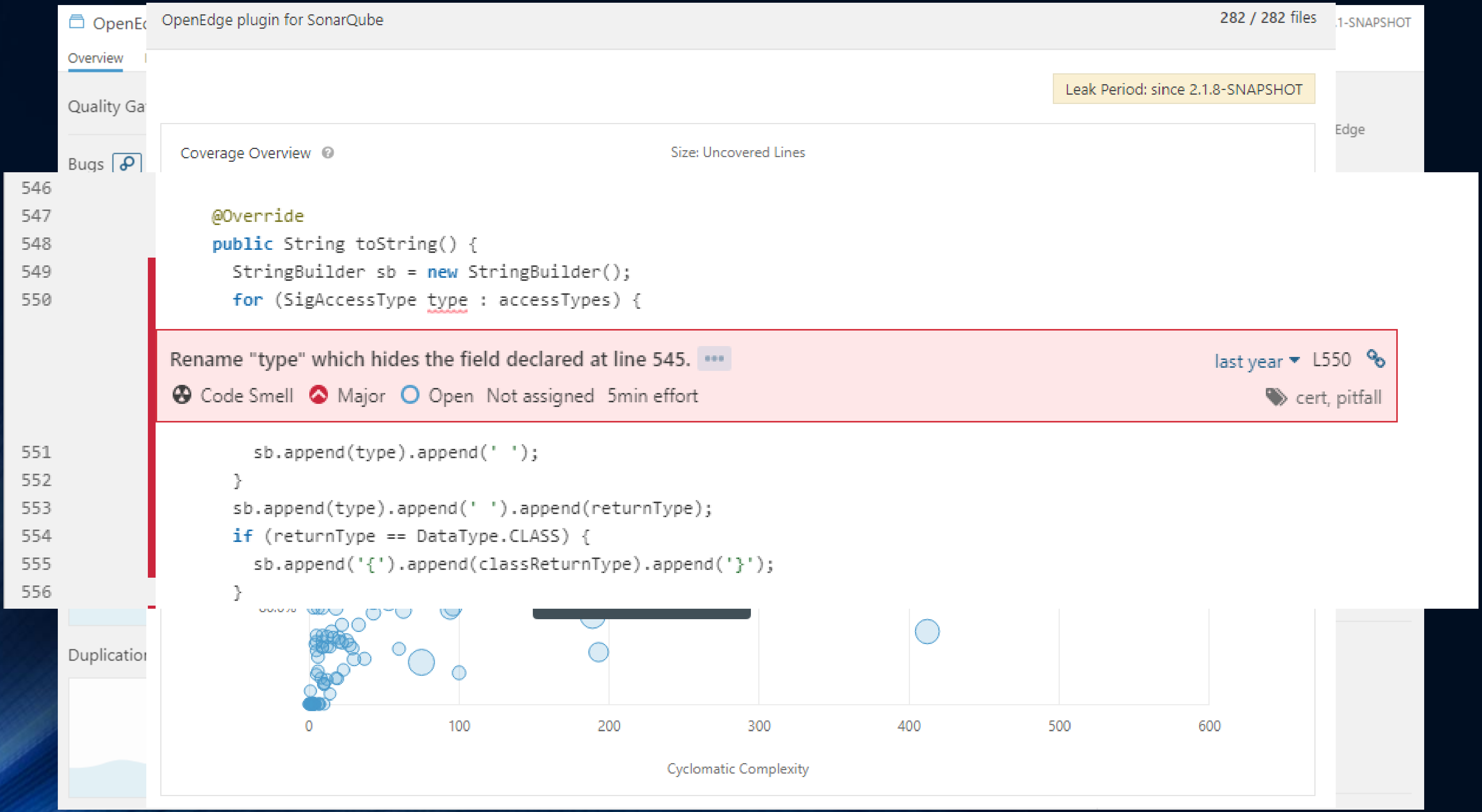
Custom rules development with SonarQube

GILLES QUERRET – RIVERSIDE SOFTWARE

About the speaker

- Pronounced \ʒil.ke.ʁe\
- Started Riverside Software in 2007
- Continuous integration in OpenEdge
- Multiple open source projects

Why this session ?



```
  @Override
  public String toString() {
    StringBuilder sb = new StringBuilder();
    for (SigAccessType type : accessTypes) {
```

Rename "type" which hides the field declared at line 545. ••• last year ▾ L550

Code Smell Major Open Not assigned 5min effort cert, pitfall

```
      sb.append(type).append(' ');
    }
    sb.append(type).append(' ').append(returnType);
    if (returnType == DataType.CLASS) {
      sb.append('{').append(classReturnType).append('}');
    }
```

Everything is a plugin

- SonarQube is an extensible platform
  - Language support provided as plugins
  - Additional rules also provided as plugins
  - Web UI can be extended by plugins
- OpenEdge plugin available under an open-source license
- Rules available under a commercial model

Requirements – Java everywhere

- JDK 8
- Maven 3.5
- Eclipse or IntelliJ recommended
- SonarQube 6.7 or 7.3
- Git

Create a new plugin

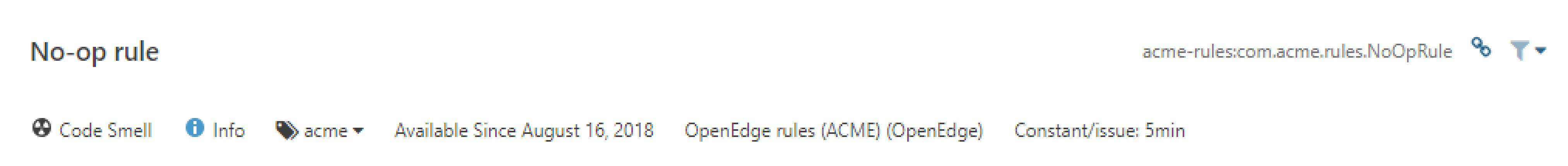
```
🔒 dev.rssw.eu ▶ gquerret ▶ home ▶ dev ▶
git clone git@github.com:Riverside-Software/rules-plugin-template.git rules

Cloning into 'rules'...
remote: Enumerating objects: 145, done.
remote: Total 145 (delta 0), reused 0 (delta 0), pack-reused 145
Receiving objects: 100% (145/145), 21.13 KiB | 21.13 MiB/s, done.
Resolving deltas: 100% (37/37), done.
```

Build the plugin

```
🔒 dev.rssw.eu ▶ gquerret ▶ home ▶ dev ▶ rules ▶
export JAVA_HOME=/opt/jdk-1.8.0_181

🔒 dev.rssw.eu ▶ gquerret ▶ home ▶ dev ▶ rules ▶
/opt/maven-3.5.4/bin/mvn clean package

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building ACME rules for OpenEdge plugin 1.1.0
[INFO] ------------------------------------------------------------------------

[INFO] ... SNIP ...

[INFO] Building jar: /home/dev/rules/target/acme-rules-plugin-1.1.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5.144 s
```

First rule – The code

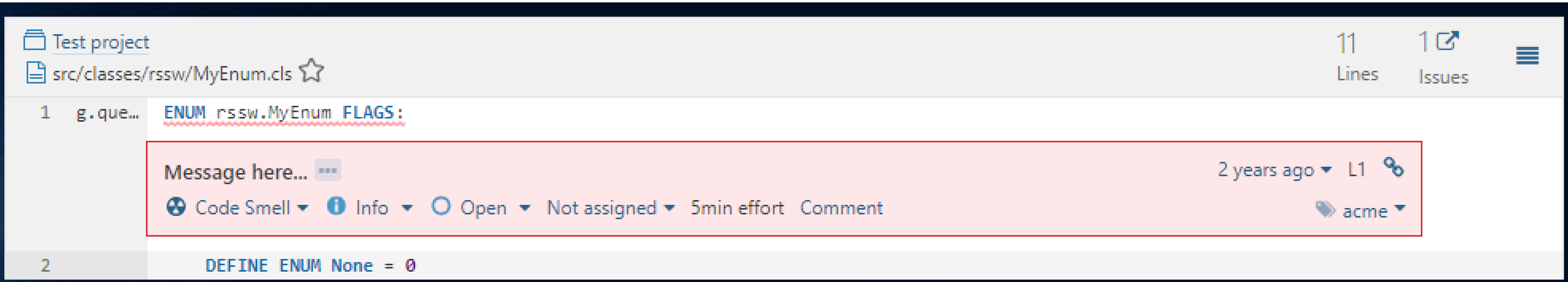
```
@Rule(priority = Priority.INFO, name = "No-op rule", tags = {"acme"}
@SqaleConstantRemediation(value = "5min")
public class NoOpRule extends OpenEdgeProparseCheck
```



No-op rule

acme-rules:com.acme.rules.NoOpRule

Code Smell   Info   acme   Available Since August 16, 2018   OpenEdge rules (ACME) (OpenEdge)   Constant/issue: 5min



Test project

src/classes/rssw/MyEnum.cls

11 Lines   1 Issues

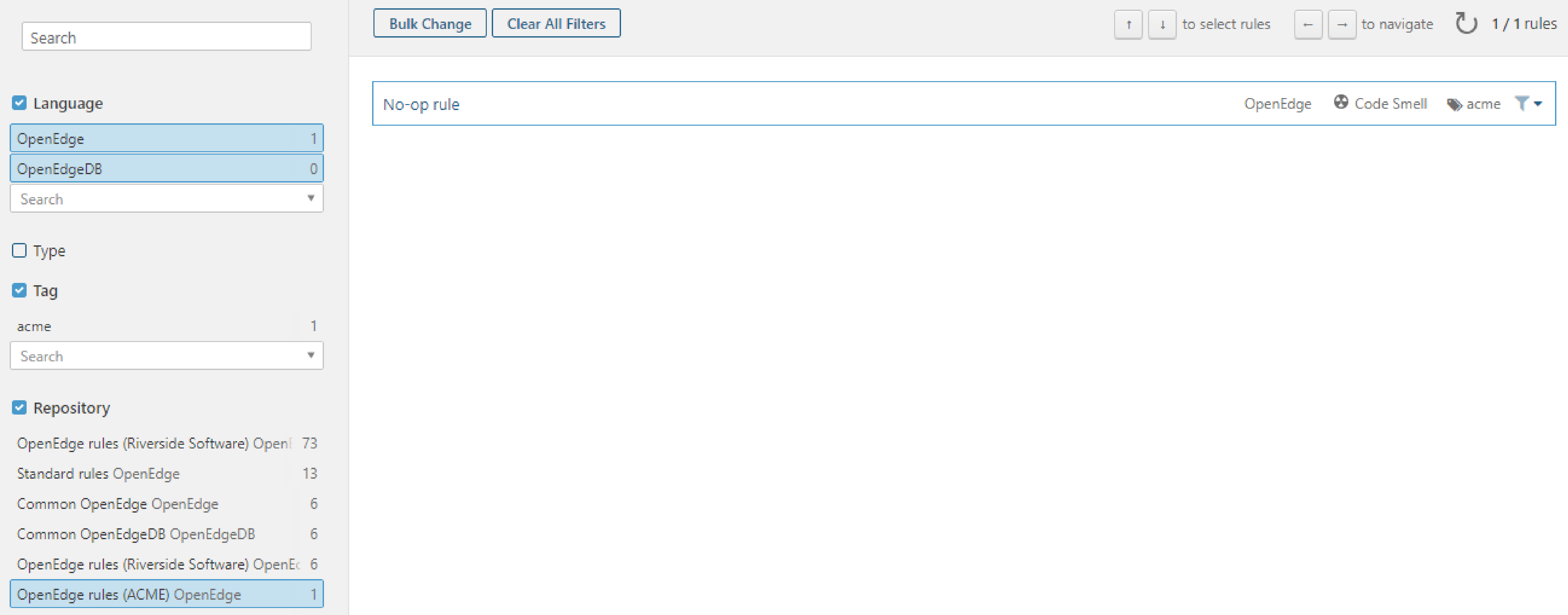
```
g.que…  ENUM rssw.MyEnum FLAGS:
```

Message here...

Code Smell   Info   Open   Not assigned   5min effort   Comment   2 years ago   L1   acme

```
            DEFINE ENUM None = 0
```

Rules registration



No-op rule — OpenEdge — Code Smell — acme

☑ Language

OpenEdge	1
OpenEdgeDB	0

Search ▾

□ Type

☑ Tag

acme	1

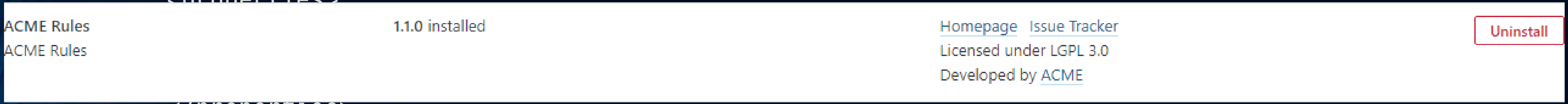
Search ▾

☑ Repository

OpenEdge rules (Riverside Software) OpenI	73
Standard rules OpenEdge	13
Common OpenEdge OpenEdge	6
Common OpenEdgeDB OpenEdgeDB	6
OpenEdge rules (Riverside Software) OpenEc	6
OpenEdge rules (ACME) OpenEdge	1

pom.xml

```
<project ...>
 ...
 <groupId>eu.rssw.sonar.openedge</groupId>
 <artifactId>acme-rules-plugin</artifactId>
 <version>1.1.0</version>
 <packaging>sonar-plugin</packaging>
 ...
 <properties>
```



ACME Rules
ACME Rules

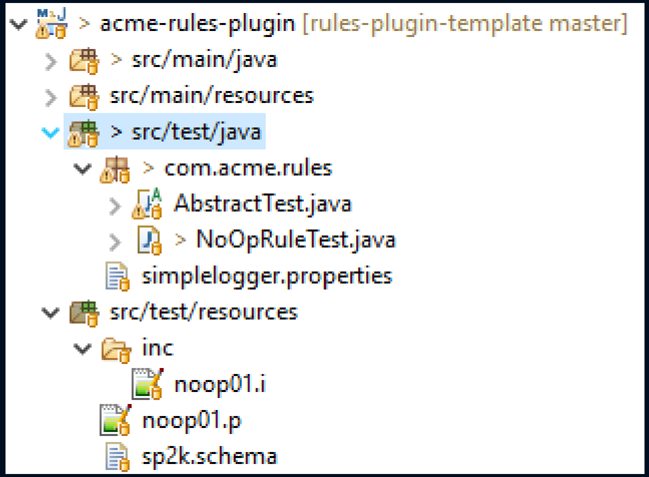
1.1.0 installed

Homepage Issue Tracker
Licensed under LGPL 3.0
Developed by ACME

```
 </properties>
 ...
 <dependency>
  <groupId>eu.rssw.openedge.checks</groupId>
  <artifactId>openedge-checks</artifactId>
  <version>2.2.0</version>
  <scope>provided</scope>
 </dependency>
 ...
```

Unit tests

```
public class NoOpRuleTest extends AbstractTest {

  @Test
  public void test1() {
    InputFile inputFile = getInputFile("noop01.p");
    NoOpRule rule = new NoOpRule();
    rule.setContext(ruleKey, context, null);
    rule.initialize();
    rule.sensorExecute(inputFile, getParseUnit(inputFile));

    // This line has to be updated to match the rule's logic
    Assert.assertEquals(context.allIssues().size(), 1);
  }

}
```

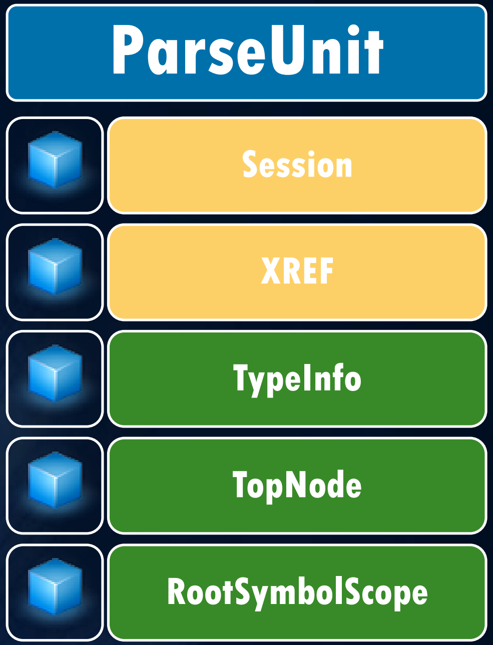


```
> acme-rules-plugin [rules-plugin-template master]
  > src/main/java
  src/main/resources
  > src/test/java
    > com.acme.rules
      AbstractTest.java
      > NoOpRuleTest.java
    simplelogger.properties
  src/test/resources
    inc
      noop01.i
    noop01.p
    sp2k.schema
```

Unit tests – Better Context

- Everything in AbstractTest class
  - Add database connections + aliases (default: sports2000, no alias)
  - Add PROPATH entries (default: src/test/resources + inc/ subdirectory)
  - Add rcode directory (default: no rcode)
  - Add XREF files (default: no XREF)
- Include PCT build is not easy
- Don’t rely on anything out of the repository

OpenEdgeProparseCheck object



ParseUnit

Session

XREF

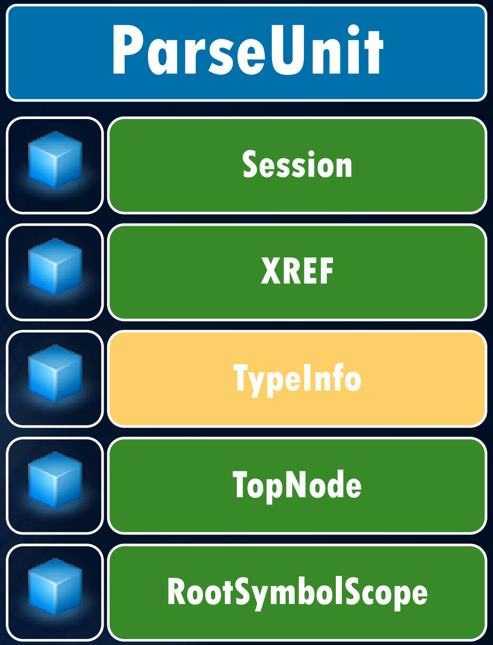
TypeInfo

TopNode

RootSymbolScope

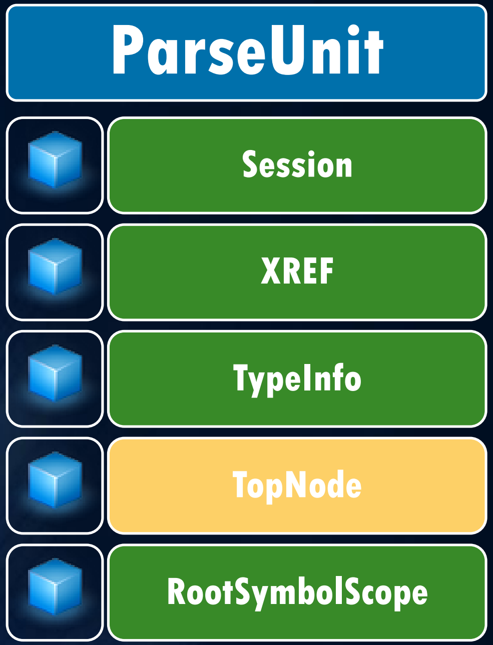
- RefactorSession
  - Schema + PROPATH + various utility methods
- XREF
  - org.w3c.dom.Document object
  - Use XPath to navigate objects

OpenEdgeProparseCheck object



- eu.rssw.pct.TypeInfo object for OOABL
  - Null if rcode is not available
  - Type name, parent object, interfaces, …
  - Flags
  - Variables, properties, events
  - Tables and buffers
  - Methods

OpenEdgeProparseCheck object

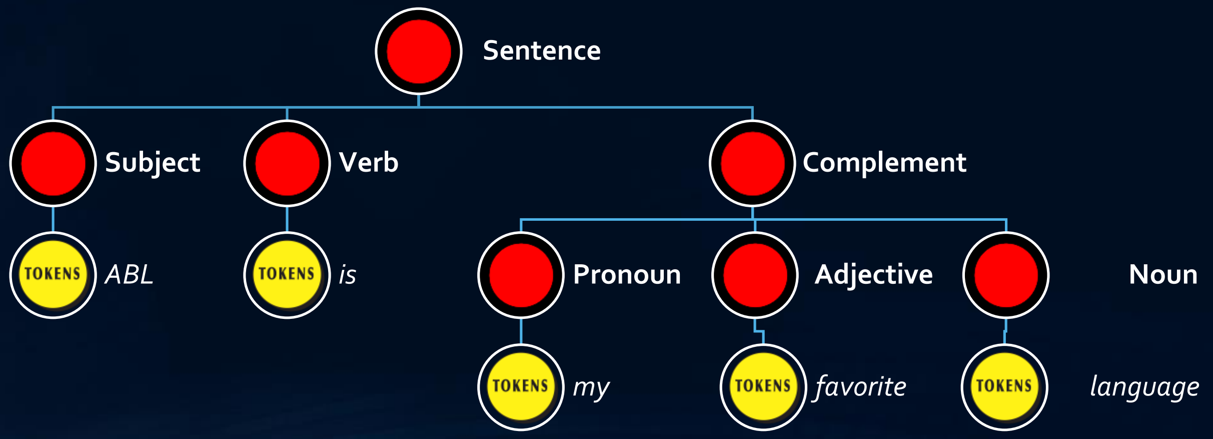


- org.prorefactor.core.ProgramRootNode object
  - Extends org.prorefactore.core.JPNode
- Tree structure representation of any source code
- First layer of the code parser

ANTLR (Another Tool for Language Recognition)

« ABL is my favorite language. »

ABL	is	my	favorite	language
Subject	verb	pronoun	adjective	noun



ANTLR – Lexer and preprocessor

```
&SCOPED-DEFINE NEW NEW

{ lib/defvar.i suffix=xx }
DEF var xx{&xyz}xx as // comment
   char /* comment 1 /* and 2 */ back
to 1 */
&UNDEFINE NEW

&IF DEFINED(NEW) &THEN
  def var yy as char no-undo.
&ENDIF
```

Preprocessor variable definition

Define suffix variable then include every token in include file

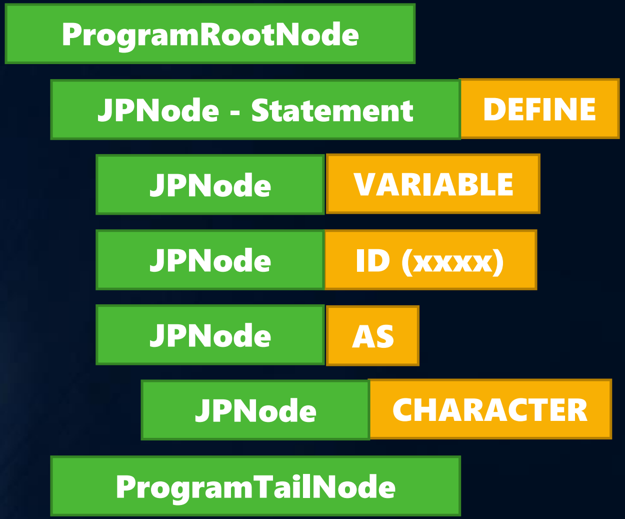
DEFINE **VARIABLE** **ID (xxxx)** **AS** **COMMENT** **CHARACTER** **COMMENT**

Undefine variable

Evaluate expression, then skip tokens until &ENDIF

- Each token is a org.prorefactor.proparse.antlr4.ProToken object
- Has a text, and begin / end position (file number, line, column)
- Has a node type and a channel ID

ANTLR – Parser



ProgramRootNode

JPNode - Statement DEFINE

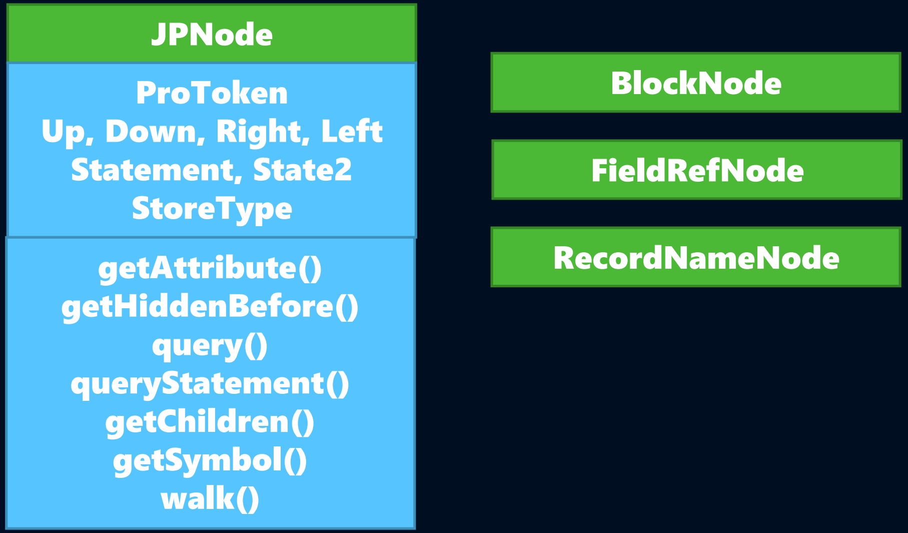
JPNode VARIABLE

JPNode ID (xxxxx)

JPNode AS

JPNode CHARACTER

ProgramTailNode



JPNode

ProToken
Up, Down, Right, Left
Statement, State2
StoreType

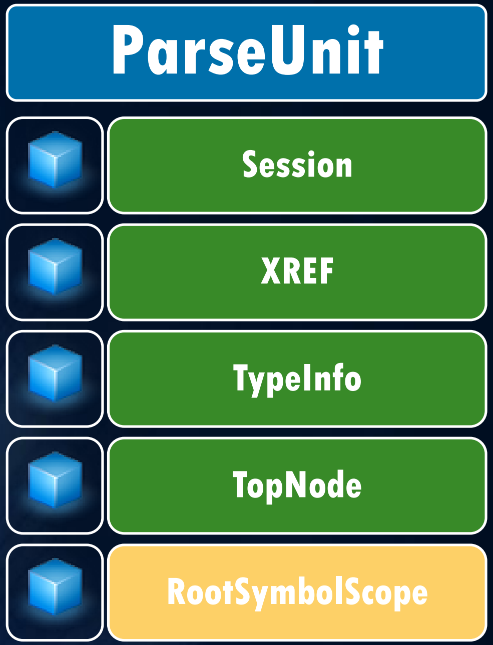
getAttribute()
getHiddenBefore()
query()
queryStatement()
getChildren()
getSymbol()
walk()

BlockNode

FieldRefNode

RecordNameNode

OpenEdgeProparseCheck object



- Tree parsers are executed on the parse tree
  - Technology being replaced by ANTLR4 visitors
- Generates more advanced information on procedures and classes
  - Procedure / Method blocks
  - Associated buffers
  - Associated parameters and variables + references to those variables
  - Calls to external procedures / functions
  - Subscopes

Example rule 1 – Backslash in string

```
@Override
public void execute(InputFile file, ParseUnit unit) {
  for (JPNode node : unit.getTopNode().query(ABLNodeType.QSTRING)) {
    visitQuotedStringNode(file, node);
  }
}

private void visitQuotedStringNode(InputFile file, JPNode node) {
  String str = ProgressString.dequote(node.getText());
  int startPos = -1;
  List<Integer> lst = new ArrayList<>();
  while ((startPos = backslashMatcher.indexIn(str, startPos + 1)) != -1) {
    // Discard backslash following a tilde
    if ((startPos > 0) && (str.charAt(startPos - 1) == '~’))
      continue;
    // Discard backslash preceding a curly brace (handled by another rule)
    if ((startPos < str.length() - 1) && (str.charAt(startPos + 1) == '{‘))
      continue;
    // Otherwise, log issue
    lst.add(startPos);
  }
  if (!lst.isEmpty()) {
    reportIssue(file, node,
         MessageFormat.format("Backslash found at position(s) {0}", Joiner.on(", ").skipNulls().join(lst)), true);
  }
}
```

Example rule 2 – No indexes

```
@Override
public void execute(InputFile file, ParseTree tree) {
  DumpFileVisitor visitor = new DumpFileVisitor("");
  tree.accept(visitor);
  for (Table tbl : visitor.getDatabase().getTables()) {
    if (tbl.getIndexes().isEmpty()) {
      reportIssue(file, tbl.getFirstLine(), "Table " + tbl.getName() + " has no index");
    }
  }
}
```

Example rule 3 – Unused variable

```
@Override
public void execute(InputFile file, ParseTree tree) {
  parseSymbolScope(file, unit.getRootScope());
  for (TreeParserSymbolScope scope : unit.getRootScope().getChildScopesDeep()) {
    parseSymbolScope(file, scope);
  }
}

private void parseSymbolScope(InputFile file, TreeParserSymbolScope scope) {
  for (Variable var : scope.getVariables()) {
    JPNode node = var.getDefineNode();
    if (node == null) node = var.getIndirectDefineIdNode();
    if ((var.getNumReads() == 0) && (var.getNumWrites() == 0))
      reportIssue(file, node, "Unused variable " + var.getName());
  }
}
```

SonarLint

- Custom rules can be executed on SonarQube and on SonarLint
- Some objects are not available in SonarLint:
  - XREF
  - Not all TypeInfo objects (depends on rcode)
- Issues reported on include files are not visible (for obvious reasons)