ABL source code analysis with SonarQube

Gilles QUERRET ● Riverside Software

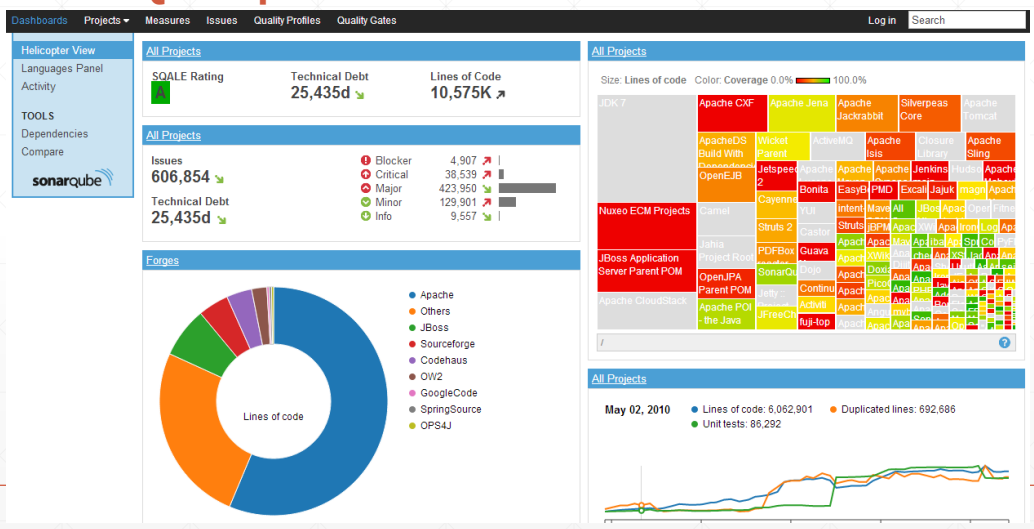
SonarQube

- SonarQube is a platform to manage code quality
- Free, open source, LGPL, web-based portal
- Backed by SonarSource
- Aggregates well-know tools from the Java world
  - Static analysis (PMD, FindBugs, Checkstyle)
  - Duplicate code (Squid, CPD)
  - Code coverage (Cobertura, JaCoCo)

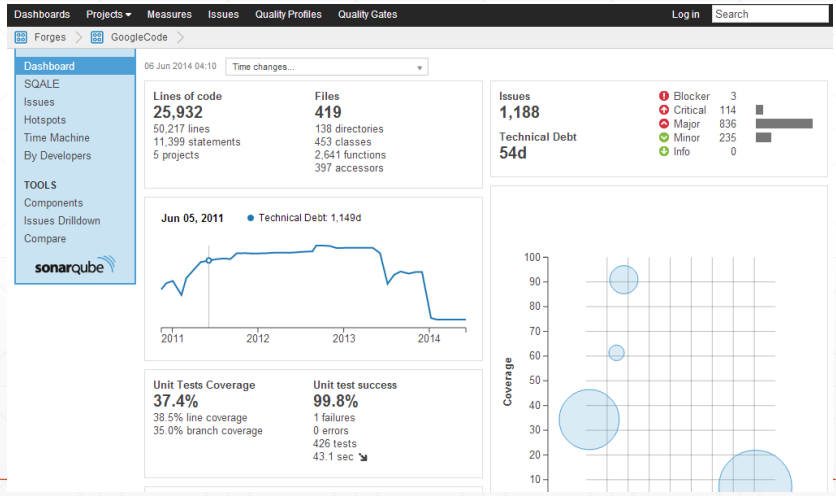
SonarQube plugins

- SonarQube is extensible, providing entry points to manage new languages
- Both open and closed source plugins (C#, Cobol, …)
- Extensions are not only for languages, but also for visualization
- And also rules which are applied on syntax trees (see next slides)

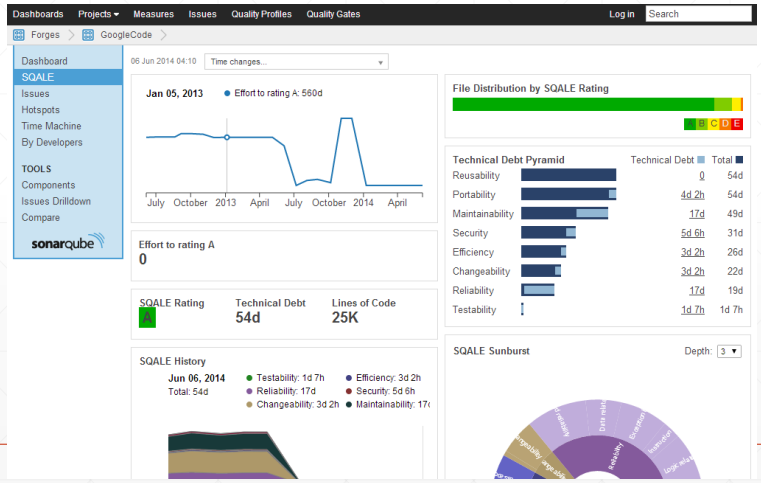
SonarQube portal



Project portal



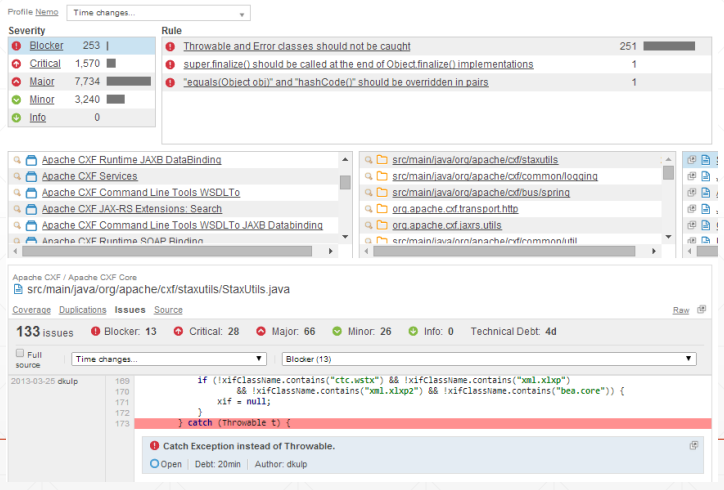
Project portal



Technical debt

- Term coined by Ward Cunningham in 1992
- Technical debt is a metaphor around the time it takes to work with low quality code :
  - If technical debt is high, you'll pay high interest (in terms of time spent) each time you work on the project
  - But you can pay down the principal by refactoring code
  - You'll increase the debt if you add quick and dirty code
- Do you manager care about code quality ?
  - If you want to catch their attention, just let them know they have to pay the bill

Drill down into the code



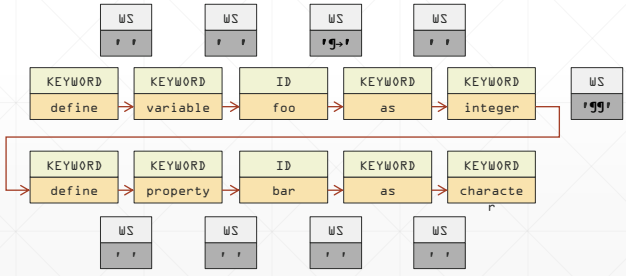
What about ABL ?

- Free, open-source, LGPL plugin provided by Riverside Software
- Plugins (and source code) can be downloaded from :
  - https://bitbucket.org/gquerret/openedge-plugin-for-sonarqube
  - https://bitbucket.org/gquerret/openedge-db-plugin-for-sonarqube
- They include parsers, rules management and issue reporting
- Sample rules also available
- Additional rules can be bought from Riverside Software
- Or you can implement them !

Lexer and parsers

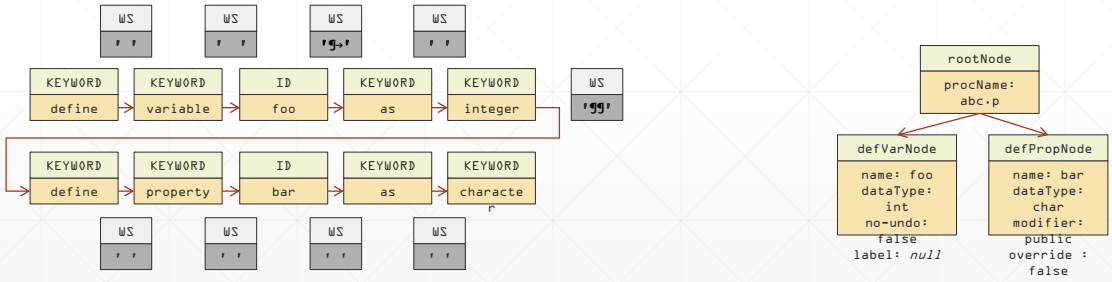
- `KEYWORD : 'define' | 'variable' | 'property' | 'as' | 'character' | 'integer'`
- `WS : (' ' | '\t' | '\n')* -> HIDDEN`
- `ID : [a-zA-z]*`
- `EOS : '.'`

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

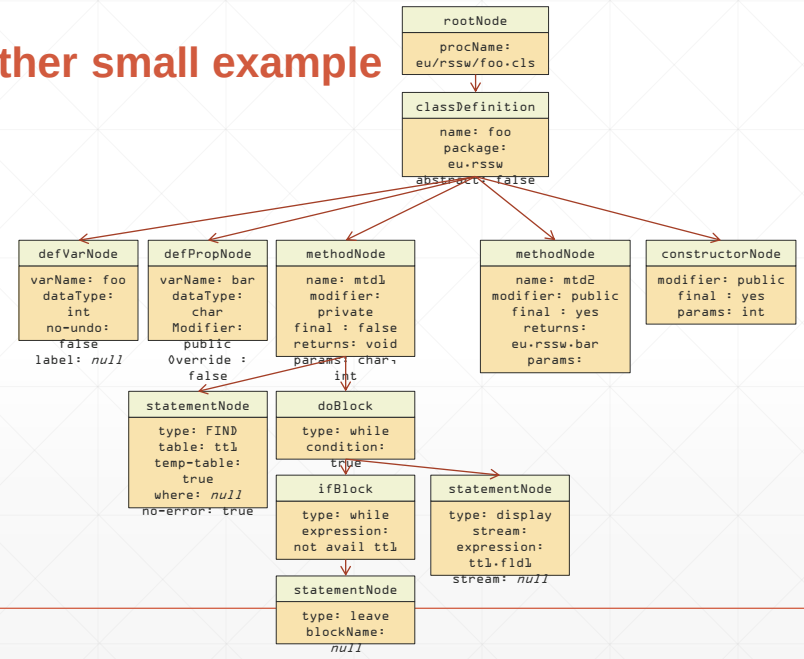


Lexer and parsers

- 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)



Another small example



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

How to install ?

- Download SonarQube 4.2 or 4.3 from http://dist.codehaus.org/sonar/
- Extract in any directory
- Download OpenEdge plugins from BitBucket to `extensions/plugins` directory
- Execute startSonar.sh !

- SonarQube uses by default an H2 database, which is OK only for tests
- Please refer to Sonar documentation if you want to use MySQL, MS-SQL or Oracle

How to analyze code ?

- Analysis requires build output from PCT
- Still not using PCT ? Go immediatly to http://code.google.com/p/pct and switch your build system to PCT !
- PCT is another open-source plugin for Ant (also from Riverside Software)
- Probably the most widely used build tool
- Perfect in combination with continuous integration

How to analyze code ?

- Use the following options in PCTCompile :
  - debugListing = true
  - listing = true
  - keepXref = true
  - xmlXref = true (PCT 186+)
  - relativePaths = false

How to analyze code ?

- Then use Sonar Ant task :

```
<property name="sonar.projectKey" value="eu.rssw.product-db:DEV" />
<property name="sonar.projectName" value="Product - DB" />
<property name="sonar.projectVersion" value="1" />
<property name="sonar.sources" value="schema" />
<property name="sonar.binaries" value="build" />
<sonar:sonar />
```

- Add the following properties to connect to a remote Sonar instance :

```
<property name="sonar.host.url" value="http://..." />
<property name="sonar.jdbc.url" value="jdbc:..." />
```

What to expect ?

- Source code import in the database
- Debug listing import in the database
- Basic metrics for source code
  - Number of files, windows, includes, classes
  - Lines of code, comments, blank lines
- Transaction scopes (from listing files)
- Code duplication (using tokenized source)

What to expect ?

- XREF basic metrics
  - Entry point to report issues and metrics
- Dump files abstract syntax tree and basic rules

Next developments

- Coming soon :
  - Dependencies between procedures and classes (tangle index)
  - ABL abstract syntax tree and rules
  - Code coverage (from profiler output)
- Coming later :
  - Prolint output management
  - Progress Developer Studio integration
- What are you looking for ?
  - Please raise your hand ☺

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 !

Implementing XREF rules

```
wholeIndexExpr = xPath.compile("//Reference[@Reference-type='SEARCH' and
Detail='WHOLE-INDEX' and File-num='1']");

NodeList nodeList = (NodeList) wholeIndexExpr.evaluate(doc,
XPathConstants.NODESET);
for (int zz = 0; zz < nodeList.getLength(); zz++) {
  Element element = (Element) nodeList.item(zz);
  reportIssue(element, "WHOLE-INDEX");
}
```

Lint rule example

Verification of variable name length :

```
@Rule(priority = Priority.INFO, name = "Short variable names", description = “Verifies that
variable names are at least X characters.")
@BelongsToProfile(title = LintList.SONAR_WAY_PROFILE, priority = Priority.MINOR)
public class ShortVarNamesRule extends AbstractLintRule {
  private static final int DEFAULT_MINIMUM_LENGTH = 2;

  @RuleProperty(key = "minimumLength", defaultValue = "" + DEFAULT_MINIMUM_LENGTH)
  public int minimumLength = DEFAULT_MINIMUM_LENGTH;

  public boolean visit(VariableDeclarationStatement decl) {
    if (decl.getName().length() <= minimumLength)
      reportIssue(getFirstLine(decl), decl.getName());

    return true;
  }
}
```

Developer Studio integration

- The Sonar database can be connected from Developer Studio
- Violations are displayed in the standard editor
- Lint rules can be executed locally

Questions ?

Reference ?

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

- Riverside Software : http://riverside-software.fr
- Contact : contact@riverside-software.fr