

# Analyzing the Quality of Model Transformations

Marcel van Amstel

SET Seminar (2IS95) September 25<sup>th</sup>, 2008

# Presentation Outline

- 1 Background
- 2 Goal
- 3 Approach
- 4 Future Work
- 5 Literature

# Background

## Model Driven Engineering

### UML class diagram

A
int a
int getA()
void setA(int arg)

### Java code

```
class A{  
    private int a;  
    public int  getA() { return a; }  
    public void setA(int arg) { a = arg; }  
}
```

### Machine code

```
mov eax, 1  
mov ebx, 10  
loop: inc eax  
      cmp eax, ebx  
      jle loop
```

Level  
of  
Abstraction



# Background

## Model Driven Engineering

- Model: abstraction with a specific purpose.
- Domain specific languages for modeling.
- Model transformations for synthesis.

# Background

## Model Driven Engineering

- Model: abstraction with a specific purpose.
- Domain specific languages for modeling.
- Model transformations for synthesis.

# Background

## Model Driven Engineering

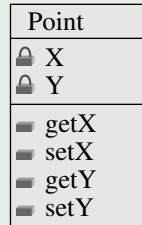
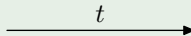
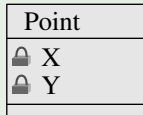
- Model: abstraction with a specific purpose.
- Domain specific languages for modeling.
- Model transformations for synthesis.

# Background

## Model Transformation

### Example

Adding accessors to a class diagram



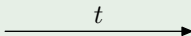
# Background

## Model Transformation

### Example

#### Generating (Java) code

Point
🔒 X
🔒 Y
🔑 getX
🔑 setX
🔑 getY
🔑 setY



```
public class Point {  
  
    private int X;  
    private int Y;  
  
    public int GetX(){  
        return X;  
    }  
    public void SetX(int val){  
        X = val;  
    }  
  
    public int GetY(){  
        return Y;  
    }  
    public void SetY(int val){  
        Y = val;  
    }  
}
```

# Background

## Model Transformation Formalisms

- ATL
- QVT
- openArchitectureWare
- ASF+SDF
- TOM
- Stratego/XT
- ...

# Background

## Model Transformation Formalisms

- ATL
- QVT
- openArchitectureWare
- ASF+SDF
- TOM
- Stratego/XT
- ...

# Background

## ASF+SDF

- Specification of (domain specific) languages in SDF.
- Specification of conditional rewrite rules in ASF.
- Transformations are syntax-safe.

# Background

## ASF+SDF

### Example

#### Syntax definition (language)

context-free syntax

```
<Name, List[[Attribute]], List[[Operation]]> -> Class
```

```
String -> Name
```

```
String -> Attribute
```

```
String -> Operation
```

# Background

## ASF+SDF

### Example

#### Syntax definition (transformation)

context-free syntax

```
addGettersAndSettersToClass(Class) -> Class
```

```
createGettersAndSetters(List[[Attribute]]) -> List[[Operation]]
```

```
createGetter(Attribute) -> Operation
```

```
createSetter(Attribute) -> Operation
```

# Background

## ASF+SDF

### Example

#### Transformation definition

equations

```
[addGettersAndSettersToClass-1]
```

```
<$Name, $AttributeList, $OperationList> := $Class,
```

```
$OperationList1 := createGettersAndSetters($AttributeList),
```

```
$OperationList' := concat($OperationList1, $OperationList),
```

```
$Class' := <$Name, $AttributeList, $OperationList'>
```

```
====>
```

```
addGettersAndSettersToClass($Class) = $Class'
```

# Background

## ASF+SDF

### Example

### Term definition

The screenshot displays the ASF+SDF Meta-Environment IDE. The main window is titled "ASF+SDF Meta-Environment" and contains a menu bar with "File", "Views", "Facts", "Module", "Term", "Structure", and "Edit". The "Process" view on the left shows a tree structure with "AddGetterSetter", "Class", "basic", and "containers". The main editor window, titled "import-graph:'sdf'", shows the term definition:

```
addGettersAndSettersToClass[<"Point", ["X", "Y"], []>]
```

The bottom panel contains a "Console" view with a table for "Name...", "Key", and "Value", and a "Progress" view showing two messages: "constructor used more than once". The status bar at the bottom indicates "Idle".

# Background

## ASF+SDF

### Example

### Transformation result

The screenshot displays the ASF+SDF Meta-Environment interface. The main window is titled "ASF+SDF Meta-Environment" and contains a menu bar with "File", "Views", "Facts", "Module", "Term", "Structure", and "Edit". Below the menu bar is a "Process" view showing a tree structure with "AddGetterSetter" (Class), "basic" (folder), and "containers" (folder). The main editor area shows the transformation result: `<"Point", ["X", "Y"], ["GetX" , "SetX" , "GetY" , "SetY"]>`. At the bottom, there is a "Console" view with a table for "Name...", "Key", and "Value", and a "Progress" view showing two messages: "constructor used more than once". The status bar at the bottom left indicates "Idle".

Name...	Key	Value
---------	-----	-------

- constructor used more than once
- constructor used more than once

# Goal

## Problem Statement

- Model driven engineering is becoming increasingly important.
- Model transformations are similar to traditional software artifacts.

# Goal

## Research Goal

- Make the quality of model transformations measurable.

Model transformation

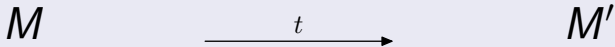


# Goal

## Research Goal

- Make the quality of model transformations measurable.

### Model transformation



# Approach

## Quality Attributes



# Approach

## Metrics

Four categories:

- Size metrics
- Function metrics
- Module metrics
- (In)consistency metrics

# Future Work

- Extend the set of metrics and quality attributes.
- Create a tool to enable automatic metrics extraction. (Almost done)
- Create a quality model.
- Apply techniques to other transformation formalisms.
- Propose a methodology for creating high-quality model transformations.

# Possible MSc Assignments

Apply similar techniques to:

- ATL
- openArchitectureWare
- ...

## Activities

- Getting familiar with model transformations and the formalism.
- Establishing a set of metrics that enable quality measurement.
- Implementing a tool that enables automatic calculation of these metrics.
- Performing case studies to validate the results.

# Possible MSc Assignments

Apply similar techniques to:

- ATL
- openArchitectureWare
- ...

## Activities

- Getting familiar with model transformations and the formalism.
- Establishing a set of metrics that enable quality measurement.
- Implementing a tool that enables automatic calculation of these metrics.
- Performing case studies to validate the results.

# Literature



Douglas C. Schmidt.

Model-driven engineering.

*Computer*, 39(2):25–31, February 2006.



M.F. van Amstel, C.F.J. Lange, and M.G.J. van den Brand.

Metrics for analyzing the quality of model transformations.

In *Proceedings of the 12th ECOOP Workshop on Quantitative Approaches on Object Oriented Software Engineering (QAOOSE08)*, pages 41–51, Paphos, Cyprus, July 2008.