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 Kapsamer ²

¹TU Wien, Austria
²JKU Linz, Austria

October 3, 2016
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
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 (xDSDML) 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
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

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

Petri net model

init=1
p1
init=0
p3
init=0
p4
init=1
p2
Example of Petri nets xDSML and model
Example of Petri nets xDSML and model

Petri net model

Bousse, Wimmer, Schwinger, Kapsamer

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

**Abstract Syntax**

```
Net
  *transitions
  input 1..*
  output 1..*
```

**Transition**

```
+name: string
```

**Place**

```
+name: string
+initialTokens: int
```

**state Metamodel**

```
PlaceState
  +tokens: int
```

**Petri net model**

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

**Execution transformation rules (summarized)**

- `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.
Example of Petri nets xDSML and model
Example of Petri nets xDSML and model

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

Place
+name: string
+initialTokens: int

transitions *

input 1..*

output 1..*

Import

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

```
1=1
p1

init=0
p3

init=0
t2
t1

init=1
p2

```

Executed model

```
(t1 fired)

```

Bousse, Wimmer, Schwinger, Kapsammer  Leveraging Executable Lang. Engineering for DSTLs 4/10
Example of Petri nets xDSML and model

**Abstract Syntax**

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

**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)**

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

**Petri net model**

**Conforms to**

**State Metamodel**

- **PlaceState**
  - +tokens: int

**Conforms to**

**Initial state**

**Executed model**

(t1 fired) (t2 fired)
Generalizing xDSMLs

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

Model transformation

Model
data flow
depends on / uses
conforms to

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 {

}
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

Bousse, Wimmer, Schwinger, Kapsammer
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"
        }
    }
}

MiniTL model
Example of MiniTL DSTL and model

MiniTL model

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

Bousse, Wimmer, Schwinger, Kapsammer

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

Bousse, Wimmer, Schwinger, Kapsammer
Leveraging Executable Lang. Engineering for DSTLs 6/10
Example of MiniTL DSTL and model

MiniTL model

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

Initial model (initialization)

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

Execture model (AToB app.)

Bousse, Wimmer, Schwinger, Kapsammer

Leveraging Executable Lang. Engineering for DSTLs
Generalizing DSTLs as specific xDSMLs
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)
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)
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.)
Appendix

Done!

Thank you! 😊

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
Appendix

Screenshot of MiniTL debugging session

Bousse, Wimmer, Schwinger, Kapsamer

Leveraging Executable Lang. Engineering for DSTLs 10/10