



AutomobiLa

An Instance of Data Transform Network

Course: CS395T

Topic: Domain Specific Languages

Author: Willie Putrajaya

Outline

- ✦ Background
- ✦ Data Transform Network
- ✦ AutomobiLa overview
- ✦ Next steps (more work)
- ✦ Future directions



Background

- ✦ What is AutomobiLa? A DSL to assist automotive builder
- ✦ Basis for AutomobiLa? XML based Data Transform Network
- ✦ How does it look like?

```
<Engine id = 'myEngine'>  
  <Block id = 'myBlock'>  
    <Cylinder id = 'myCylinder1' diameter='100' />  
    <Cylinder id = 'myCylinder2' diameter='100' />  
  </Block>  
</Engine>
```



Background (cont)

★ Why AutomobiLa?

- + Ability to start as a simple DSL
- + Ability to extend to a full simulator
- + Ability to extend to detailed syntax

★ What is Data Transform Network?

- + Modularized Computational Framework
- + Collection of Transforms connected by Edges
- + Recursive definitions of Transforms
- + Supports multiple evaluation functions

Background (cont)

- ✦ Project status (currently):

- + Data Transform Network:

 - Missing the implementation of Configurable

- + AutomobiLa:

 - Missing the implementation of Configurable and wrapper Calculator

Data Transform Network

- ✦ Written in Java 1.4.2 (using JDOM and JUNIT libraries)
- ✦ Classes and Interfaces:
 - + Transform (A) → TransformComposite, SampleTransform
 - + TransformData (A) → TransformDouble, TransformInteger
 - + Configurable (I) → Transform
 - + TransformHelper and TransformException
- ✦ Transform (A) has Input & Output Signatures
- ✦ Signatures are Maps of TransformData (A)
- ✦ The Maps act as the state of Transform as well (currently)

Data Transform Network (cont)

- ★ Key Class to form Network: TransformComposite
- ★ TransformComposite contains Transform instances
- ★ TransformComposite has Input and Output Transforms (boundaries)
- ★ TransformComposite has Mapper (connection) list
- ★ TransformComposite performs error checking
- ★ TransformComposite performs evaluation
- ★ Error checking and Evaluation may have different implementations

Data Transform Network (cont)

- ✦ A Simple Example: SampleTransform
 - + Input Signature: $x(d), y(d), \text{oper}(s)$
 - + Output Signature: $\text{res}(d), \text{oper}(d)$
 - + Perform simple evaluations: $x + y$ or $x * y$
- ✦ Another Example: CompositeTransform
 - + Has 2 SampleTransform and 1 SampleTransformA (modified signature to pass error-checking)
 - + Both SampleTransforms are connected to SampleTransformA
 - + Perform evaluations: $(x + y) * (x + y)$ for example

Data Transform Network (cont)

- ★ Configurable Interface

- + Main methods: toConfiguration() and fromConfiguration()

- + Example for SampleTransform:

```
<Transform class='framework.model.SampleTransform'>
```

```
  <Data id='x' class='framework.model.TransformDouble' value='2'>
```

```
  <Data id='y' class='framework.model.TransformDouble' value='3'>
```

```
</Transform>
```

Data Transform Network (cont)

☀ Example for TransformComposite:

```
<Transform class='framework.model.TransformComposite' id='C1'>
  <TransformList>
    <Transform class='framework.model.SampleTransform' id='T1'>
      <Transform class='framework.model.SampleTransform' id='T2'>
        <Transform class='framework.model.SampleTransformA' id='T3'>
        </Transform>
      </Transform>
    </Transform>
  </TransformList>
  <MapperList>
    <Mapper source='T1' sink='T3' type='full' />
    <Mapper source='T2' sink='T3' type='full' />
  </MapperList>
  <InputTransforms> T1, T2 </InputTransforms>
  <OutputTransforms> T3 </OutputTransform>
</Transform>
```

Data Transform Network (cont)

- ★ TransformComposite: Transform trigger ordering method (currently)
 - + Basic tree traversal ordering
 - + Starting from Input Transforms
 - + (Hopefully) Ending with Output Transforms
 - + (Input) Values are propagated through from Input to Output

Data Transform Network (cont)

- ✦ TransformComposite: Mapper validity/error checking method (currently)
 - + Mapper = Source & Sink Transforms
 - + Source output signature \rightarrow subset of Sink input signature
 - + Composite Input signature \rightarrow union of Input Transforms input signature
 - + Composite Output Signature \rightarrow union of Output Transforms output signature



An Instance: AutomobiLa

✦ Building blocks for AutomobiLa:

- + Chassis, Engine, Cylinder, Piston, Radiator, Turbine, Intercooler, Caliper, Rotor, and even more models.
- + Simple Data Transform for every model
- + Gradually move to complex simulator
- + Transforms will run within the Framework

An Instance: AutomobiLa (cont)

★ An example Configuration:

```
<Transform class='framework.model.TransformComposite' id='TestCar1'>
  <TransformList>
    <Transform class='framework.model.TransformComposite' id='engine1'>
      <Transform class='framework.model.TransformComposite' id='chassis1'>
        <Transform class='framework.model.TransformComposite' id='brakeSet1'>
        </Transform>
      </Transform>
    </Transform>
  </TransformList>
  <MapperList>
    <Mapper source='chassis1' sink='engine1' type='full' />
    <Mapper source='engine1' sink='brakeSet1' type='full' />
  </MapperList>
  <InputTransforms> chassis1 </InputTransforms>
  <OutputTransforms> brakeSet1 </OutputTransform>
</Transform>
```

An Instance: AutomobiLa (cont)

☀ Example Configuration (cont):

```
<Transform class='framework.model.TransformComposite' id='Engine1'>
  <TransformList>
    <Transform class='framework.model.Block' id='block1'>
      <Transform class='framework.model.Piston' id='piston1'>
      <Transform class='framework.model.Piston' id='piston2'>
    </TransformList>
    <MapperList>
      <Mapper source='piston1' sink='block1' type='full' />
      <Mapper source='piston2' sink='block1' type='full' />
    </MapperList>
    <InputTransforms> block1 </InputTransforms>
    <OutputTransforms> block1 </OutputTransform>
  </Transform>
```



Next Steps (More Work)

- ✦ Fully implement TransformComposite
- ✦ Define and implement Configurable for all AutomobiLa Transform
- ✦ Cover most of AutomobiLa original design scope
- ✦ Move on to Future Directions

Future Directions

- ✦ Introduce some syntactic sugars for AutomobiLa DSL (XML)
- ✦ Implement basic first principle simulator for some AutomobiLa blocks
- ✦ Implement more advanced evaluation method (Transform trigger ordering)
- ✦ Implement Time-series version of Data Transform Network