Designing a Domain-Specific Language for Model Consistency

Proposal for the Master’s Thesis of Dominik Werle | October 12, 2015

1st Advisor: Dipl.-Inform. Max E. Kramer
2nd Advisor: M.Sc. Michael Langhammer
Motivation: VITRUVIUS

public class comp_1
    implements A1 {
  @Override
  public void methodFromA1 {
    doSomething();
  }
}

(Kramer, Burger, and Langhammer 2013)
Motivation: VITRUVIUS

(component diagram)

```
public class comp_1
    implements A1 {
  @Override
  public void methodFromA1 {
    doSomething();
  }
}
```

sync. overlap when one model changes?

(Kramer, Burger, and Langhammer 2013)
Motivation: VITRUVIUS

(Kramer, Burger, and Langhammer 2013)
Motivation: VITRUVIUS

public class comp_1 implements A1 {
  @Override
  public void methodFromA1 {
    doSomething();
  }
}

(Kramer, Burger, and Langhammer 2013)
PIBA

Problem: GPL $\Rightarrow$ similar consistency problems are solved repeatedly.

Idea: Design DSL for consistency rules. Generate transformations.

Benefit: Separate consistency rules from generic model transformation code.

Actions:
- Case studies $\Rightarrow$ identify recurring patterns.
- Evaluate how code can be generated.
- Extend DSL editor and code generation.
Example (DSL sketch)

```plaintext
map m1.Function and m2.Block {
    with port[FunctionPort] and inPorts[InPort] {
        when { port.dir = in } }
```

(Kramer 2015)
Example (DSL sketch)

```rust
map mm1.Function and mm2.Block {
    with port[FunctionPort] and inPorts[InPort] {
        when { port.dir = in } }
}```

(Kramer 2015)
Example (DSL sketch)

```plaintext
map mm1.Function and mm2.Block {
    with port(FunctionPort) and inPorts(InPort) {
        when { port.dir = in } ... }
```

(Kramer 2015)
Example (DSL sketch)

```plaintext
map mm1.Function and mm2.Block {
  with port[FunctionPort] and inPorts[InPort] {
    when { port.dir = in } } ... }
```

(Kramer 2015)
Example (DSL sketch)

```java
map mm1.Function and mm2.Block {
    with port[FunctionPort] and inPorts[InPort] {
        when { port.dir = in } } ... }
```

f1:Function corresponds b1:Block

fp:FunPort

dir =

ipb1:InPort

(Kramer 2015)
Example (DSL sketch)

map `mm1.Function` and `mm2.Block` {
    with `port[FunctionPort]` and `inPorts[InPort]` {
        when { `port.dir = in` } ... }

(f1:Function corresponds b1:Block)

(fp:FunPort dir = in)

(ipb1:InPort)

(Kramer 2015)
Example (DSL sketch)

```plaintext
map mm1.Function and mm2.Block {
    with port[FunctionPort] and inPorts[InPort] {
        when { port.dir = in } } ... }
```

- `f1`: Function
- `b1`: Block
- `fp`: FunPort
- `ipb1`: InPort

(Kramer 2015)
Example (DSL sketch)

```plaintext
map mm1.Function and mm2.Block {
    with port[FunctionPort] and inPorts[InPort] {
        when { port.dir = in } } ...
}
```

(Kramer 2015)
BX + CDT: $t_{AB}^{-1}(\Delta_A) = \Delta_B$; $\text{apply}(B, \Delta_B) = B'$

- Granularity of $\Delta$? / States in which synchronization happens?
- Input needed for consistency? Transformation state? A and/or B?
Motivation PIBA Example Foundations Conception Related Work Timeline Conclusion

Dominik Werle – A DSL for Model Consistency

October 12, 2015 5/9

BX + CDT: $t_{AB}^{-1}(\Delta_A) = \Delta B$; apply$(B, \Delta B) = B'$

- Granularity of $\Delta$? / States in which synchronization happens?
- Input needed for consistency? Transformation state? $A$ and/or $B$?
BX + CDT: $t_{AB}^{-1}(\Delta_A) = \Delta_B$; $apply(B, \Delta_B) = B'$

- Granularity of $\Delta$? / States in which synchronization happens?
- *Input* needed for consistency? *Transformation state*? $A$ and/or $B$?

BX + CDT: $t_{AB}^\leftarrow(\Delta_A) = \Delta_B$; $\text{apply}(B, \Delta B) = B'$

- Granularity of $\Delta$? / States in which synchronization happens?
- Input needed for consistency? Transformation state? A and/or B?

BX + CDT: $t_{AB}^\leftarrow(\Delta_A) = \Delta B$; apply$(B, \Delta B) = B'$

- Granularity of $\Delta$? / States in which synchronization happens?
- Input needed for consistency? Transformation state? $A$ and/or $B$?
**BX + CDT:** \( t_{AB}^{-1}(\Delta_A) = \Delta_B; \ apply(B, \Delta_B) = B' \)

- **Granularity of \( \Delta \)? / States in which synchronization happens?**
- **Input needed for consistency? Transformation state? A and/or B?**

BX + CDT: $t_{AB}^{-1}(\Delta_A) = \Delta B$; apply($B, \Delta B$) = $B'$

- Granularity of $\Delta$? / States in which synchronization happens?
- Input needed for consistency? Transformation state? $A$ and/or $B$?
Conception

- DSL: MIR (mappings, invariants, responses).

- Casestudies: PCM–JaMoPP, Energy Domain, Automotive Domain
Related Work

- Transformation Languages: TGGs, QVT(-R)
- Formal: Bidirectionality (Stevens 2008), Lenses (Foster et al. 2007)

Altoggether more abstract level
### Timeline

<table>
<thead>
<tr>
<th>2015</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct</td>
<td>2015</td>
</tr>
<tr>
<td>Nov</td>
<td>2016</td>
</tr>
<tr>
<td>Dec</td>
<td>2016</td>
</tr>
<tr>
<td>Jan</td>
<td>2016</td>
</tr>
<tr>
<td>Feb</td>
<td>2016</td>
</tr>
<tr>
<td>Mar</td>
<td>2016</td>
</tr>
<tr>
<td>A</td>
<td>2016</td>
</tr>
</tbody>
</table>

#### Writing
- Related Work
- Case Studies
- Feature Iterations
  - F1
  - F2
  - F3
  - F4
  - F5
  - ... (Buffer)

For each feature:
- Identify pattern
- If feasible: code to generate, language feature, generator

---

**Dominik Werle – A DSL for Model Consistency**

October 12, 2015
Problem: GPL ⇒ similar consistency problems are solved repeatedly.

Idea: Design DSL for consistency rules. Generate transformations.

Benefit: Separate consistency rules from generic model transformation code.

Actions:
- Case studies ⇒ identify recurring patterns.
- Evaluate how code can be generated.
- Extend DSL editor and code generation.

Motivation

PIBA

Example

Foundations

Conception

Related Work

Timeline

Conclusion
Foundations

- Metamodels
- Mappings
- Invariants
- Responses

Editor (Xtext)

Generator

Bidirectional Java

Based on

Constrain

Restore

Invariants specify invariants

Constrains responses

Maintains instances

Triggers changes

Parameterizes transformations

Generates

Correspondences

Instantiates

Uses

View type

View

Kramer, Burger, and Langhammer 2013

Dominik Werle – A DSL for Model Consistency

October 12, 2015
Risks

- dynamic scaling of workload based on findings during thesis
- unexpected technical problems with EMF or Xtext ⇒ forum, group. Test in parts of models/meta models instead of original system.
- invasiveness of implementation (regarding VITRUVIUS, MIR frameworks)
Limitations & Future Work

- Empirical study for evaluation ⇒ maintainability, usability
- Formal reasoning, verification
- Implement all identified features
- Test on complete meta models, models.


