Feature-Based Survey of Model Transformation Approaches

Krzysztof Czarnecki and Simon Helsen
in IBM Systems Journal, 2006

Mahsa Hasani Sadi
Department of Computer Science
University of Toronto
mhsadi@cs.toronto.edu

CS2125: Modeling Methods Tools and Techniques, Fall 2012
Instructor: Marsha Chechik
Background (1)


Research Question:
How to design a model transformation approach?
Outline of the Paper

- Specification of the features of model transformation approaches
- Classification of model transformation approaches

The outline of presentation:

Studying a set of the design concerns (features) of a model transformation language through examples

Krzysztof Czarnecki
ECE Department
University of Waterloo
Features of Model Transformation

- Specification
- Transformation Rules
- Rule Application Control
- Rule Organization
- Model Transformation
- Source – Target Relationship
- Incrementality
- Directionality
- Tracing
Example: Transforming Class Diagrams to Relational Tables (1)

(a) Simple UML metamodel

(b) Simple RDBMS metamodel
Example: Transforming Class Diagrams to Relational Tables (2)

(a) Sample UML model

Instances also matter!
transformation umlRdbms (uml : SimpleUML, rdbms : SimpleRDBMS) {
}

relation PackageToSchema /* map each package to a schema */ {

domain uml p:Package {name=pn}

domain rdbms s:Schema {name=pn}
}

Not Executable (Declarative Language)
**Specification Feature (2)**

```java
transformation Uml2Rdbms ( in uml:UML , out rdbms:RDBMS ) {

    // the entry point for the execution of the transformation

    main()

    {
        uml.objectsOfType(Package) -> map packageToSchema();
    }

    ....}
```

*Directly Executable (Imperative Language)*
Model Transformation Language Design Concerns

transformation umlRdbms (uml : SimpleUML, rdbms : SimpleRDBMS)
{
relation PackageToSchema /* map each package to a schema */ {
    domain uml p:Package {name=pn}
    domain rdbms s:Schema {name=pn}
}
}

transformation Uml2Rdbms (in uml:UML, out rdbms:RDBMS) {
    // the entry point for the execution of the transformation
    main()
    {
        uml.objectsOfType(Package) -> map packageToSchema();
    }
    ....
}
transformation Uml2Rdbms ( in uml:UML , out rdbms:RDBMS ) {
// the entry point for the execution of the transformation
main()
{
    uml.objectsOfType(Package)  ->  map packageToSchema();
}
....}
Transformation Rules Feature (1)

relation ClassToTable /* map each persistent class to a table */ {

  domain uml c:Class { namespace = p:Package {},
    kind='Persistent', name=cn  }

  domain rdbms t:Table { schema = s:Schema {}, name=cn}

  when { PackageToSchema(p, s); }
  where { AttributeToColumn(c, t); }
}

12 of 29
Transformation Rules Feature (2)

transformation umlRdbms (uml : SimpleUML, rdbms : SimpleRDBMS )

{ top relation PackageToSchema {…}

top relation ClassToTable {…}

relation AttributeToColumn {…}

}
Transformation Rules Design Concerns

relation PackageToSchema /* map each package to a schema */ {
}
enforce domain uml p:Package {name=pn}
checkonly domain rdbms s:Schema {name=pn}
}

relation PackageToSchema /* map each package to a schema */ {
}
enforce domain uml p:Package {name=pn}
enforce domain rdbms s:Schema {name=pn}
}
Source – Target Relationship Feature

- New Target
- Existing Target
  - Update
    - Destructive
    - Extension Only
  - In-Place
Tracing

relation ClassToTable {

enforce domain.uml c:Class {namespace=p:Package {}, kind='Persistent', name=cn}

enforce domain.rdbms t:Table {schema=s:Schema {}, name=cn}

}

class TClassToTable

{
    c: Class;
    p: Package;
    t: Table;
    s: Schema;
}

16 of 29
Incrementality Feature

Target-Incrementality

Source-Incrementality

Preservation of User Edits in Target
Pattern Matching Feature

Pattern = {
   e1: <classname1>, e2: <classname2> ... en: <classnameN>
L1: <assoc1> (ei, ej) ... Lm: <assocM> (eu, ew)
where <predicate>
}

<object>.<property> = <variable>
<object>.<property> = <expression>
library MyUmlFacilities(UML);

......

transformation Uml2Rdbms (in uml:UML, out rdbms:RDBMS)

extends BasicUml2Rdbms, -- extending a transformation

extends library UMLUtilities(UML) -- extending a library

access library MathLibrary; -- accessing a math library

......
Parallel Transformations

```java
var tr1 := new Req2Pimgui(req, pimGui);

var tr2 := new Req2Pimbehavior(req, pimBehavior);

var st1 := tr1.parallelTransform(); // forks the PIM GUI transformation

var st2 := tr2.parallelTransform(); // forks the PIM Behavior transformation

this.wait(Set{st1, st2}); // waits patiently

if (st1.succeeded() and st2.succeeded()) // creates the executable model
    new Pim2Psm(pimGui, pimBehavior, psm).transform();
```
Rule Application Control Feature

- Rule Application Control
  - Scheduling
    - Rule-Selection
      - Non-Determinism
      - Explicit Condition
      - Conflict Resolution
      - Interactive
    - Phasing
Categorization of Model Transformation Approaches

- Model to Text
- Model to Model

```java
public class Customer {
    private String name;
    private Address addr;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public void setAddress(String name) {
        this.addr = addr;
    }

    public String getAddress() {
        return this.addr;
    }
}
```

(a) Simple UML metamodel

(b) Simple RDBMS metamodel
Model to Text Approaches

- Visitor-Based
- Template-Based

```java
<<DEFINE Root FOR Class>>
public class <<name>> {
    <<FOREACH attrs AS a>>
        private <<a.type.name>> <<a.name>>;
    <<ENDFOREACH>>
    <<EXPAND AccessorMethods FOREACH attribute>>
}
<<ENDDEFINE>>

<<DEFINE AccessorMethods FOR Attribute>>
public <<type.name>> get<<name.toFirstUpper>>() () {
    return this.<<<name>>;
}
public void set<<name.toFirstUpper>>( <<type.name>> <<name>> ) {
    this.<<<name>> = <<name>>
}
<<ENDDEFINE>>
```
Model-to-Model Approaches

- Direct Manipulation
- Structure-Driven
- Operational
- Relational
- Template-Based
- Graph-Transformation-Based
- Hybrid
Cited by 536 (publication year: 2006)

Revised version of another paper of the same authors, “Classification of Model Transformation Approaches”, in OOPSLA 2003. Cited by 732

Evaluation of the paper

- A Well-written journal paper
- A journal paper
- Novelty
- Significance of contributions

Cited by 536

28 of 29
Discussion

- Why a separate language for model-transformation?

- What types of model transformations does the language support? (Does it make any difference?)
  - Structural model to structural model?
    - Example: class to table?
  - Behavioral model to behavioral model?
    - Example: state diagram to petri nets?
  - Structural model to behavioral model / vice versa?