The Whole Platform

An Experience Report

Philip Mitchell
David R. Cheriton School of Computer Science
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada N2L 3G1
pmitchel@uwaterloo.ca

April 1, 2012

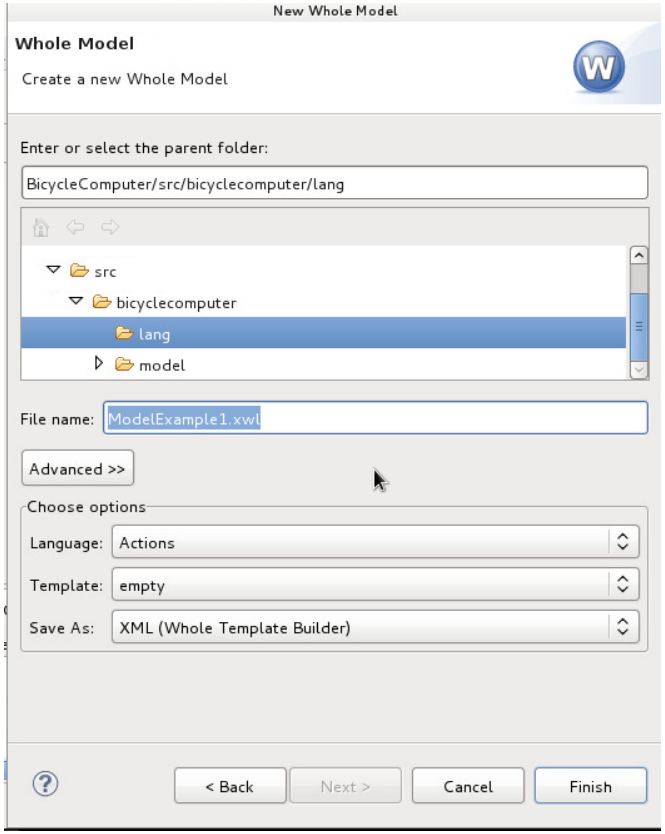
Outline

The Whole Platform

- Eclipse-based language workbench
- Designed to be used for:
  - developing new languages
  - manipulating them using domain notations
  - transforming them using a generative model driven approach

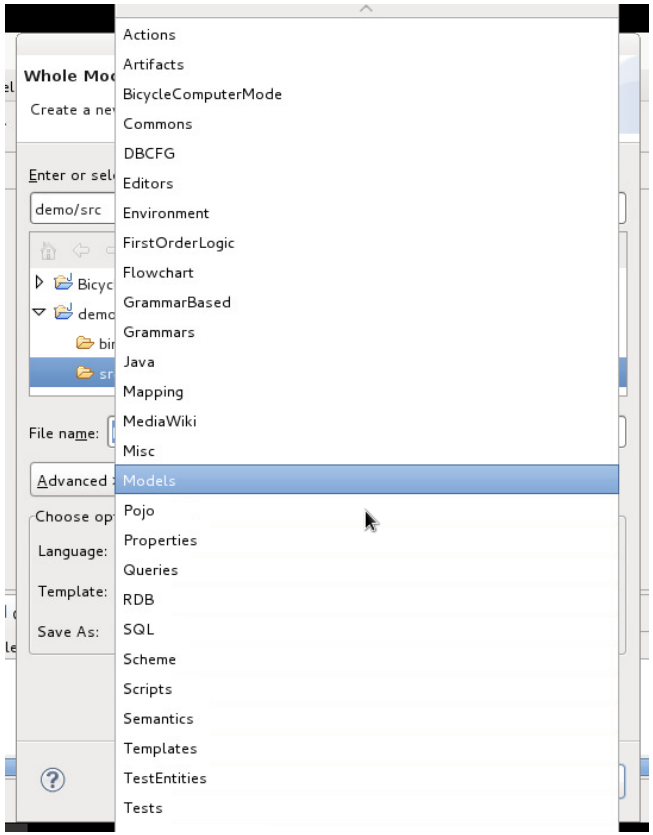
The Whole Platform

Creating a new whole model



The Whole Platform

Bundled languages



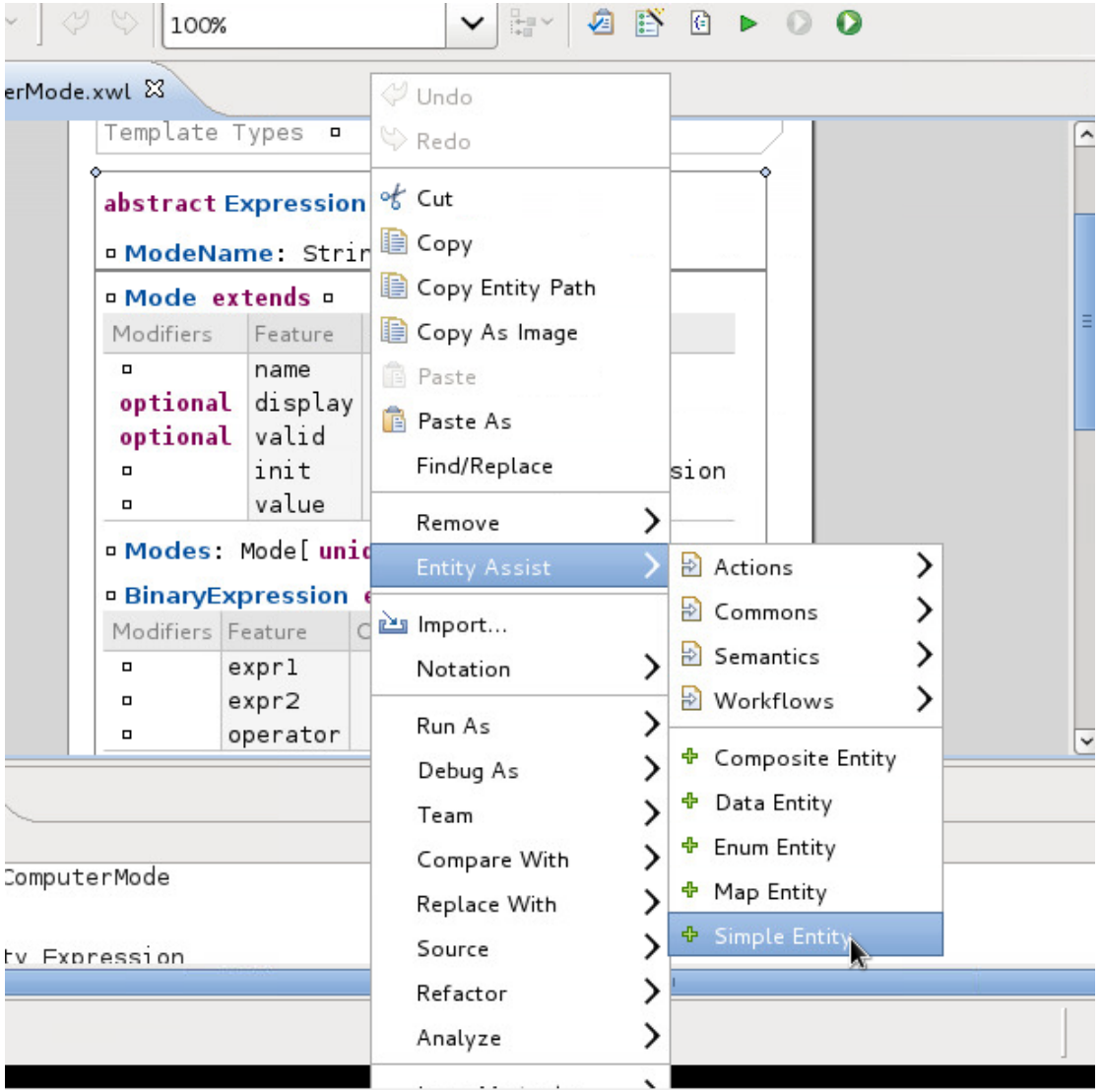
The Whole Platform

Bundled languages

- modeling languages
- query and transformation languages
- data integration languages for grammars, XSD, RDB and Java libraries
- popular general-purpose languages such as Java, Objective C and XML

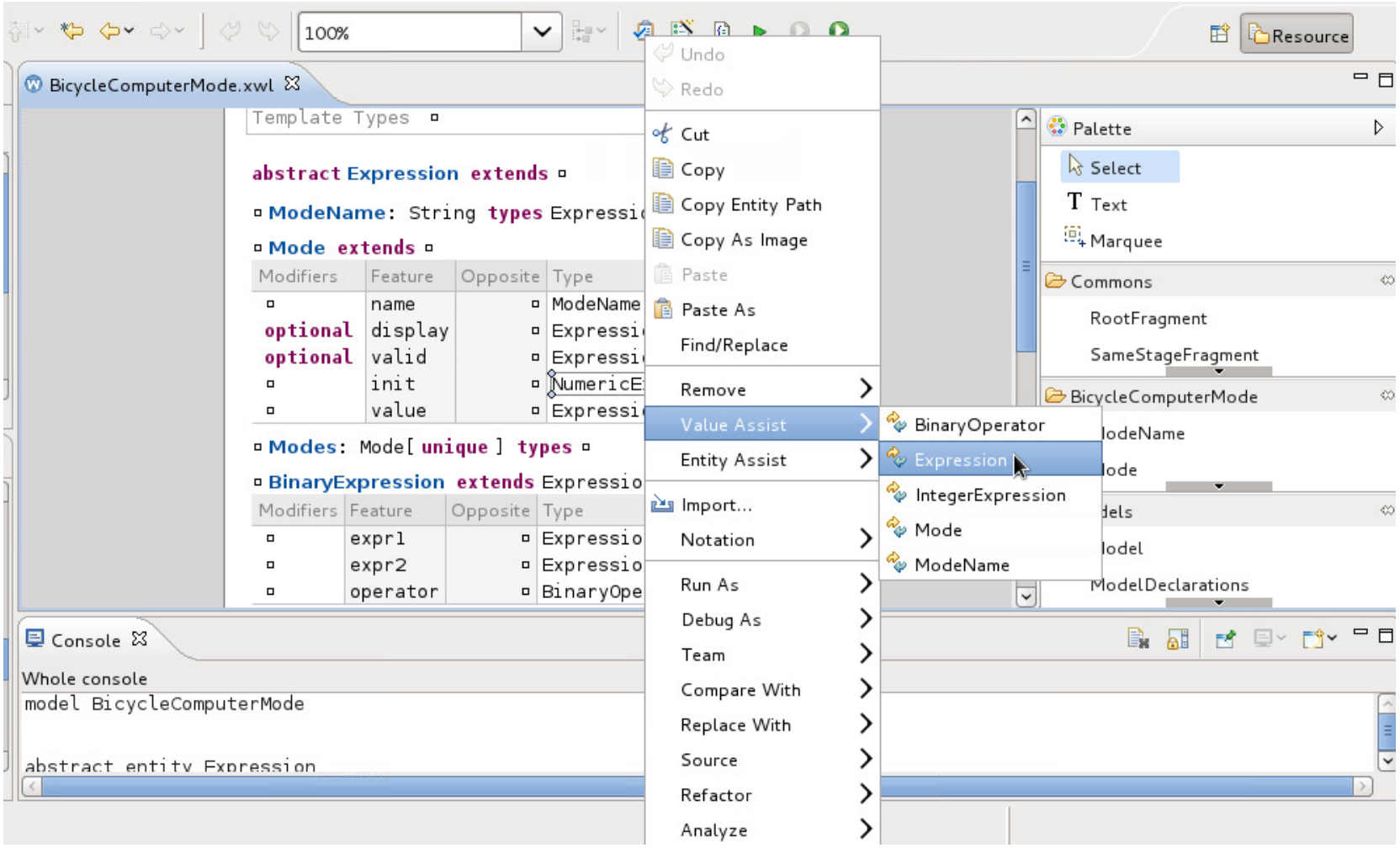
Creating the metamodel

Entity Assist



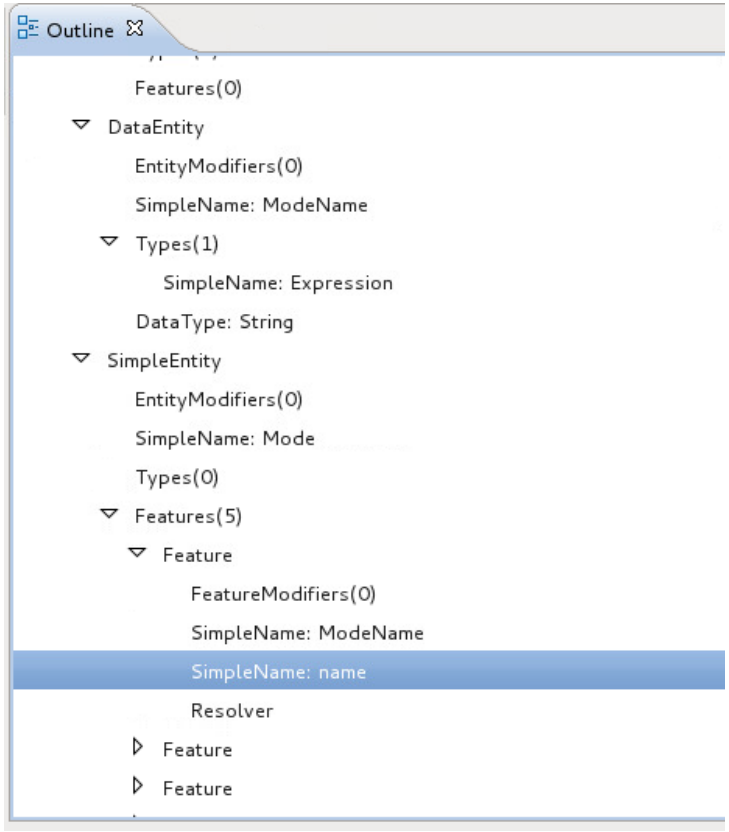
Creating the metamodel

Value Assist



Creating the metamodel

Outline



Creating the metamodel

Full meta model (part 1)

URI	http://lang.bicyclecomputer/Mode
Namespace	bicyclecomputer.mode
Model Name	BicycleComputerMode
Version	▫
Template Types	▫

abstract Expression **extends** ▫

▫ **ModeName**: String **types** Expression

▫ **Mode** **extends** ▫

Modifiers	Feature	Opposite	Type
▫	name	▫	ModeName
optional	display	▫	Expression
optional	valid	▫	Expression
▫	init	▫	NumericExpression
▫	value	▫	Expression

▫ **Modes**: Mode[**unique**] **types** ▫

▫ **BinaryExpression** **extends** Expression

Modifiers	Feature	Opposite	Type
▫	expr1	▫	Expression
▫	expr2	▫	Expression
▫	operator	▫	BinaryOperator

▫ **BinaryOperator**: String **types** ▫

Creating the metamodel

Full meta model (part 2)

▫ **BinaryOperator**: String **types** ▫

▫ **NumericExpression**: double **types** Expression

▫ **RationalExpression** **extends** Expression

Modifiers	Feature	Opposite	Type
▫	numerator	▫	IntegerExpression
▫	denominator	▫	IntegerExpression

▫ **IntegerExpression**: int **types** Expression

▫ **TernaryExpression** **extends** Expression

Modifiers	Feature	Opposite	Type
▫	expr1	▫	Expression
▫	expr2	▫	Expression
▫	condition	▫	Expression

▫ **LastValueExpression** **extends** Expression

▫ **StringExpression**: String **types** Expression

▫ **StringCatenationExpression** **extends** Expression

Modifiers	Feature	Opposite	Type
▫	expr1	▫	Expression
▫	expr2	▫	Expression

▫ **IntegerCastExpression** **extends** Expression

Modifiers	Feature	Opposite	Type
▫	value	▫	Expression

Creating the metamodel

Storage formats

- XWL*
- Java*
- Generated XML schema
- Grammars

Creating the metamodel

Storage formats: XWL

BicycleComputerMode.xwl | BicycleComputerMode.xwl

```
models:DataEntity
  models:EntityModifiers initialCapacity="0"
  models:SimpleName value="ModeName"
  models:Types initialCapacity="1"
    models:SimpleName value="Expression"
  models:DataType value="String"
models:SimpleEntity
  models:EntityModifiers initialCapacity="0"
  models:SimpleName value="Mode"
  models:Types initialCapacity="0"
  models:Features initialCapacity="5"
    models:Feature
      models:FeatureModifiers initialCapacity="0"
      models:SimpleName value="ModeName"
```

Creating the metamodel

Storage formats: XML Schema

```
Environment
schema xmlns: = http://lang.bicyclecomputer/Mode xmlns: xsd = http://www.w3.org/2001/XMLSchema targetNamespace="http://lang.bicyclecomputer/Moc
  element name="modeName" type="ModeName"
  simpleType name="ModeName"
    restriction base="xsd:string"
  element name="mode" type="Mode"
  complexType name="Mode"
    sequence
      element name="name" type="ModeName"
      group minOccurs="0" ref="Expression"
      group minOccurs="0" ref="Expression"
      element name="init" type="NumericExpression"
      group ref="Expression"
  element name="modes" type="Modes"
  complexType name="Modes"
    sequence minOccurs="0" maxOccurs="unbounded"
      element ref="mode"
  element name="binaryExpression" type="BinaryExpression"
```

Creating the metamodel

Storage formats: XML Mapping

Environment

schema xmlns: = http://lang.bicyclecomputer/Mode xmlns: xsd = http://www.w3.org/2001/XMLSchema targetNamespace="http://lang.bicyclecomputer/Moc

MappingStrategy

namespace	http://lang.bicyclecomputer/Mode
schemaLocation	BicycleComputerMode.xsd
synthesized	true
elementsFormQualified	true
mappings	

RootMapping

name	modeName
entityType	ModeName

RootMapping

name	mode
entityType	Mode

ElementMapping

contextEntityType	Mode
name	name
entityType	ModeName
featureType	name

ElementMapping

contextEntityType	Mode

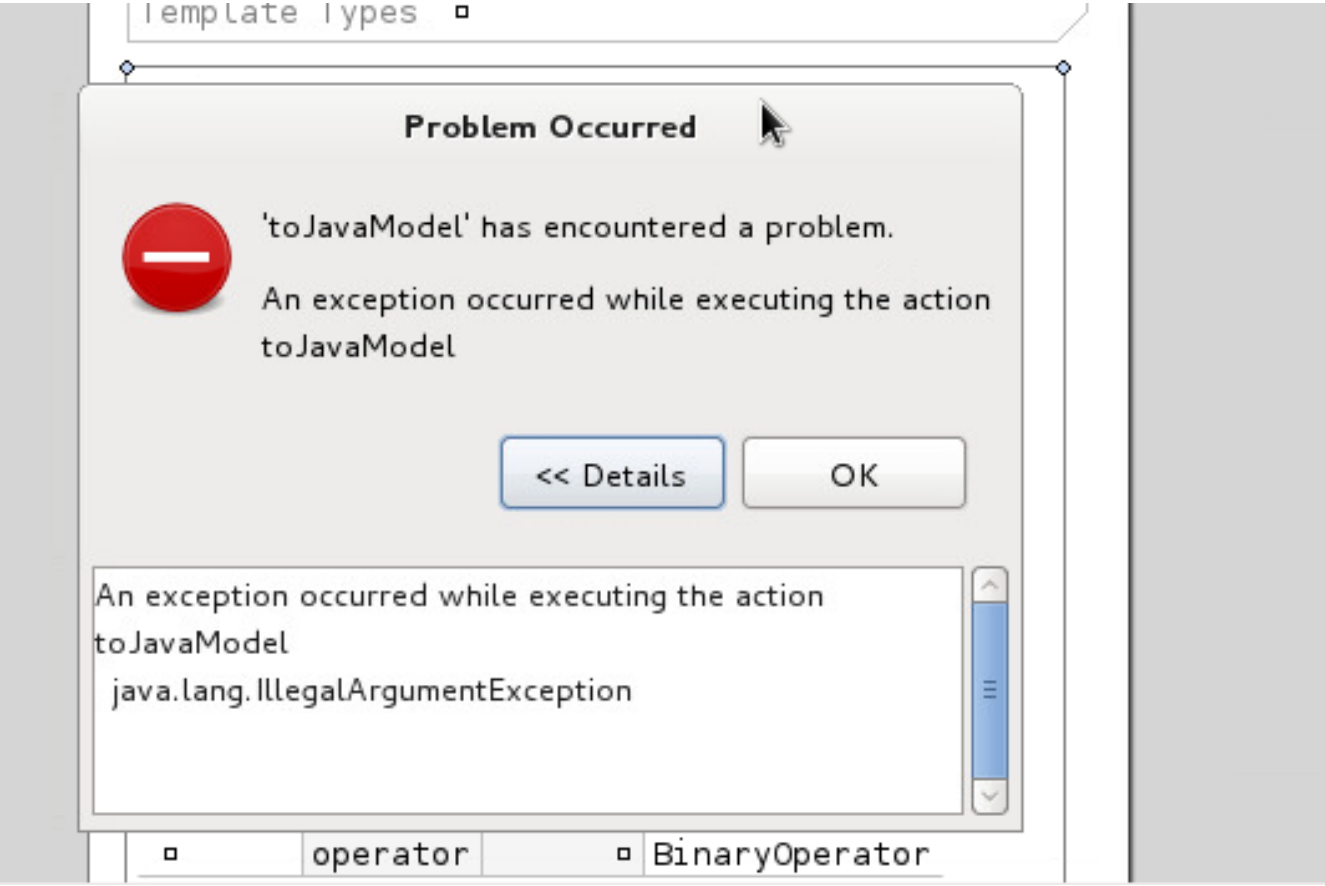
Using the metamodel

Generate Java code

- Generate Java code
  - creates classes required to create in-memory representations of models conforming to the metamodel
  - the Whole platform provides a library to load XWL- or Java-persisted models directly into a Java program
- Create a model transformation
  - transform models conforming to the metamodel into anything
  - query-based approach
  - generate target-model fragments in response to entities in source-model
  - several target models can be used in a single transformation

Using the metamodel

Generate Java code



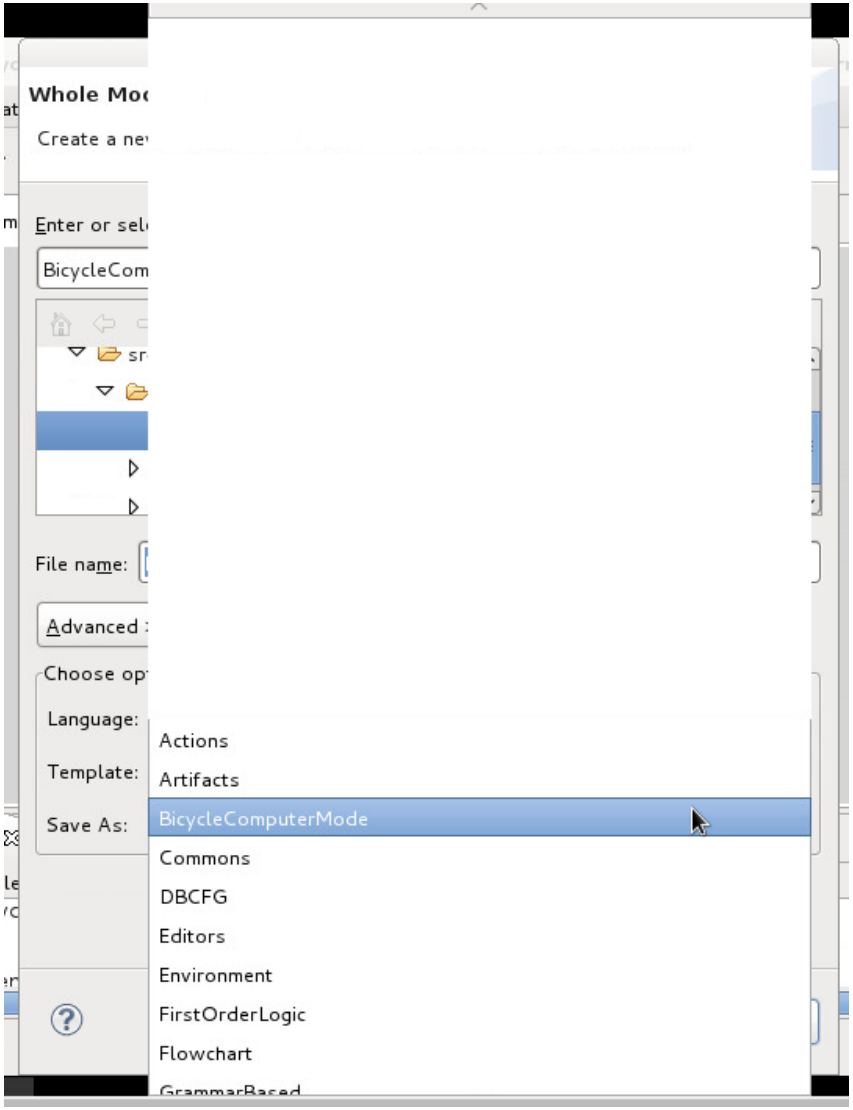
Using the metamodel

Transform to Java code

```
Mode Generator
query toJavaExpression
left, right, operator, modeName, expr1, expr2

select
    Artifacts
    Workspace
        projectName
        sourceFolderName
            beansPackageName
            interfacePackageName
            factoryPackageName

from self
    packageName+".model.beans" as beansPackageName
    packageName+".model" as factoryPackageName
    packageName+".model" as interfacePackageName
    "ModeFactory" as factoryClassName
    "IMode" as interfaceClassName
    factoryPackageName+"."+factoryClassName as qualifiedFactoryName
    interfacePackageName+"."+interfaceClassName as qualifiedInterfaceName
    Artifacts
    interfaceClassName.java as interfaceClass

    where
        select
            Artifacts
            beanName.java as beanClass
            Java
            if ( beanName . equals ( modeName ) ) {
              return qualifiedBeanName . getInstance ( ) ; as beanFactorySwitch
            }
            Java
            getMode ( modeName ) . computeValue ( ) ; as beanRecomputeCall
            Java
            getMode ( modeName ) . resetValue ( ) ; as beanResetCall
            Commons
            beanName as beanNameList

        from descendant-or-self [BicycleComputerMode Mode][BicycleComputerMode name] as beanName
            beansPackageName+"."+beanName as qualifiedBeanName
```

Creating the model

Create the new BicycleComputerMode model



Creating the model

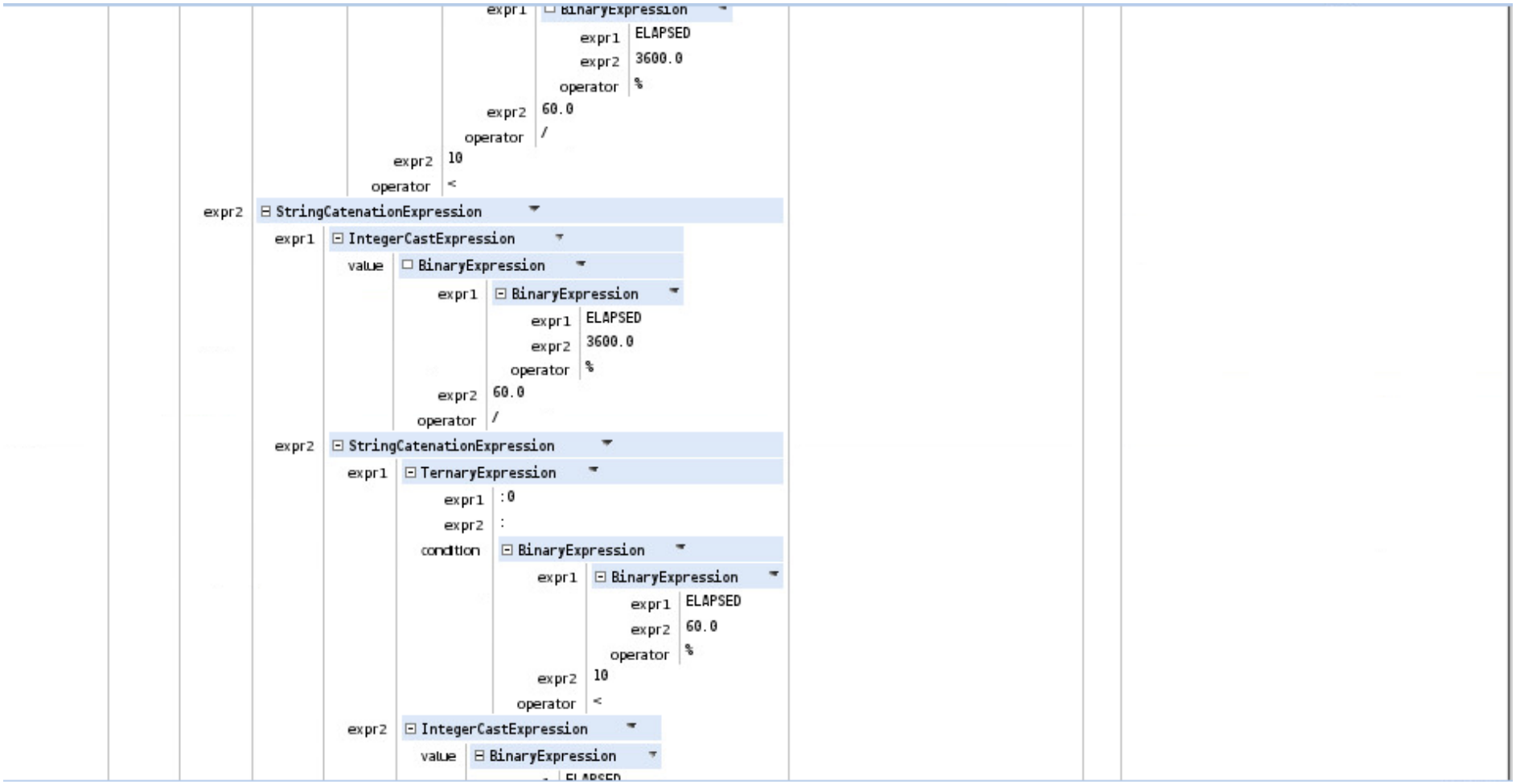
Full model (part 1)

Modes

name	display	valid	init	value
IROTATIONS			0.0	LastValueExpression
IDURATION			0.0	LastValueExpression
WHEELSIZE			0.0	LastValueExpression
SPEED	StringCatenationExpression expr1: IntegerCastExpression value: BinaryExpression expr1: SPEED expr2: 3.6 operator: * expr2:		0.0	BinaryExpression expr1: BinaryExpression expr1: IROTATIONS expr2: WHEELSIZE operator: * expr2: IDURATION operator: /
ELAPSED	StringCatenationExpression expr1: IntegerCastExpression value: BinaryExpression expr1: BinaryExpression expr1: ELAPSED expr2: 36000.0 operator: % expr2: 3600.0 operator: / expr2: StringCatenationExpression expr1: TernaryExpression expr1: :0 expr2: : condition: BinaryExpression expr1: BinaryExpression expr1: BinaryExpression expr1: ELAPSED		0.0	BinaryExpression expr1: LastValueExpression expr2: IDURATION operator: +

Creating the model

Full model (part 2)



Creating the model

Full model (part 3)

					value	BinaryExpression expr1: ELAPSED expr2: 60.0 operator: %			
MAXSPEED	StringCatenationExpression expr1: IntegerCastExpression value: BinaryExpression expr1: MAXSPEED expr2: 3.6 operator: * expr2: km/hr							0.0	TernaryExpression expr1: SPEED expr2: LastValueExpression condition: BinaryExpression expr1: SPEED expr2: LastValueExpression operator: >
TRIP	StringCatenationExpression expr1: StringCatenationExpression expr1: IntegerCastExpression value: BinaryExpression expr1: BinaryExpression expr1: TRIP expr2: 1000000.0 operator: % expr2: 1000.0 operator: / expr2: StringCatenationExpression expr1: . expr2: StringCatenationExpression expr1: IntegerCastExpression value: BinaryExpression expr1: BinaryExpression expr1: TRIP expr2: 1000.0 operator: %							0.0	BinaryExpression expr1: BinaryExpression expr1: IROTATIONS expr2: WHEELSIZE operator: * expr2: LastValueExpression operator: +

Creating the model

Full model (part 4)

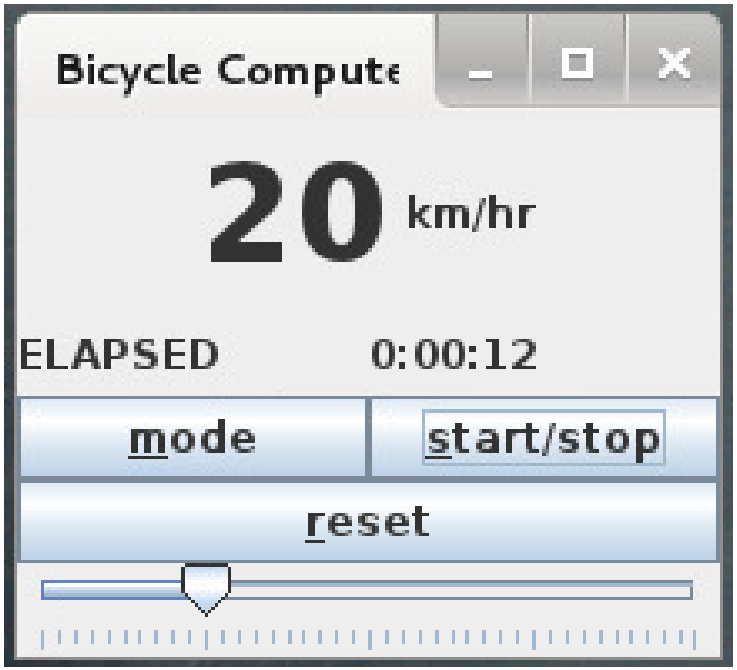
	expr2: StringCatenationExpression expr1: . expr2: StringCatenationExpression expr1: IntegerCastExpression value: BinaryExpression expr1: BinaryExpression expr1: TRIP expr2: 1000.0 operator: % expr2: 100.0 operator: / expr2: IntegerCastExpression value: BinaryExpression expr1: BinaryExpression expr1: TRIP expr2: 100.0 operator: % expr2: 10.0 operator: / expr2: km			
AVGSPEED	StringCatenationExpression expr1: IntegerCastExpression value: BinaryExpression expr1: AVGSPEED expr2: 3.6 operator: * expr2: km/hr	BinaryExpression expr1: BinaryExpression expr1: ELAPSED expr2: 360000 operator: < expr2: BinaryExpression expr1: TRIP expr2: 1000000 operator: < operator: &&	0.0	BinaryExpression expr1: TRIP expr2: ELAPSED operator: /

Other platform features

- Debugging
- Testing
- Model validation
- Model normalization
- Versioning
- Deployment
- ...

Demo

The bicycle computer simulator



Lessons learned

- The Whole platform focuses on generic editors
- All metamodels and models are created equal
- Graphical editors are very expressive
  - Following tutorials can be very tricky
  - How do I input that symbol?
- It is convenient to separate language structure from syntax
- Documentation for the platform is very sparse
  - Primarily one document (their submission to the Language Workbench Competition 2011)
  - Several screencasts are available, but hard to follow