TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science Institute for Software and Multimedia Engineering, Software Technology Group

Model-driven Modernisation of Java Programs with JaMoPP

F. Heidenreich, J. Johannes, J. Reimann, M. Seifert, C. Wende, C. Werner, C. Wilke, U. Aßmann

Oldenburg, March, 1st 2011

Agenda

1. **Motivation / Idea**
2. **JaMoPP**
3. **Model-driven Modernisation with JaMoPP**
4. **Examples**
5. **Evaluation**
6. **Conclusion**

Motivation

- **Languages evolve**
  - Introduction of new language features
  - Replace constructs with more compact ones
    - Generics (Java 5)
    - **For Loops (Java 5)**
    - **Closures (Java 8)**
  - But, backward compatibility (almost always) required

- **Modernisation of old code is sensible**
  - Readability
  - Maintainability

- **Automated code modernisation would be nice**

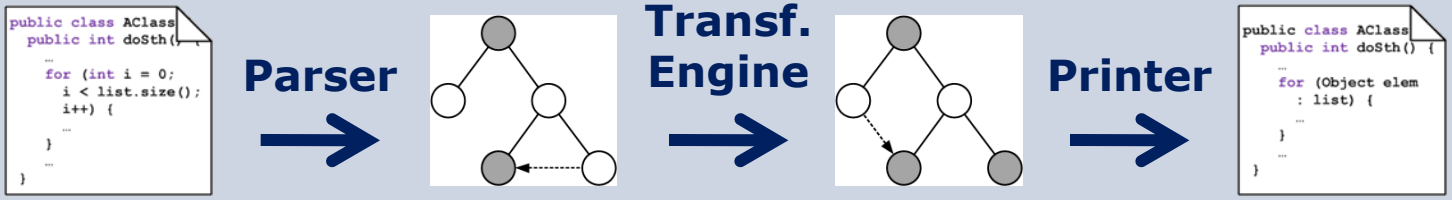
Idea

„Everything is a model.“

(Jean Bézivin)

Idea

- **Application of model transformation techniques** for automatic Java code modernisation



- **No modernisation-specific languages required**

- **Requires a model representation** of Java code
  1. Metamodel
  2. Parser
  3. Printer

  -> JaMoPP

Java Model Parser and Printer

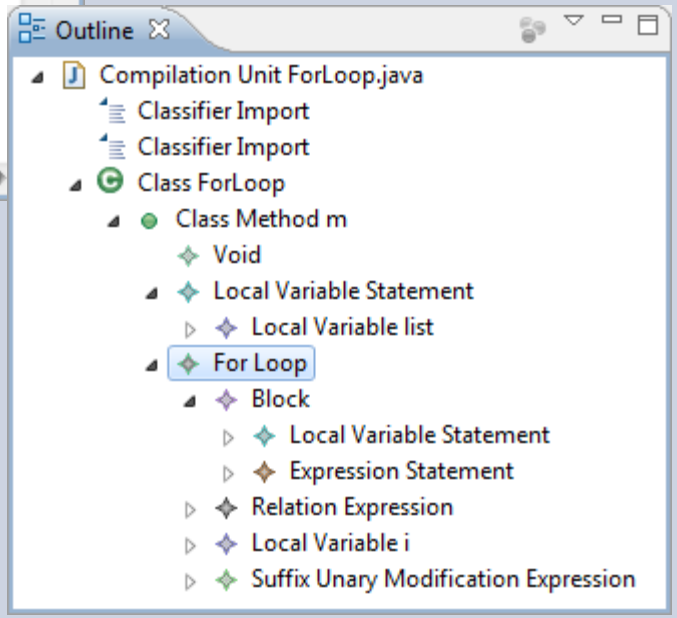
JaMoPP

ForLoop.java

```
import java.util.ArrayList;
import java.util.List;

public class ForLoop {

    public void m() {
        List<String> list = new ArrayList<String>();
        for (int i = 0; i < list.size(); i++) {
            String next = list.get(i);
            System.out.println(next);
        }
    }
}
```

Parses Java to EMF

Prints EMF to Java



Java Model Parser and Printer

- **EMF/Ecore-based metamodel** (abstract syntax graph) for Java
- **Tooling to parse/print Java** source code
  - Layout preservation
  - Name and type analysis (static semantics)
- **Based on EMFText**
  - **-> easy syntax extensions/integrations**
  - E.g., closures.
- Currently **supports Java 5**
  - Tested using a large amount of Java source code (~80k source files, ~15mil. non-empty LOC)
- http://www.jamopp.org/

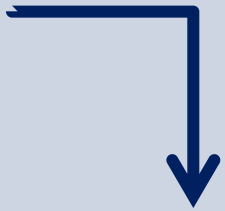
Model-Driven Source Code Transformation

- JaMoPP as Java parser
  -> EMF representation
- QVT-O as transformation engine
  (unidirectional transformation)
- JaMoPP as Java printer

For Loop Example (1/2)

ForLoop.java

```
import java.util.ArrayList;
import java.util.List;

public class ForLoop {

    public void m() {
        List<String> list = new ArrayList<String>();
        for (int i = 0; i < list.size(); i++) {
            String next = list.get(i);
            System.out.println(next);
        }
    }
}
```



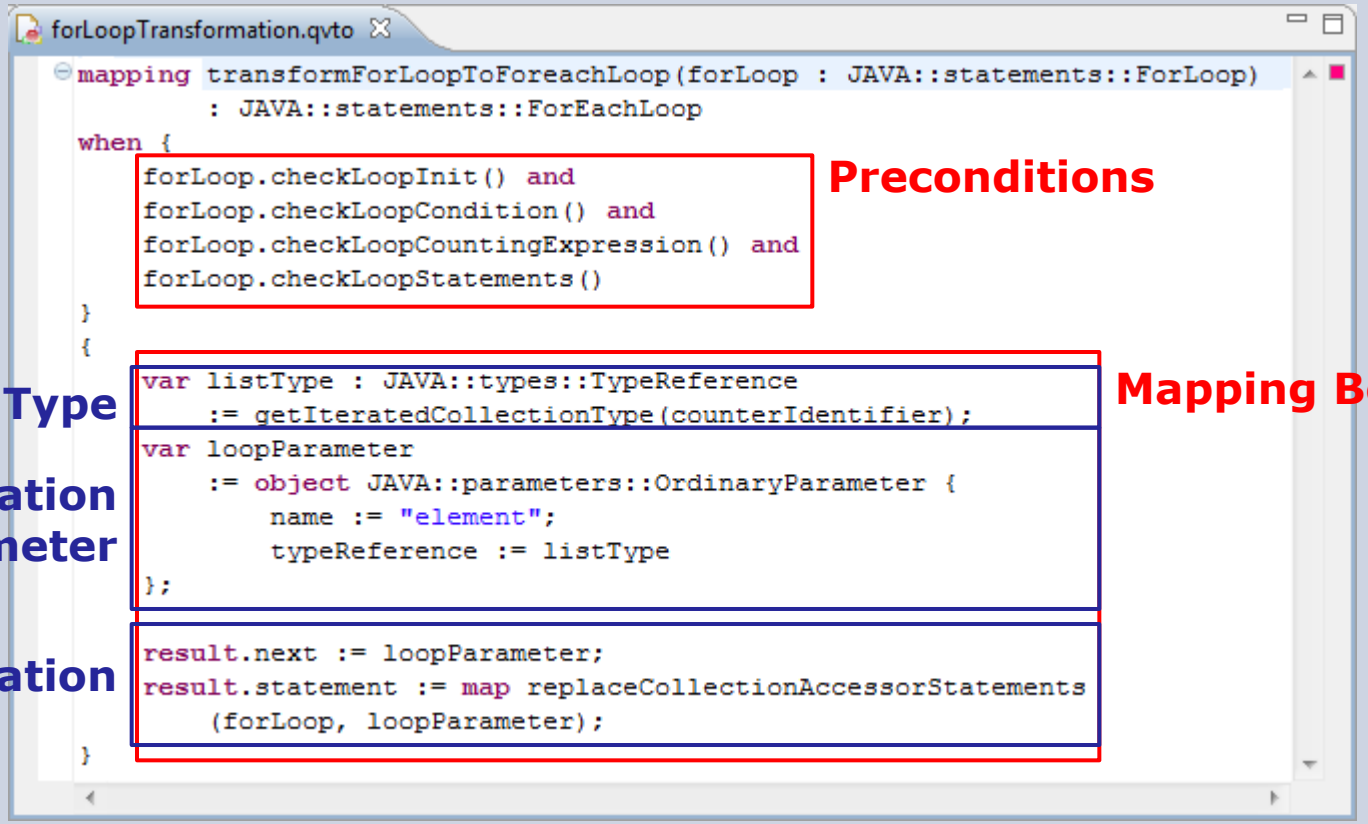
ForLoop.java

```
import java.util.ArrayList;
import java.util.List;

public class ForLoop {

    public void m() {
        List<String> list = new ArrayList<String>();
        for ( String element : list ) {
            String next = element;
            System.out.println(next);
        }
    }
}
```

Preconditions:

1. Index var = 0
2. Increment by 1
3. Iteration over complete collection
4. Only get(i) access

For Loop Example (2/2)



Closure Example (1/2)

- Closures emerged from functional programming
  - Idea: Use of functions as parameters/variables

```
public static void main( String[] args ) {
  { => System.out.println("Hallo Welt"); }.invoke();
}
```

```
{ int, int => int } addMethod = { int a, int b => (a + b) }
System.out.println( addMethod.invoke( 10, 20 ) );
```

- Java 6 only supports anonymous classes
- Closures are currently planned for Java 8
- Examples taken from:
  Zu früh gefreut: Java 7 ohne Closures - Closures werden nicht in Java 7 integriert sein. Jaxenter, February 2009.
  http://it-republik.de/jaxenter/artikel/Zu-frueh-gefreut-Java-7-ohne-Closures-2131

Closure Example (2/2)

SimpleCompare.java

```
import java.util.Comparator;

public class SimpleCompare {

    public void demo() {
        Comparator<String> comparator = new Comparator<String>() {

            public int compare(String x, String y) {
                /* Compares Strings in reverse order (Z < A) */
                return -(x.compareTo(y));
            }
        };

        System.out.println("Z < A? " + comparator.compare("Z", "A"));
    }
}
```

SimpleCompare.closure

```
import java.util.Comparator;

public class SimpleCompare {

    public void demo() {
        Comparator<String> comparator =
                {String, String => int} Comparator<String> :=
            {String x, String y =>
                /* Compares Strings in reverse order (Z < A) */
                return -(x.compareTo(y));
            };

        System.out.println("Z < A? " + comparator.invoke("Z", "A"));
    }
}
```

Preconditions:

1. Anonymous class used
2. Contains only one method

Evaluation

- Transformations on a set of public Java frameworks

Questions

- Is a general purpose modelling environment scalable enough?
- How many resources are required?
- Pro‘s and con‘s of modeltransformation-based code mondernisation

Source Code

- 10 open source projects (including Ant, Tomcat, JBoss)
- 16.402 Java files

Machine

- Dual Core AMD Opteron, 2.4 GHz, 4 GB RAM
- Only one core used (thread safety)

Evaluation: Performance

Framework	Files	For Loop	Closure
		Exec. in min.	Exec. in min.
AndroMDA 3.3	698	3	8
Apache Ant 1.8.1	829	5	21
Comns. Math 1.2	395	1	5
Tomcat 6.0.18	1127	4	15
GWT 1.5.3	1850	5	23
Jboss 5.0.0.GA	6414	16	70
Mantissa 7.2	242	1	3
Spring 3.0.0.M1	3096	8	43
Struts 2.1.6	1035	3	13
XerecesJ 2.9.1	716	3	12
Total	**16402**	**49**	**213**

Evaluation: Pattern Detection / Application

Framework	Files	For Loop		Closure	
		replaced	occurrence	replaced	occurrence
AndroMDA 3.3	698	4	959	8	201
Apache Ant 1.8.1	829	24	1028	12	99
Comns. Math 1.2	395	25	845	0	25
Tomcat 6.0.18	1127	65	1437	52	125
GWT 1.5.3	1850	29	1044	26	670
Jboss 5.0.0.GA	6414	472	2744	197	591
Mantissa 7.2	242	7	652	18	29
Spring 3.0.0.M1	3096	82	680	31	1403
Struts 2.1.6	1035	7	130	8	158
XerecesJ 2.9.1	716	21	1111	62	94
Total	**16402**	**736**	**10630**	**414**	**3395**

Advantages Using Modelling Tools

QVT-O

- Tooling helped a lot
  - Syntax highlighting and code completion in QVT-O editor
  - QVT-O debugger
  - Tracing information of QVT-O interpreter
- OCL knowledge was helpful

Modelling/EMF

- Graph-structure allows navigation through references
  - Eased specification of preconditions, e.g., for variables/references

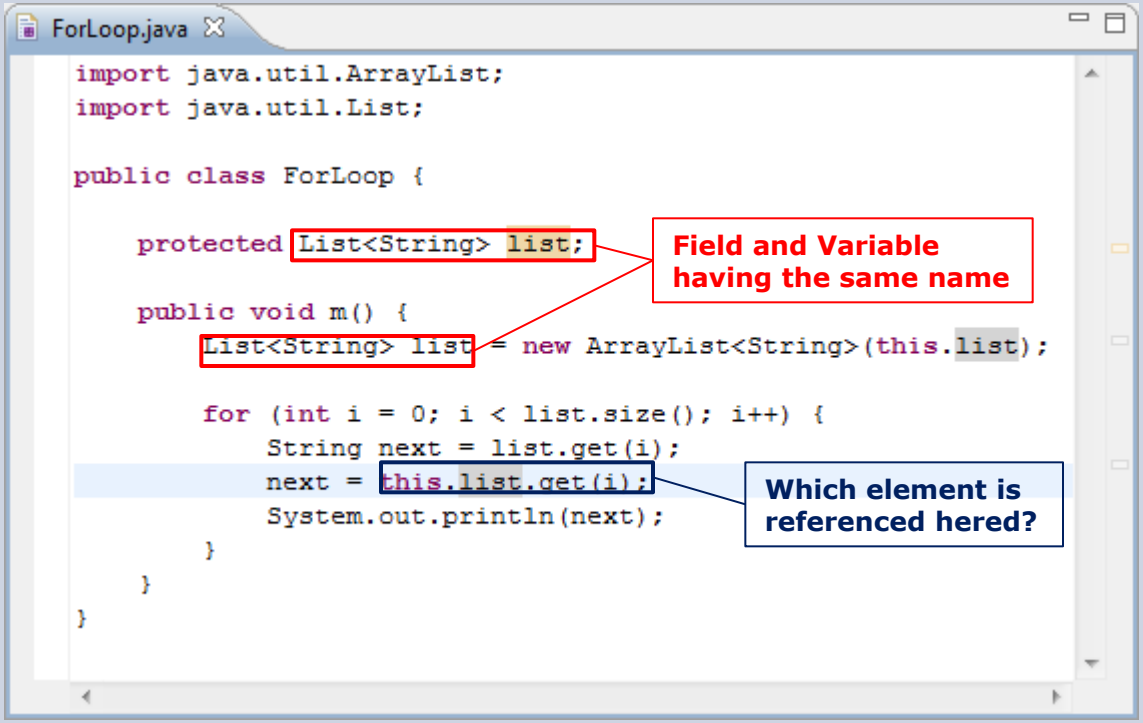
Existing Tooling

- Syntax and semantics checks are already there!
- All features can be reused, e.g. layout preservation

Why Graph-based Models are sensible

- Graphs allow references (e.g., to variables, fields, operations)
  - References improve pre/postcondition definitions

- Example: variable/field overlap



- **How to express this precondition in a tree-based structure?**
- **Name comparison is not enough!**

Disadvantages Using Modelling Tools

Work on Abstract Syntax

- Java metamodel is complex
  - Good knowledge of metamodel required

- Writting patterns in terms of the metamodel is complicate
  - Metamodel is concrete-syntax driven
  - Again, good knowledge/understanding required

- Execution time?

Conclusion

Modelling tools are applicable for source code modernisation

- Acceptable transformation time
- Acceptable match rates for patterns

Future Work

- Application of further patterns / modernisations
  - Elvis operator
  - JUnit 3 to JUnit 4
  - …
- Automated validation of transformation results (e.g., reexecution of unit tests)
- Comparison of JaMoPP/QVT-O approach with Java-specific source code transformation tools

Thank You!

emftext

http://www.emftext.org/

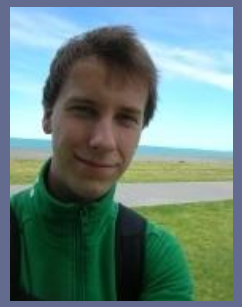
JaMoPP

http://www.jamopp.org/

Software Technology Group

http://st.inf.tu-dresden.de

claas.wilke@tu-dresden.de



Europa fördert Sachsen. ESF Europäischer Sozialfonds SACHSEN

Backup Slides

Closures as a Java Feature

- Closures were planned for Java 7 but were deferred to Java 8
- Currently Java 8 is scheduled for Late 2012

Literature:

[1] **Re-thinking JDK 7.** There's not a moment to lose! Mark Reinhold's Blog, September 2010. http://blogs.sun.com/mr/entry/rethinking_jdk7

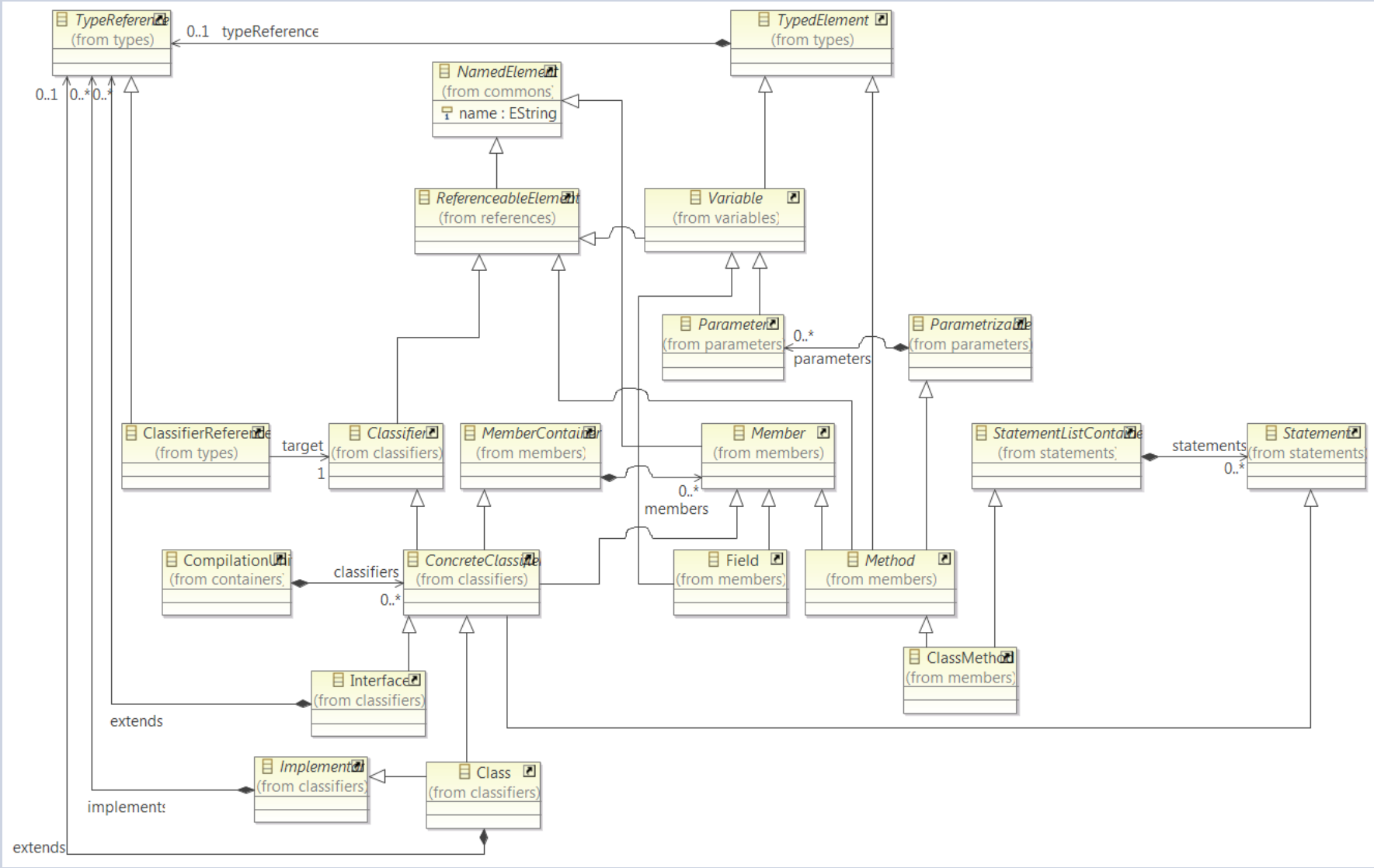
[2] **Zu früh gefreut: Java 7 ohne Closures - Closures werden nicht in Java 7 integriert sein.** Jaxenter, February 2009. http://it-republik.de/jaxenter/artikel/Zu-frueh-gefreut-Java-7-ohne-Closures-2131

[3] **Zu früh gejammert: Closures doch in Java 7 - Closures kommen mit Version 7 des JDK.** Jaxenter, December 2009. http://it-republik.de/jaxenter/artikel/Zu-frueh-gejammert-Closures-doch-in-Java-7-2748.html

JaMoPP Metamodel

- Metamodell is „concrete syntax-driven“
- Contains 243 classes

JaMoPP Metamodel



JaMoPP Layout Preservation

- JaMoPP parses Java source code to EMF
- White spaces and new lines are stored as well
  - Not directly within the Java metamodel
  - But as separate metadata
- During printing, white spaces and new lines are re-inserted into the source code
  - Defining default white spaces is possible for newly created elements within the concrete syntax specification

JaMoPP vs. MoDisco

- **Both are model-driven**

- Both use model-transformations, parsers and printers
  - JaMoPP uses own parser/printer
  - MoDisco uses JDK

- **Difference: Layout Preservation**
  - JaMoPP parses white spaces/new lines and stores them together with the model
  - Printer reuses these infos for layout preservation

- **Difference: JaMoPP parser can be extended**
  - Embedded DSLs
  - Embedded Java
  - Language evolution

JaMoPP vs. Stratego

- JaMoPP is graph-based (EMF/model)
- Stratego is tree-based (syntax-tree)
- Both can be used for language extensions/integrations