Source code analysis with SonarSource

Gilles QUERRET ● Riverside Software ● US PUG Challenge 2013

What’s source code analysis?

- Part of QA
- Extracts metrics from source code, and reports violations against rules
- Various types of rules :
  - Coding standards
  - Bugs
  - …

Why would you want to analyze source code?

- Refactor source code
- Performance problems
- Portability problems
- Coding standards
- Save time and money, as it’s faster than manual review

Existing tools for OpenEdge

- Limited number of tools :
  - Proparse analyses source code
  - Prolint detects pattern in Proparse output, and store the result
- Prolint output is quite minimalist

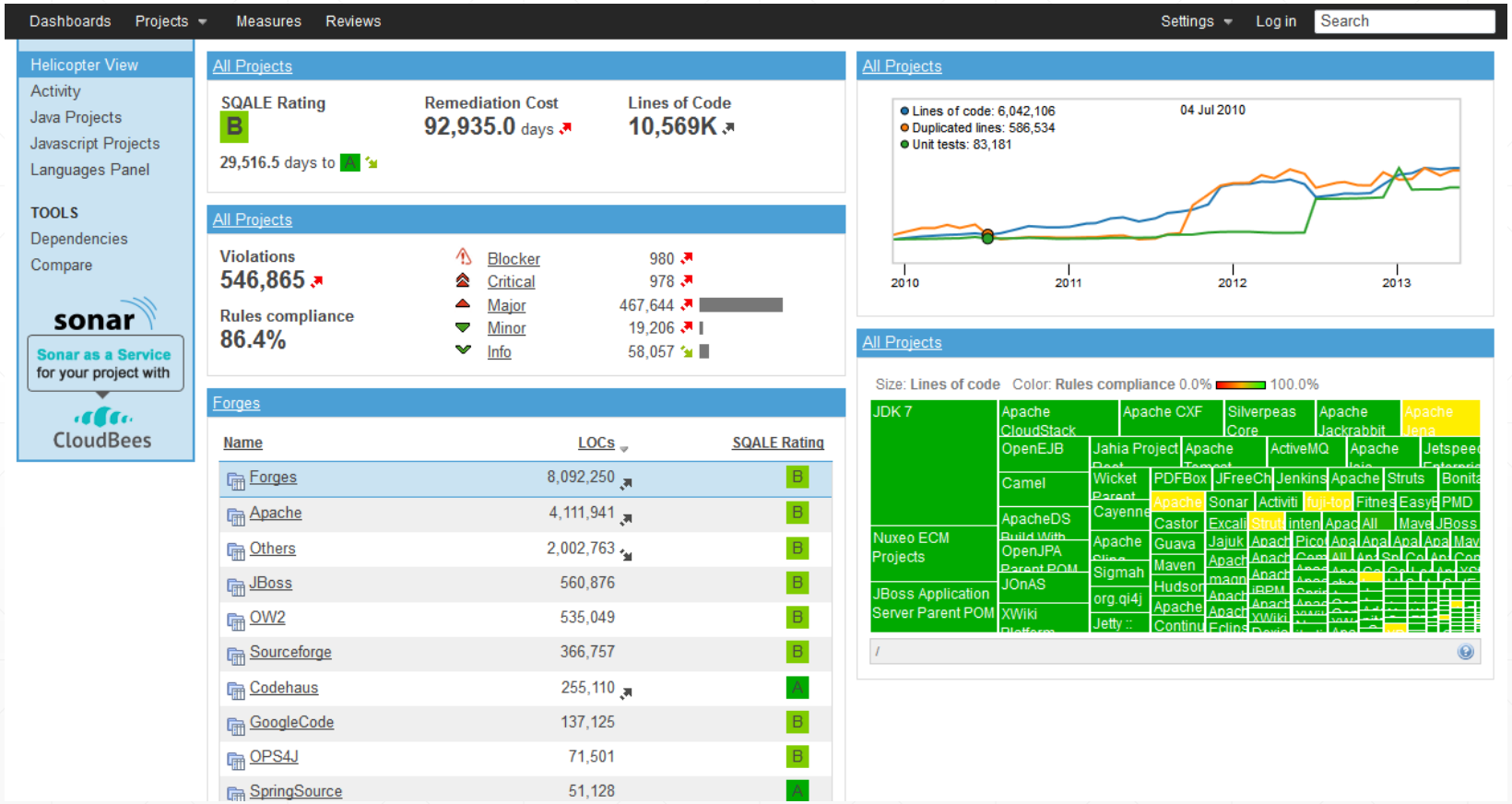
Compared to what's available in Java

- Bytecode analyzer : FindBugs, ObjectWeb ASM, …
- Static code analysis : Checkstyle, PMD, …
- Code coverage : JaCoCo, …

- And this is only for the opensource part.
- Many commercial products

And SonarSource to bind them all !

- SonarSource gather the results in a single database, and display them in a very nice web UI.



How does it work ?

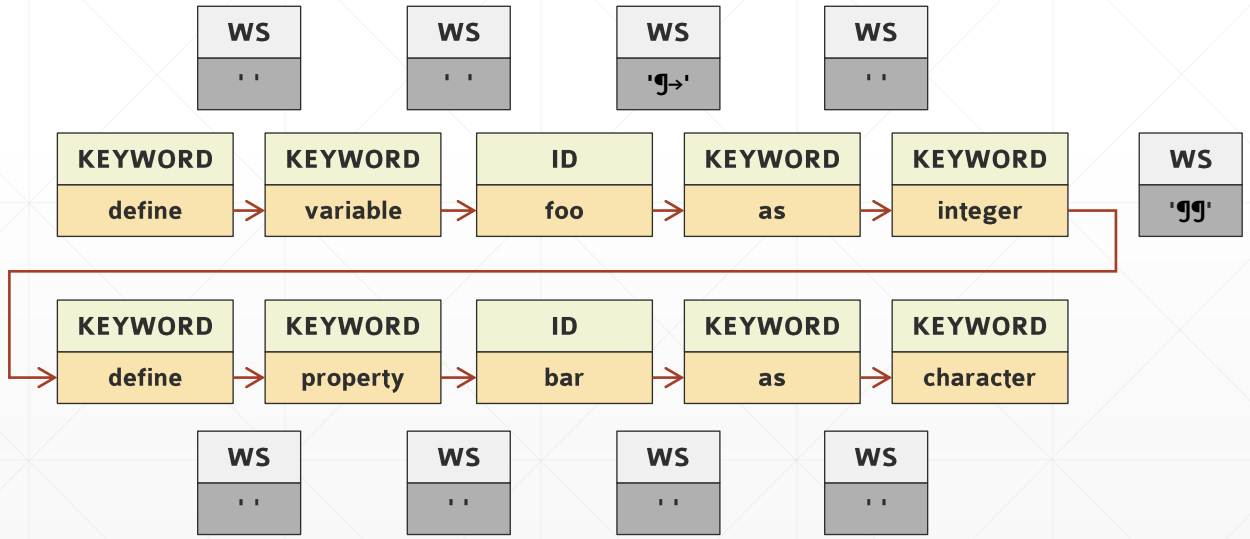
- Static code analysis usually rely on Lexer and Parsers
- Lexer converts a sequence of characters into a sequence of tokens
- Parser converts a sequence of tokens into a syntax tree

- Let’s see a simple example :

Lexer

- `KEYWORD : 'define' | 'variable' | 'property' | 'as' | 'character' | 'integer'`
- `WS : (' ' | '\t' | '\n')* -> HIDDEN`
- `ID : [a-zA-z]*`
- `EOS : '.'`

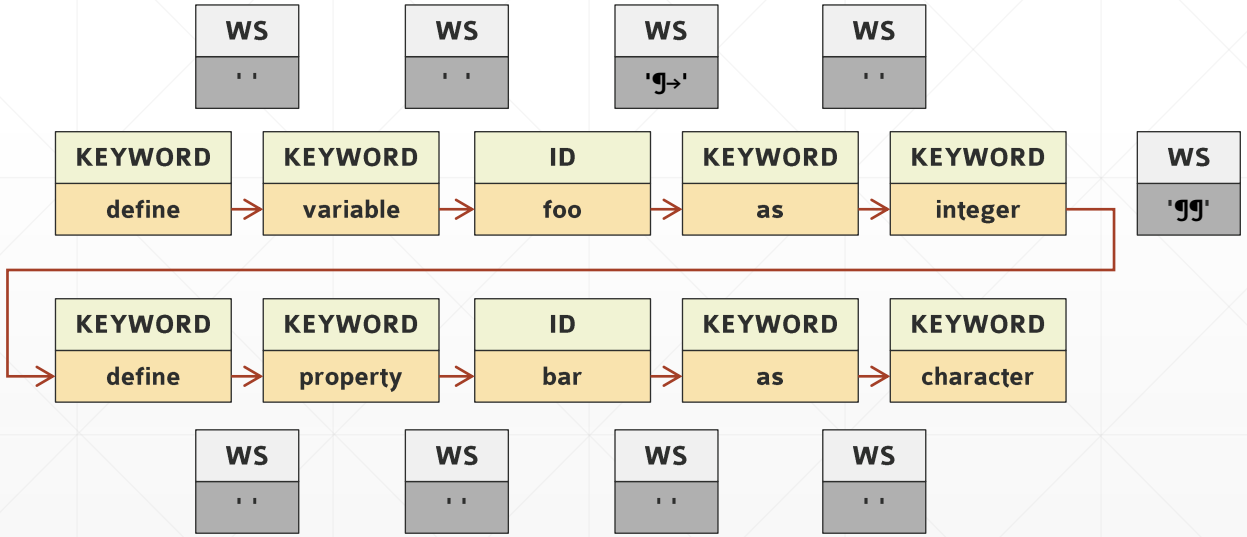
```
define␣variable␣␣foo¶
 → as␣integer.¶
¶
define␣property␣bar␣as␣character.
```

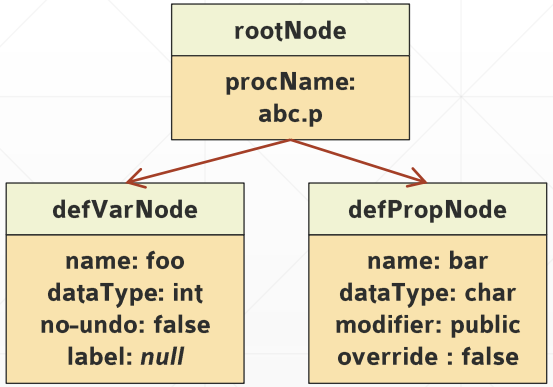


Parser

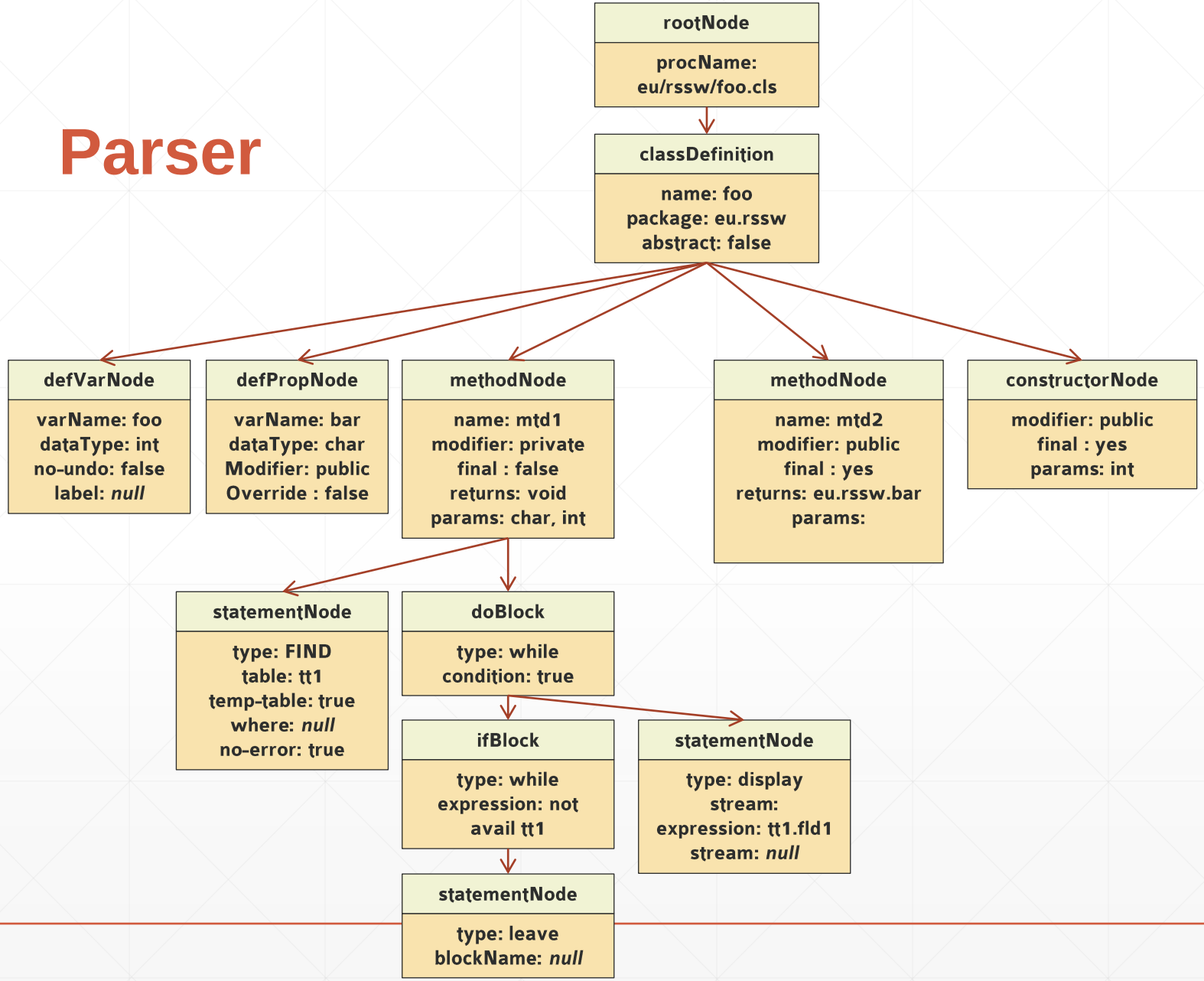
- defVar: KW_DEFINE KW_VARIABLE ID KW_AS DATATYPE KW_NO_UNDO? (KW_LABEL QUOTED_STRING)?

- defProp: KW_DEFINE (KW_PUBLIC | KW_PROTECTED | KW_PRIVATE)?
  (KW_STATIC | KW_ABSTRACT)? KW_OVERRIDE? KW_PROPERTY ID
  KW_AS (DATATYPE | CLASSNAME)





Parser


rootNode
procName: eu/rssw/foo.cls
classDefinition
name: foo
package: eu.rssw
abstract: false
defVarNode
varName: foo
dataType: int
no-undo: false
label: null
defPropNode
varName: bar
dataType: char
Modifier: public
Override : false
methodNode
name: mtd1
modifier: private
final : false
returns: void
params: char, int
methodNode
name: mtd2
modifier: public
final : yes
returns: eu.rssw.bar
params:
constructorNode
modifier: public
final : yes
params: int
statementNode
type: FIND
table: tt1
temp-table: true
where: null
no-error: true
doBlock
type: while
condition: true
ifBlock
type: while
expression: not avail tt1
statementNode
type: display
stream:
expression: tt1.fld1
stream: null
statementNode
type: leave
blockName: null

Only parsing source code ?

- Parsing can be done :
  - On profiler output (data structures for execution times, code coverage, …)
  - On DF files to know the database structure
  - On XREF files

Disclaimer

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

OpenEdge extension for SonarSource

- SonarRunner is the batch process to analyse local source code, and push the results to the remote SonarSource instance
- An extension is made of several « sensors », each sensor analyzing a specific set of data
- Current sensors :
  - Rcode sensor
  - Dump files sensor
  - Code duplication sensor
  - Code coverage sensor
  - Source code sensor

Rcode sensor

- Extract information from rcode, and generate metrics :
  - Coupling between classes
  - Circular references between packages
  - Number of public / private methods
  - Common OO violations (using variables in classes, constructor for utility classes, …)

Dump file sensor

- Extract information from DF files:
  - Allow working with a known database structure when analyzing source code
  - Violations on table/field/index names, useless indexes, common SQL problems

Code duplication sensor

- Copy / paste is widely used during software development
- High code duplication means you should refactor your code
- Duplicated code usually mean that a fix is not applied everywhere

- Doesn’t do the refactor for you !

Code coverage sensor

- Use profiler output to exactly know which lines of code have been tested
- Multiple profiler files can be parsed and aggregated
- Let you know exactly which part of your code is being tested (or not tested)

Source code sensor

- Generate syntax tree from every class / procedure
- Then execute a set of lint rules, and report violations
- Source code uploaded to SonarSource server in order to display it with annotations
- Lint rules can be written by users

Writing your own rules

- Lint rules will have access either to the source tree, or to a fully built object
- The source tree know the exact position of your tokens
- Sonar entry point to report violations

- Rules have to be written in Java
- Implementation time is nothing compared to design time !

Lint rule example

```
// info is the object representing a class definition
if (!info.isAbstract() && !info.isInterface()) {
  for (MethodElement method : info.getMethods()) {
    if (method.isPublic() && !method.isStatic()) {
      boolean isImplementation = false;
      for (TypeInformation interfaces : info.getInterfaces()) {
        if (isImplementation(method, classInfos.get(interfaces.getName())))
            isImplementation = true;
      }
      if (isImplementation)
        System.out.println(" --> Method " + method.getName() + " is an implementation");
      else
        System.out.println(" --> Method " + method.getName() + " isn't an implementation");
    }
  }
```

Questions ?

Reference ?

- Sonar Source : http://www.sonarsource.com
- Sonar OE plugin demo site : http://sonar.riverside-software.fr

- Riverside Software : http://riverside-software.fr
- Gilles QUERRET : g.querret@riverside-software.fr