MetaProgramming using the OpenModelica Java Interface

Martin Sjölund

Outline

- External Java Functions
- Interactive Java session
- Applications

External Java Functions

```
function logJava
   input Real x;
   output Real y;
external ”Java”
   y = ’java.lang.Math.log’(x)
   annotation (
       JavaMapping=”simple”
   );
end logJava;
```

- Not standardized
- Dymola-style function name
- Own datatype mappings
  - Simple types
  - Full MetaModelica

Mapping of Datatypes

Modelica	External Java
Real	ModelicaReal
Integer	ModelicaInteger
Boolean	ModelicaBoolean
String	ModelicaString
Record	ModelicaRecord
Uniontype	IModelicaRecord
list<T>	ModelicaArray<T>
tuple<T1,...,Tn>	ModelicaTuple
Option<T>	ModelicaOption<T>
T[:]	ModelicaArray<T>
array<T>	???

Modelica	External Java
Real	double
Integer	int
Boolean	bool
String	String

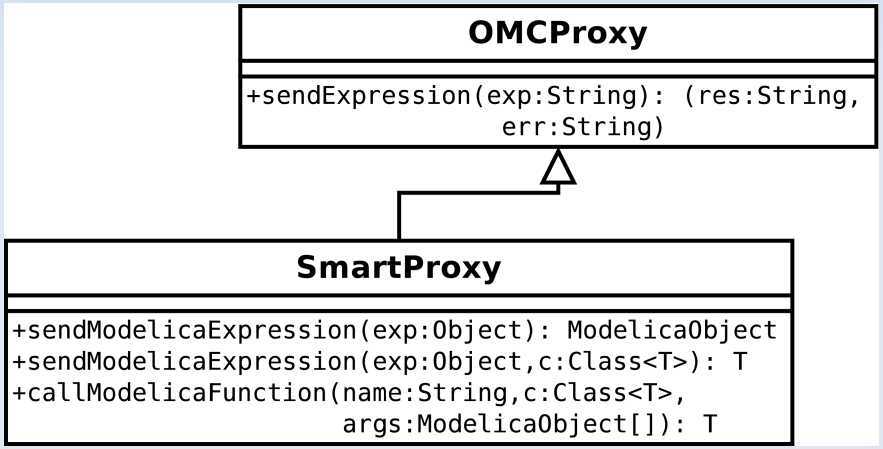
JNI

- Java Native Interface
  - C Runtime
  - Loaded dynamically
  - Uses modelica_java.jar runtime

Interactive Java Session

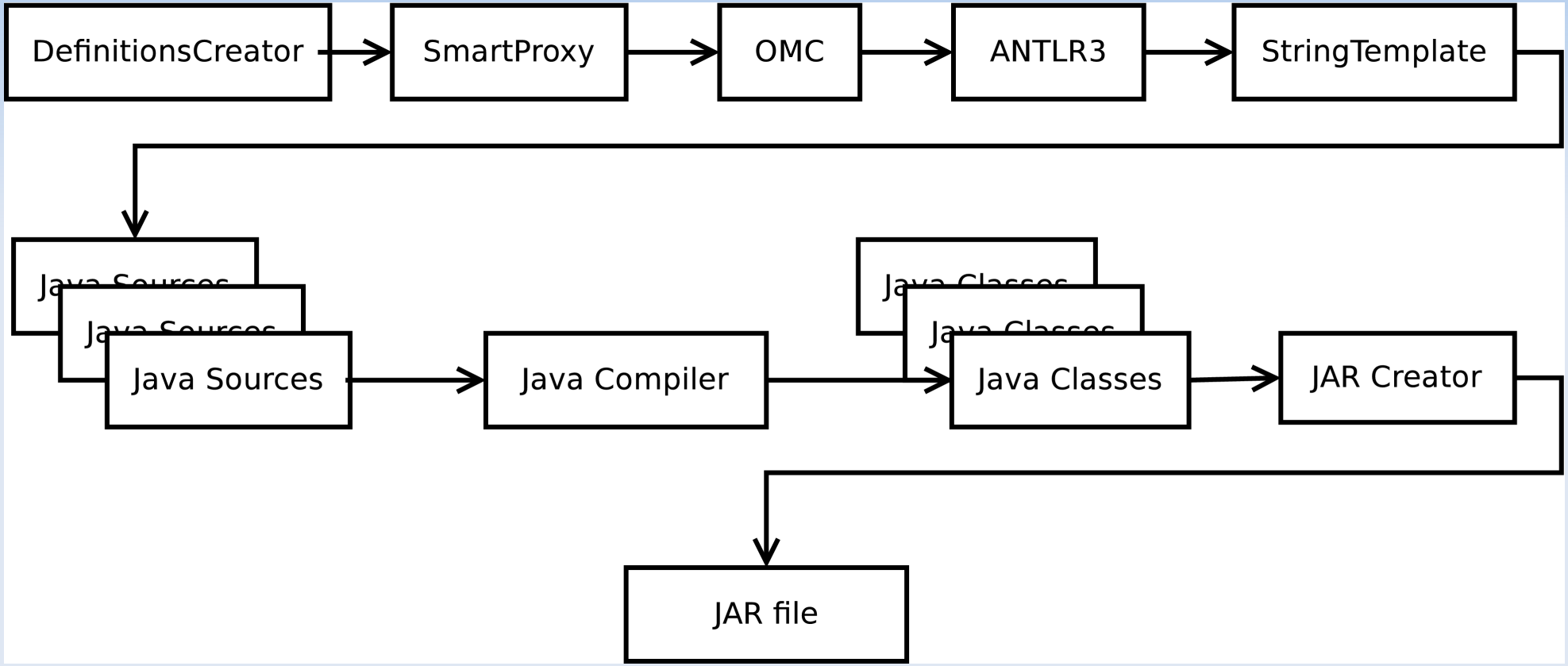
- Interactive Module
  - CORBA, untyped
  - CORBA, typed
- Not implemented
  - Socket
  - JNI

SmartProxy



- Existing CORBA communicator
- Custom parser
  - ANTLR too slow
- Typed interface

Translating MetaModelica Definitions to Java Interface



Example Functions

```
package Simple
function AddOne
    input Integer i;
    output Real out;
    Integer one = 1;
algorithm
    out := i+one;
end AddOne;
```

```
function AddTwo
    input Integer i;
    output Integer out1;
    output Integer out2;
algorithm
    out1 := i+1;
    out2 := i+2;
end AddTwo;
end Simple;
```

Generated Java Code (1/3)

```
public class AddOne extends ModelicaFunction {
    public AddOne (SmartProxy proxy) {
        super(”AddOne”, proxy) ;
    }
    public ModelicaReal call(ModelicaInteger i) throws
            ParseException, ConnectException
    {
        return proxy.callModelicaFunction(”Simple.AddOne”,
                ModelicaReal.class, i);
    }
}
```

Generated Java Code (2/3)

```
public class AddTwo extends ModelicaFunction {
   public ModelicaTuple call(ModelicaInteger i) ...;
   public void call(ModelicaInteger i,
                    ModelicaInteger out1,
                    ModelicaInteger out2) ...;
}
```

Generated Java Code (3/3)

```
public class BINARY  extends ModelicaRecord implements Exp_UT {

public BINARY(org.openmodelica.AbsynTest.Absyn.Exp_UT __exp1,
org.openmodelica.AbsynTest.Absyn.Operator_UT __op,
org.openmodelica.AbsynTest.Absyn.Exp_UT __exp2);

public BINARY(ModelicaObject o);

public BINARY(String recordName, Map map);

public  org.openmodelica.AbsynTest.Absyn.Exp_UT get_exp1();

public  org.openmodelica.AbsynTest.Absyn.Operator_UT get_op();

public  org.openmodelica.AbsynTest.Absyn.Exp_UT get_exp2();

public static BINARY parse(Reader r);

public int get_ctor_index();

}
```

Using the typed interface (1/2)

```
String printExpStr(Exp_UT e) {
  if (e instanceof INTEGER)
    return ((INTEGER)e).get_value().toString();
  if (e instanceof REAL)
    return ((REAL)e).get_value().toString();
  if (e instanceof BOOL)
    return ((BOOL)e).get_value().toString();
  if (e instanceof CREF)
    return printComponentRefStr(((CREF)e).get_componentRef());
}
```

Using the typed interface (2/2)

```
VAR updateBinding(VAR daeVar, ModelicaOption<Exp_UT> newBinding){
  VAR result = new VAR(daeVar);
  result.set_binding(newBinding);
  return result;
}
Element_UT updateBinding(Element_UT element,
ModelicaOption<Exp_UT> newBinding) {
  if (element instanceof VAR)
    return updateBinding((VAR)element,newBinding);
  else
    return element;
}
```

Applications

- With API extensions
  - MDT
  - Transforming the AST in Java (or bootstrap OMC)
- Java template languages
  - DefinitionsCreator
  - Could do unparsing in Java
- Simulations
  - Accessing tools written in Java
  - JDBC

Questions