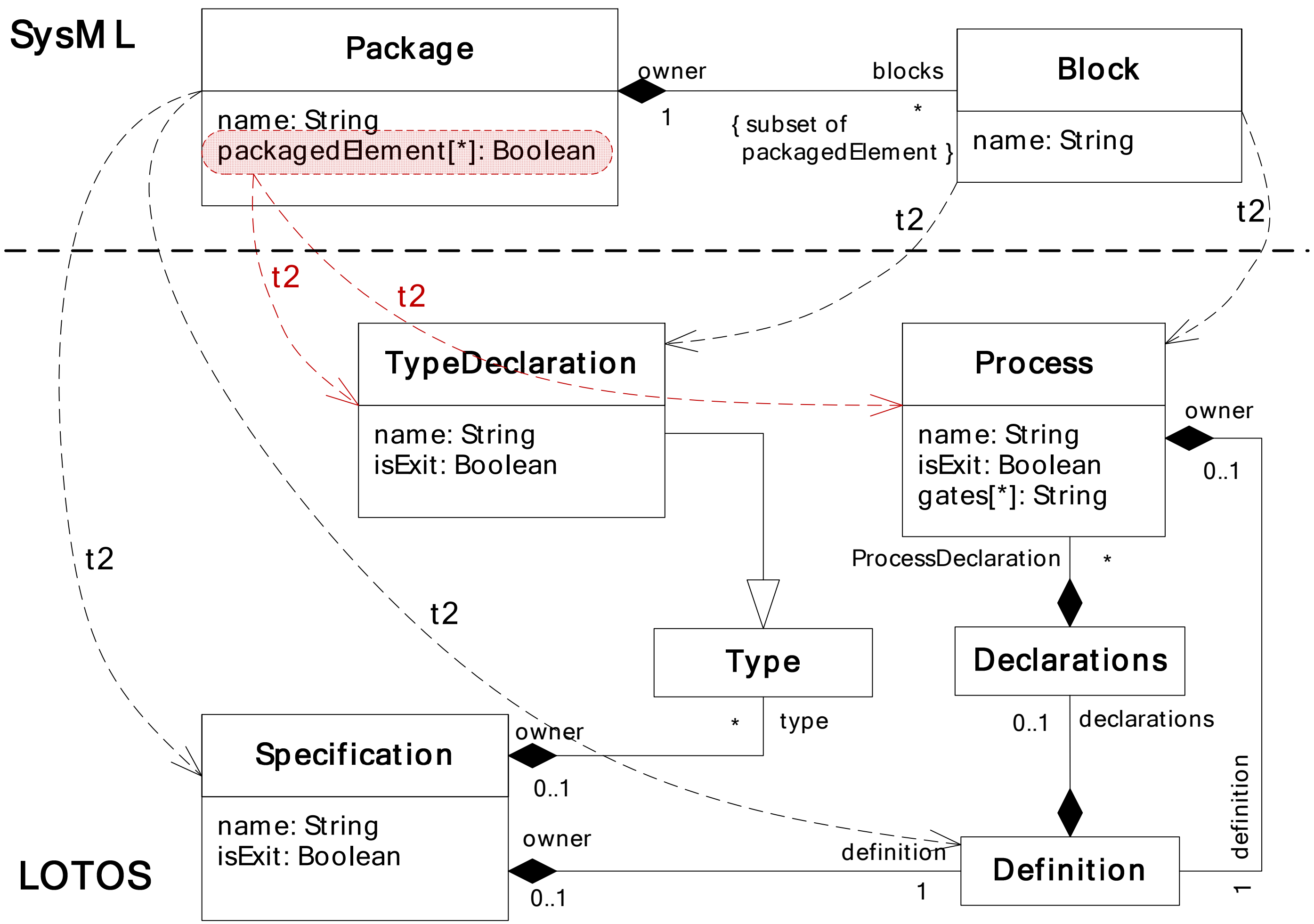
SysML-to-LOTOS bridge construction

- Assuming that *S* is the SysML metamodel and that *L* is the LOTOS metamodel:
  - a semantics for SysML can be provide by the LOTOS semantics contained in the metamodel generated through the use of $\Phi(S) = L$
  - the function Φ is defined through the definition of SysML-to-LOTOS mapping definitions (in short, S2Ls)

Structural constructs mapping

- S2L 1 Top-level package
- S2L 2 Top-level block
- S2L 3 Block-to-TypeDefinition *(ref. S2L 2)*
- S2L 4 Block-to-Process *(ref. S2L 2)*
- S2L 5 Nested block *(ref. S2L 2)*
- S2L 6 DistributedProperty
- S2L 7 ParticipantProperty *(ref. S2L 6)*

SysML package and owning blocks to LOTOS



The Package transformation rule

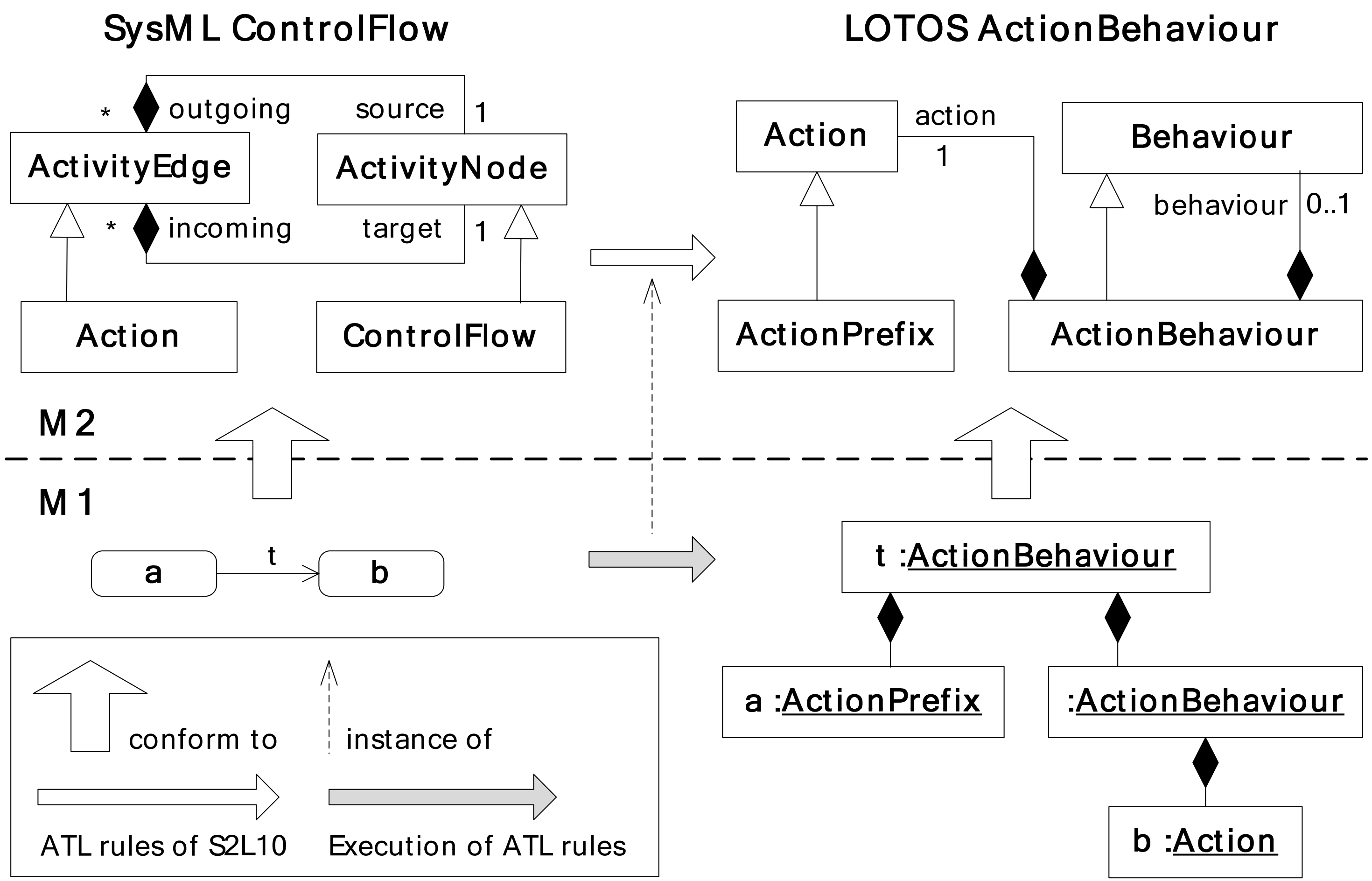
```
helper context SysML!Package def: blocks: Sequence(SysML!Block) =
    self.packagedElement -> select(e| e.oclIsTypeOf(SysML!Block));

rule Package {
    from
        p: SysML!Package(
            p.isTopLevelPackage
        )
    to
        s: LOTOS!Specification(
            name <- p.name + '_Spec',
            isExit <- true,
            types <- p.blocks -> collect(b| thisModule.resolveTemp(b, 'td'))
                ,
            definition <- deft
        ),
        deft: LOTOS!Definition(
            declarations <- p.blocks->collect(b| thisModule.resolveTemp(b,'p
                '))
        )
}
```

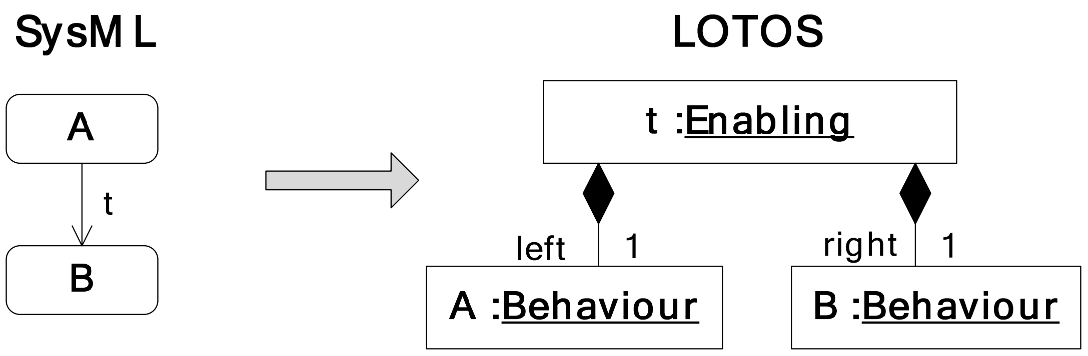
Behavioural constructs mapping

- S2L 8 Top-level activity
- S2L 9 Sub-activity
- S2L 10 Simple ControlFlow
- S2L 11 Common ControlFlow
- S2L 12 Recursive DecisionNode
- S2L 13 Branch DecisionNode
- S2L 14 Fork-join nodes
- S2L 15 Simple actions
- S2L 16 CallBehaviorAction

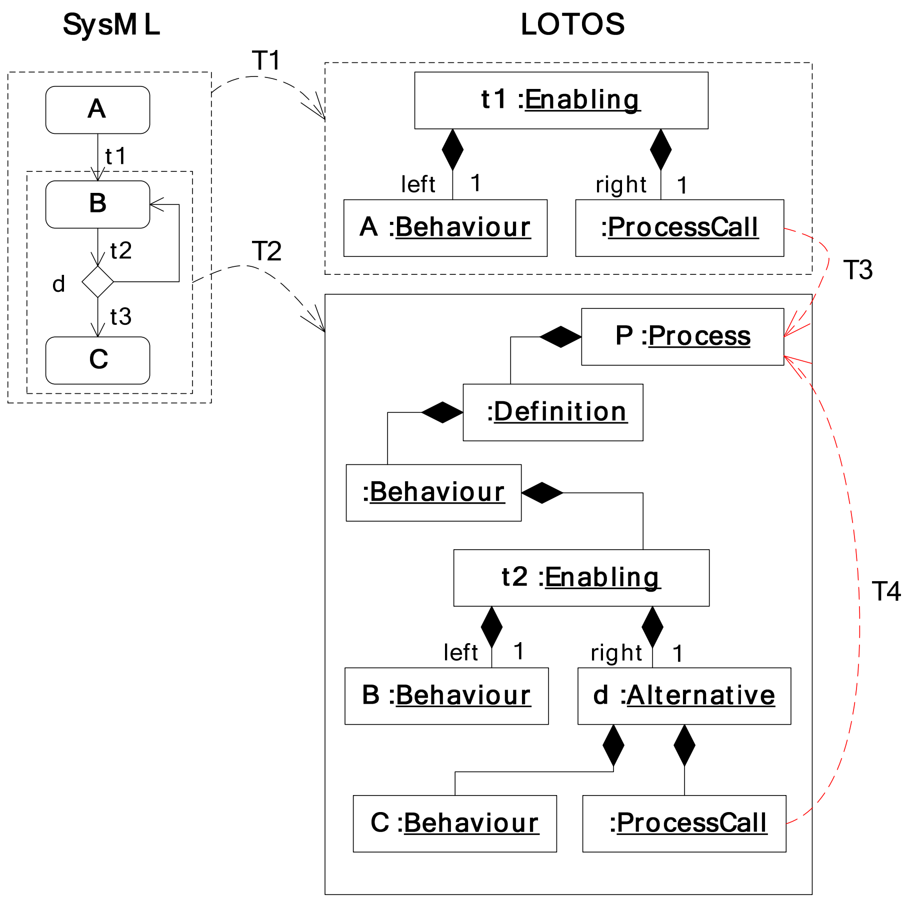
S2L10 Simple ControlFlow



A generic application of S2L11: *Common ControlFlow*



A generic application of S2L12: *Recursive DecisionNode*



Conclusion

- Based on the AMMA platform, our approach is practical and useful to design notations in both Modelware and Grammarware domains
  - More general usability
    - Both graphical and textual notations are covered
  - More reusable
    - Bridges can be registered in AM3 and published through the Internet so as to be searched and reused by different applications
    - ATL rules and TCS syntax can be reused in different bridge constructions

Future work

- Currently, semantic matching is discovered and built manually
- Future work will define the more convenient methods for semantic matching development between different notations

Thanks ！