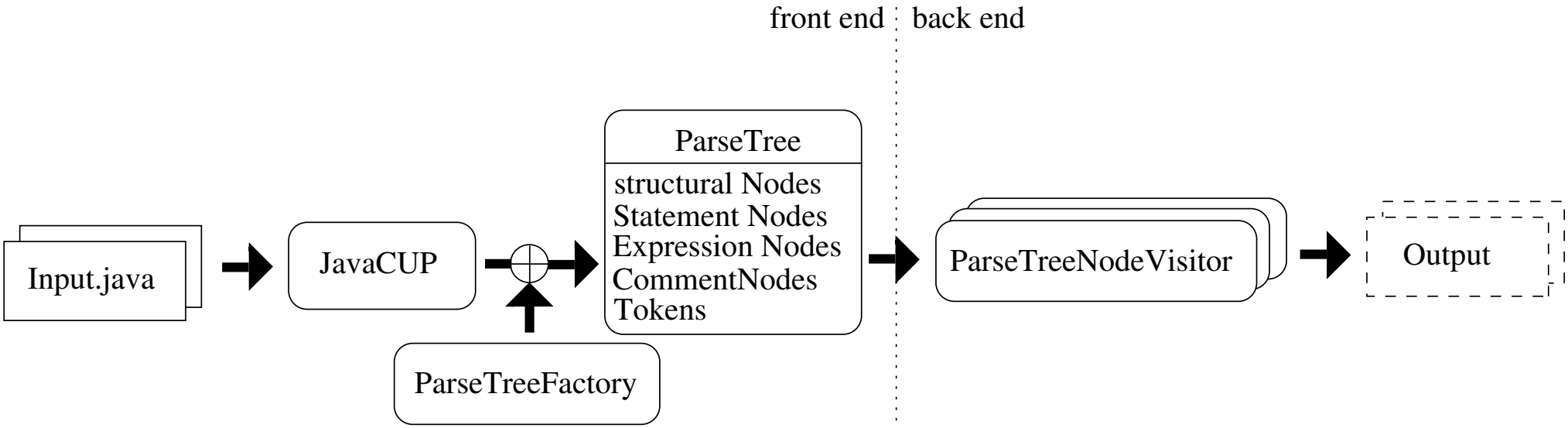
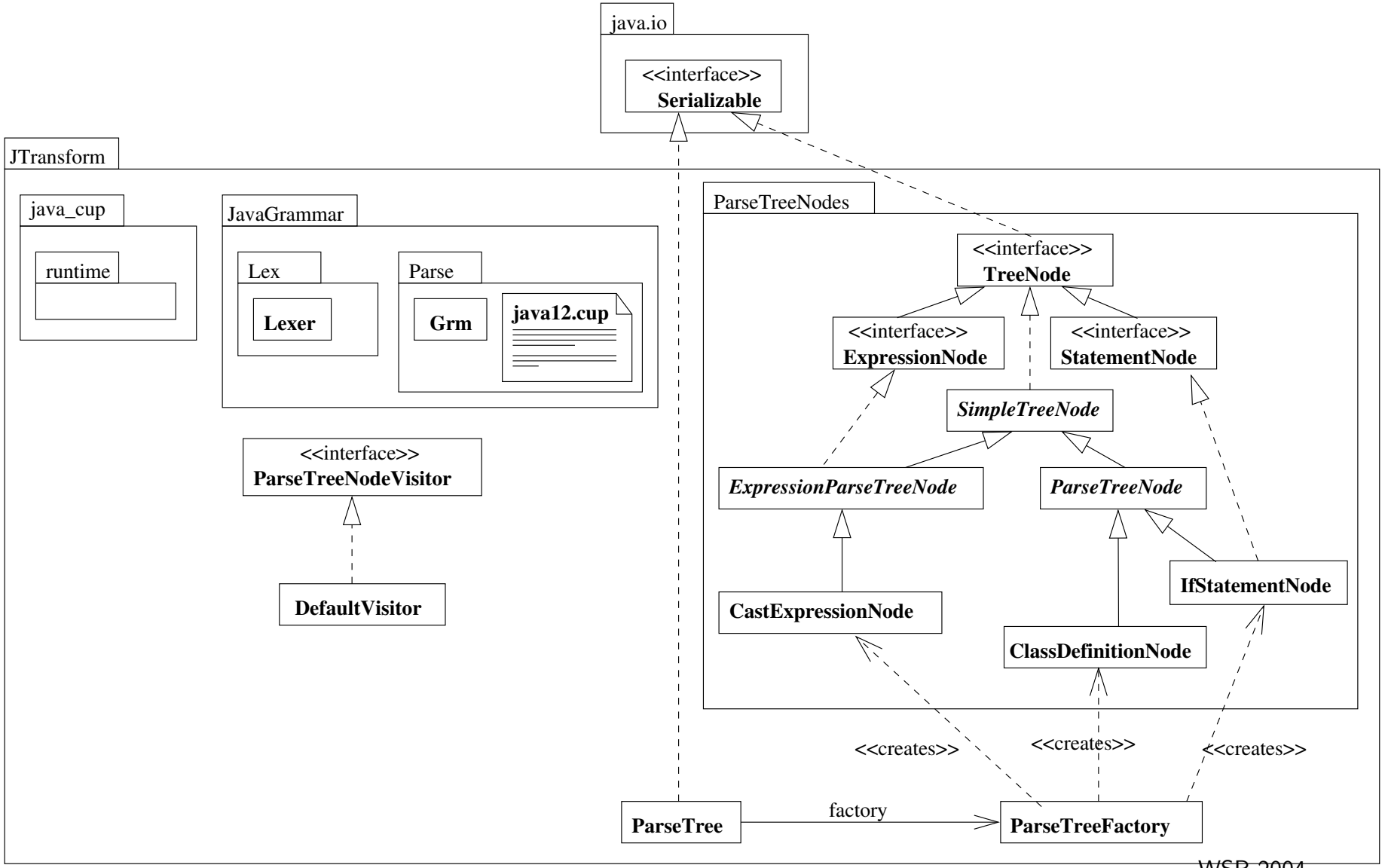
JTransform, a Tool for Source Code Analysis

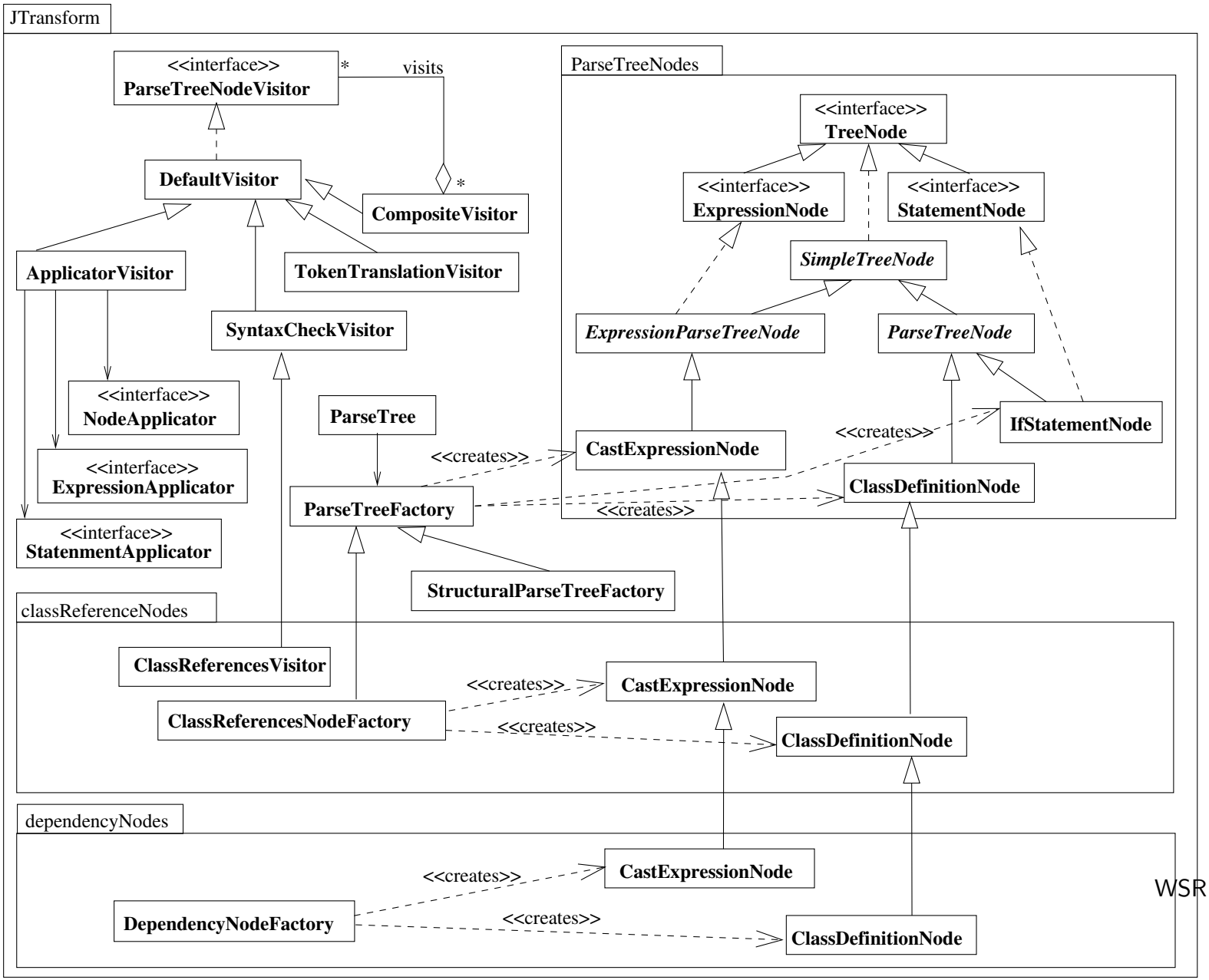
Holger Eichelberger and Jürgen Wolff von Gudenberg
Universität Würzburg

6. Workshop Software-Reengineering
Bad Honnef, 5.5.2004

Contents


front end
back end
ParseTree
structural Nodes
Statement Nodes
Expression Nodes
CommentNodes
Tokens
Input.java
JavaCUP
ParseTreeFactory
ParseTreeNodeVisitor
Output


java.io
<<interface>>
Serializable
JTransform
java_cup
runtime
JavaGrammar
Lex
Lexer
Parse
Grm
java12.cup
<<interface>>
ParseTreeNodeVisitor
DefaultVisitor
ParseTreeNodes
<<interface>>
TreeNode
<<interface>>
ExpressionNode
<<interface>>
StatementNode
SimpleTreeNode
ExpressionParseTreeNode
ParseTreeNode
IfStatementNode
CastExpressionNode
ClassDefinitionNode
<<creates>>
<<creates>>
<<creates>>
ParseTree
factory
ParseTreeFactory


JTransform
<<interface>>
ParseTreeNodeVisitor
*
visits
*
DefaultVisitor
CompositeVisitor
ApplicatorVisitor
TokenTranslationVisitor
SyntaxCheckVisitor
<<interface>>
NodeApplicator
<<interface>>
ExpressionApplicator
<<interface>>
StatenmentApplicator
ParseTree
ParseTreeFactory
StructuralParseTreeFactory
ParseTreeNodes
<<interface>>
TreeNode
<<interface>>
ExpressionNode
<<interface>>
StatementNode
SimpleTreeNode
ExpressionParseTreeNode
ParseTreeNode
IfStatementNode
<<creates>>
CastExpressionNode
ClassDefinitionNode
<<creates>>
<<creates>>
classReferenceNodes
ClassReferencesVisitor
<<creates>>
CastExpressionNode
ClassReferencesNodeFactory
<<creates>>
ClassDefinitionNode
dependencyNodes
<<creates>>
CastExpressionNode
DependencyNodeFactory
<<creates>>
ClassDefinitionNode

Configuration of Back Ends

- all back ends descendants of `MainTemplate`
  - redirect input and output streams
  - specify class loader
  - instance `main` method
  - chaining of back ends possible
- specialized visitor and node factory
- Interpretation and implementation of options

Dynamic Configuration

XML configuration file

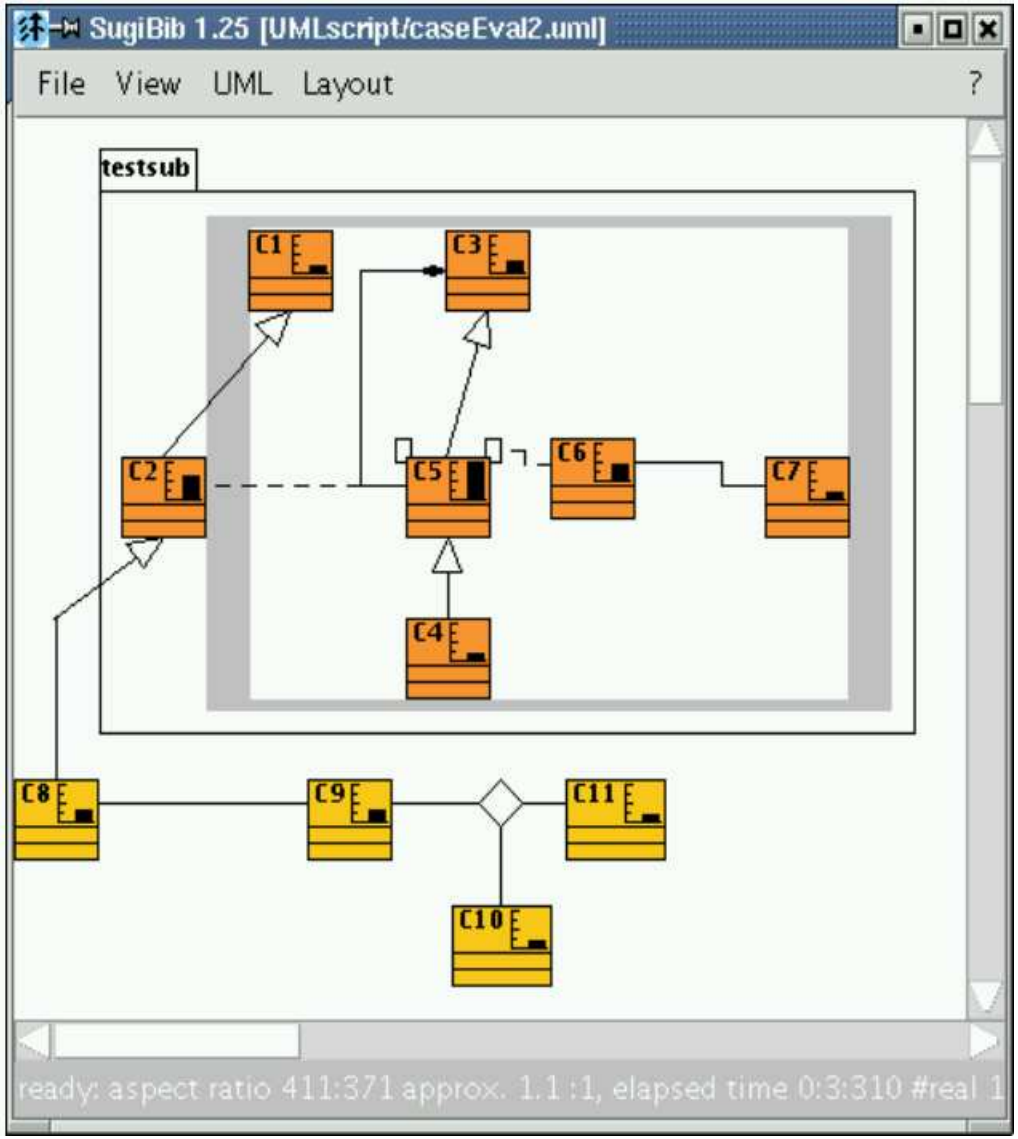
- interpreted by visitor specific configuration class
- delegate parts to components

Components

- check method
- report result of check
- information on node to be checked
- interpret configuration file

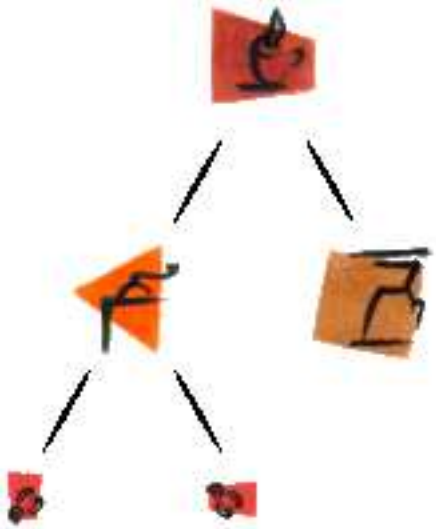
Visualisation by UML Diagrams

A visitor transforms the program into UMLscript that is input for the diagram drawing framework Sugibib


SugiBib 1.25 [UMLscript/caseEval2.uml]
File View UML Layout ?
testsub
C1
C3
C2
C5
C6
C7
C4
C8
C9
C11
C10
ready: aspect ratio 411:371 approx. 1.1 :1, elapsed time 0:3:310 #real 1

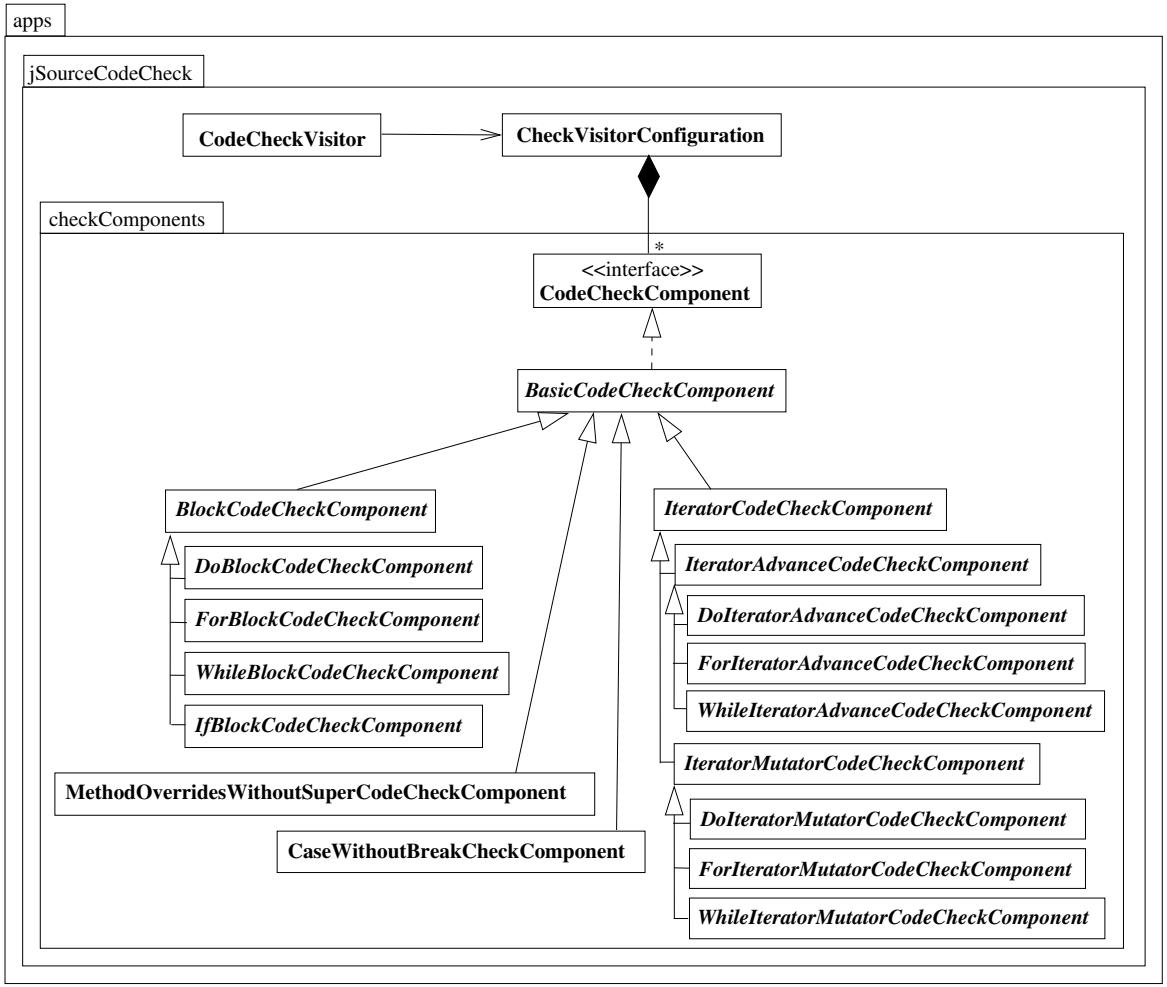
JOP

http://jop.informatik.uni-wuerzburg.de



Assessment of Programs

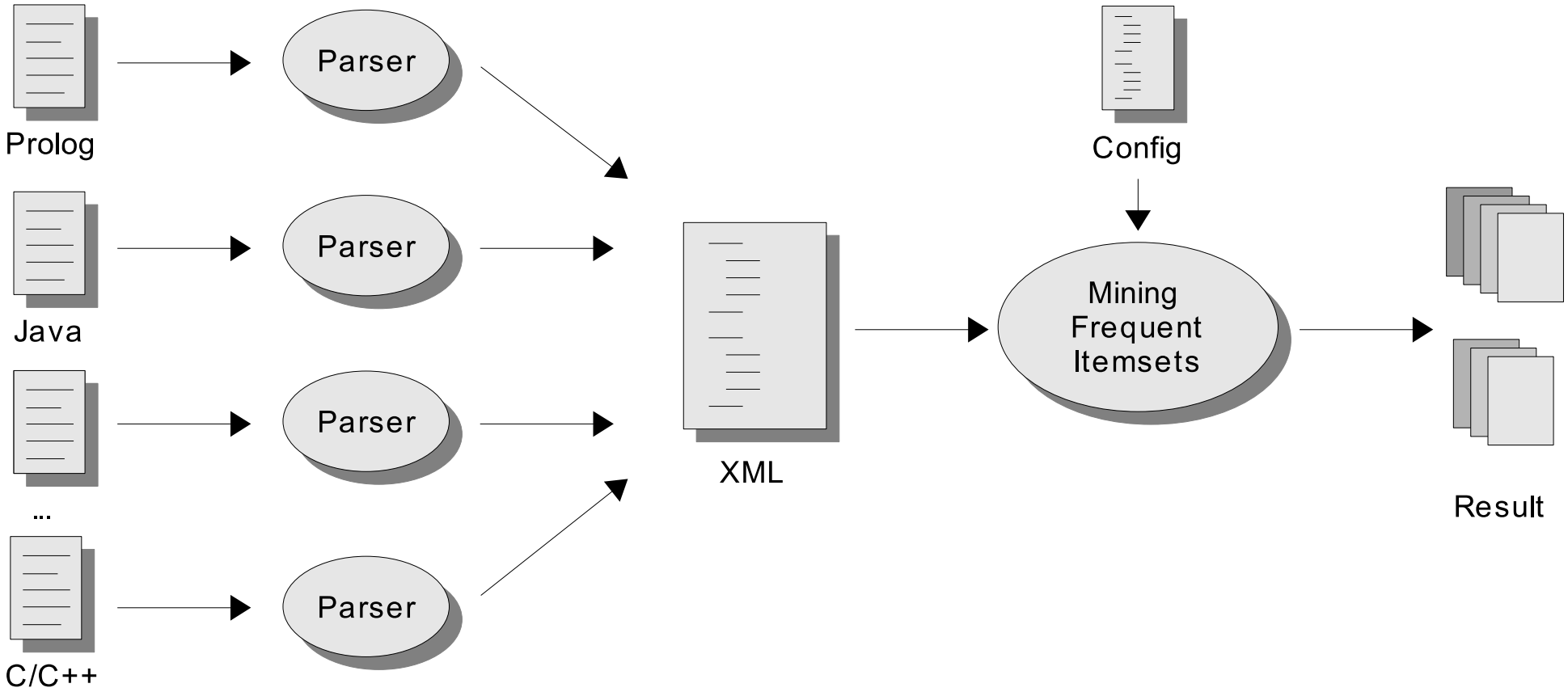
- Source code formatting and naming conventions
- Source code analysis
- Forbidden Classes and Packages
- Intelligent Comparison with master solution


apps
jSourceCodeCheck
CodeCheckVisitor
CheckVisitorConfiguration
checkComponents
*
<<interface>>
CodeCheckComponent
BasicCodeCheckComponent
BlockCodeCheckComponent
DoBlockCodeCheckComponent
ForBlockCodeCheckComponent
WhileBlockCodeCheckComponent
IfBlockCodeCheckComponent
MethodOverridesWithoutSuperCodeCheckComponent
CaseWithoutBreakCheckComponent
IteratorCodeCheckComponent
IteratorAdvanceCodeCheckComponent
DoIteratorAdvanceCodeCheckComponent
ForIteratorAdvanceCodeCheckComponent
WhileIteratorAdvanceCodeCheckComponent
IteratorMutatorCodeCheckComponent
DoIteratorMutatorCodeCheckComponent
ForIteratorMutatorCodeCheckComponent
WhileIteratorMutatorCodeCheckComponent

Source code transformation to XML (JaML)

```
<method-invocation
   name="sort" qualifier=""
   argument-count="2"
   argument-0="l" argument-1="m"
   signature= "sort(int, int)"
   return-type-ref="void">
   <literal-expression
      literal="sort" type-ref="">
      <identifier>sort</identifier>
   </literal-expression>
   <symbol kind="left-parenthesis">
      ( </symbol>
   <literal-expression
      literal="l" type-ref="int">
      <identifier>l</identifier>
   </literal-expression>
   <symbol kind="comma">
      , </symbol>
   <literal-expression
      literal="m" type-ref="int">
      <identifier>m</identifier>
   </literal-expression>
   <symbol kind="right-parenthesis">
      ) </symbol>
</method-invocation>
```

Clone Detection



Metrics

- Coupling of classes, subsystems
- Locality

Optimization

O1: avoid excessive instance creation

O2: avoid iterators

O3: use pooling

O4: use caching

O5: use string comparisons carefully

O6: avoid excessive exception throwing

O7: avoid excessive local variable declaration

O8: avoid long parameter lists

O9: avoid irrelevant or convenient method calls:

O10: avoid type casts

O11: avoid unnecessary use of standard collection classes:

O12: avoid `instanceof` for equality checks

O13: avoid recursive implementations:

O14: check the sequence in switch-statements and if-chains

O15: use Java-style listeners:

O16: avoid reflection for class information retrieval:

HotSpot JVM Sun

source code	files	LOC	v1.00 [s]	v1.06 [s]	v2.01 [s]		
					JDK	BCEL	CR
parsetree	1	3194	11.320	4.690	2.530	2.540	2.330
jamlvisitor	1	1091	9.460	3.200	1.720	1.840	1.660
jaml2visitor	1	2453	53.000	5.740	4.280	4.420	4.090
testsuite	416	2995	687.580	476.310	315.000	333.000	323.800
treeview	82	17414	92.100	15.120	10.190	9.190	8.750

JET native compiler

source code	v2.01 native [s]		
	JDK	BCEL	CR
parsetree	1.828	1.500	1.564
jamlvisitor	0.789	0.765	0.738
jaml2visitor	2.406	2.624	2.579
testsuite	151.399	153.758	156.347
treeview	9.028	7.994	6.744

Summary

- Architecture - OOP, Design Patterns
- Configuration - multiple
- Applications - just a few
- Performance - experience
-
- open source