Leveraging Executable Language Engineering for Domain-Specific Transformation Languages (Position Paper)
EXE 2016, Saint-Malo, France

Erwan Bousse ¹  Manuel Wimmer ¹  Wieland Schwinger ²
Elisabeth Kapsammer ²

¹TU Wien, Austria
²JKU Linz, Austria

October 3, 2016
Observations

- **Domain Specific Transformation Language (DSTL)** = model transformation language tailored for specific tasks (eg. strings renaming, code generation)

- DSTLs more and more common:
  - Two papers on DSTLs at ICMT’16 in a dedicated “Model Transformation Languages” session
  - This year TTC’16 use case: data-flow based DSTL
  - Increasing need for methods to develop DSTLs

- Progress in executable Domain-Specific Modeling language (xDsML) engineering:
  - Generic **syntactic** services (eg. editors using Xtext or Sirius)
  - Generic **runtime** services (eg. debugger using GEMOC studio)
  - Easier and easier to obtain a tool-supported xDSML
Observations

- **Domain Specific Transformation Language (DSTL)** =
  model transformation language tailored for specific tasks (e.g., strings renaming, code generation)

- DSTLs more and more common:
  - Two papers on DSTLs at *ICMT’16* in a dedicated “*Model Transformation Languages*” session
  - This year *TTC’16* use case: data-flow based DSTL
  - Increasing need for methods to develop DSTLs

- Progress in executable Domain-Specific Modeling language (xDsML) engineering:
  - Generic *syntactic* services (e.g., editors using Xtext or Sirius)
  - Generic *runtime* services (e.g., debugger using GEMOC studio)
  - Easier and easier to obtain a tool-supported xDSML
Observations

- **Domain Specific Transformation Language (DSTL)** = model transformation language tailored for specific tasks (e.g., strings renaming, code generation)

- DSTLs more and more common:
  - Two papers on DSTLs at *ICMT’16* in a dedicated “*Model Transformation Languages*” session
  - This year *TTC’16* use case: data-flow based DSTL
  - **Increasing need for methods to develop DSTLs**

- Progress in executable Domain-Specific Modeling language (xDSML) engineering:
  - Generic *syntactic* services (e.g., editors using Xtext or Sirius)
  - Generic *runtime* services (e.g., debugger using GEMOC studio)
  - **Easier and easier to obtain a tool-supported xDSML**
Questions

Is it possible to apply techniques from xDSML engineering to define DSTLs?

How are xDSMLs and DSTLs related?
Questions

Is it possible to apply techniques from xDSML engineering to define DSTLs?

How are xDSMLs and DSTLs related?
Example of Petri nets xDSML and model
Example of Petri nets xDSML and model

Petri net model
Example of Petri nets xDSML and model

Petri net model

init=1
p1
init=0
p3
init=0
p4
init=1
p2

Bousse, Wimmer, Schwinger, Kapsammer

Leveraging Executable Lang. Engineering for DSTLs
Example of Petri nets xDSML and model

Petri net model
Example of Petri nets xDSML and model

Petri net model
Example of Petri nets xDSML and model

Abstract Syntax

Net
+transitions: Transition
+places: Place

Transition
+name: string
+input: 1..*
+output: 1..*
+initialTokens: int

Place
+name: string
+tokens: int
+initialTokens: int

Operational semantics

run(Net)
: while there is an enabled transition, fires it.

fire(Transition)
: removes a token from each input Place, and adds a token to each output Place.

Execution transformation rules (summarized)

Petri net model

p1 t1 p3 t2 p4
init=1 init=0 init=0

p2
init=1

Leveraging Executable Lang. Engineering for DSTLs
Example of Petri nets xDSML and model

Abstract Syntax

Net
- transitions *
  - input 1..*
  - output 1..*

Place
- name: string
- initialTokens: int

Transition
- name: string

Operational semantics

run(Net)
: while there is an enabled transition, fire it.

fire(Transition)
: removes a token from each input Place, and adds a token to each output Place.

Execution transformation rules (summarized)

Petri net model

- init=1
- init=0
- init=0

 Executed model

- init=1
- init=0
- init=0
Example of Petri nets xDSML and model

### Abstract Syntax

- **Net**
  - **Place**
    - **+name**: string
    - **+initialTokens**: int
  - **Transition**
    - **+name**: string

### Operational Semantics

- **run(Net)**: while there is an enabled transition, fire it.
- **fire(Transition)**: removes a token from each input Place, and adds a token to each output Place.

**Execution transformation rules (summarized)**

- while there is an enabled transition, fire it.
- removes a token from each input Place, and adds a token to each output Place.

**Petri net model**

```
init=1
 p1 ----> t1 ----> p3 ----> t2 ----> p4
init=0
init=0
```

**State Metamodel**

- **PlaceState**
  - **+tokens**: int

**Conforms to**

- Bousse, Wimmer, Schwinger, Kapsammer

**Leveraging Executable Lang. Engineering for DSTLs 4/10**
Example of Petri nets xDSML and model

Abstract Syntax
input 1..* output 1..*
Net
Place
+name: string +initialTokens: int
Transition
+name: string

Operational semantics
: while there is an enabled transition, fires it.
: removes a token from each input Place, and adds a token to each output Place.

Execution transformation rules (summarized)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>run(Net)</td>
<td>while there is an enabled transition, fires it.</td>
</tr>
<tr>
<td>fire(Transition)</td>
<td>removes a token from each input Place, and adds a token to each output Place.</td>
</tr>
</tbody>
</table>

Petri net model

init=1
init=0
init=0

p1 t1 p3 t2 p4
init=1

Petri net model

Execution model
(t1 fired) (t2 fired)
Generalizing xDSMLs

Diagram:

- **Parameter metamodel**
- **Execution transformation**
- **State metamodel**
- **Abstract syntax**
- **Parameter model**
- **Executable model**
- **Execution state**

**Abstract syntax**
**xDSML**
**Executable model**
**Parameter metamodel**
**State metamodel**
**Execution transformation**

**Bousse, Wimmer, Schwinger, Kapsammer**

**Leveraging Executable Lang. Engineering for DSTLs**
Example of MiniTL DSTL and model
Example of MiniTL DSTL and model

MiniTL model
Example of MiniTL DSTL and model

```
transformation simpleAtoB {
}
```

MiniTL model
Example of MiniTL DSTL and model

transformation simpleAtoB {
  rule AToB {

  }
}

MiniTL model
Example of MiniTL DSTL and model

transformation simpleAtoB {
    rule AToB {
        from a : metamodelA.A
        to b : metamodelB.B {
            y = a.x + "_out";
        }
    }
}

MiniTL model
Example of MiniTL DSTL and model

transformation simpleAtoB {
    rule AtoB {
        from a : metamodelA.A
        to b : metamodelB.B {
            y = a.x + "_out"
        }
    }
}
Example of MiniTL DSTL and model

transformation simpleAtoB {
    rule AToB {
        from a : metamodelA.A
        to b : metamodelB.B {
            y = a.x + "_out";
        }
    }
}

MiniTL model
Example of MiniTL DSTL and model

**MiniTL model**

```plaintext
transformation simpleAtoB {
  rule AToB {
    from a : metamodelA.A
    to b : metamodelB.B {
      y = a.x + "_out";
    }
  }
}
```

**Execution transformation rules (partial)**
- `run(Transformation)`: applies all the rules
- `apply(Rule)`: while there is a match, applies the rule on the match

**Ecore (partial)**
- `EClass
  eClass
  type 1`
- `EObject`

**Abstract Syntax (partial)**
- `Transformation
  rules
  *`  
- `ObjectTemplate
  +name: string
  input
  output
  type 1
  imports`  
- `Rule
  +name: string`  
- `outputModel`  

**State metamodel (partial)**
- `TransformationState
currentRule 0..1`  
- `EObject
eClass 1`  
- `input`  
- `outputModel`  
- `Operational semantics`  

**Conforms to**

**conforms to**

**Bousse, Wimmer, Schwinger, Kapsammer**

**Leveraging Executable Lang. Engineering for DSTLs** 6/10
Example of MiniTL DSTL and model

transformation simpleAtoB {
  rule AToB {
    from a : metamodelA.A
    to b : metamodelB.B {
      y = a.x + "_out";
    }
  }
}

MiniTL model

(AToB app.)

Bousse, Wimmer, Schwinger, Kapsammer

Leveraging Executable Lang. Engineering for DSTLs 6/10
Observations

From xDSMLs to DSTLs

Conclusion and future work

Example of MiniTL DSTL and model

```
transformation simpleAtoB {
  rule AtoB {
    from a : metamodelA.A
    to b : metamodelB.B {
      y = a.x + "_out";
    }
  }
}
```

MiniTL model

(Bousse, Wimmer, Schwinger, Kapsammer)
Generalizing DSTLs as specific xDSMLs
Research directions

- Experiment with **generic and generative approaches** for DSTL engineering:
  - Reuse xDSML engineering approaches, *eg.* getting a debugger “for free” for a given DSTL
  - Define/adapt new generic approaches for DSTL engineering

- Evaluate the **implications of DSTL specificities**: *eg.* can we generate a usable/relevant debugger using generic approaches?

- **DSTLs as case studies** for xDSML engineering (cf. TTC’16)
Research directions

- Experiment with **generic and generative approaches** for DSTL engineering:
  - Reuse xDSML engineering approaches, *e.g.* getting a debugger “for free” for a given DSTL
  - Define/adapt new generic approaches for DSTL engineering

- Evaluate the **implications of DSTL specificities**: *e.g.* can we generate a usable/relevant debugger using generic approaches?

- **DSTLs as case studies** for xDSML engineering (cf. TTC’16)
Research directions

- Experiment with **generic and generative approaches** for DSTL engineering:
  - Reuse xDSML engineering approaches, *e.g.* getting a debugger “for free” for a given DSTL
  - Define/adapt new generic approaches for DSTL engineering

- Evaluate the **implications of DSTL specificities**: *e.g.* can we generate a usable/relevant debugger using generic approaches?

- **DSTLs as case studies** for xDSML engineering (cf. TTC’16)
Conclusion and future work

- Just an observation: DSTLs are a sort of xDSMLs, complex and with interesting characteristics
- Prospects:
  - Use state of the art xDSML engineering for DSTL engineering?
  - Consider DSTLs as nice case studies for model execution?

Future work

- **Short term**: Experiment (more) xDSML engineering on some transformation languages, eg. MiniTL
- **Long term**: analyse a DSTL to automatically provide it with a white-box testing framework (test model generation, coverage metrics, fault localization, etc.)
Implementation of MiniTL example:
https://github.com/tetrabox/minitl

Contact:
erwan.bousse@tuwien.ac.at
http://big.tuwien.ac.at/staff/ebousse

Research project:
TETRA Box: http://modeltransformation.net/tetrabox/
we have funding and an open position for a PhD student!
Generalisation of Metamodel-specific DSTLs
Screenshot of MiniTL debugging session

---

Bousse, Wimmer, Schwinger, Kapsammer