Thomas Zierer

Combining SonarQube and PL/I V5 for Continuous Code Quality Inspection

October 12, 2017, GSE EM Working Group Meeting, Nürnberg

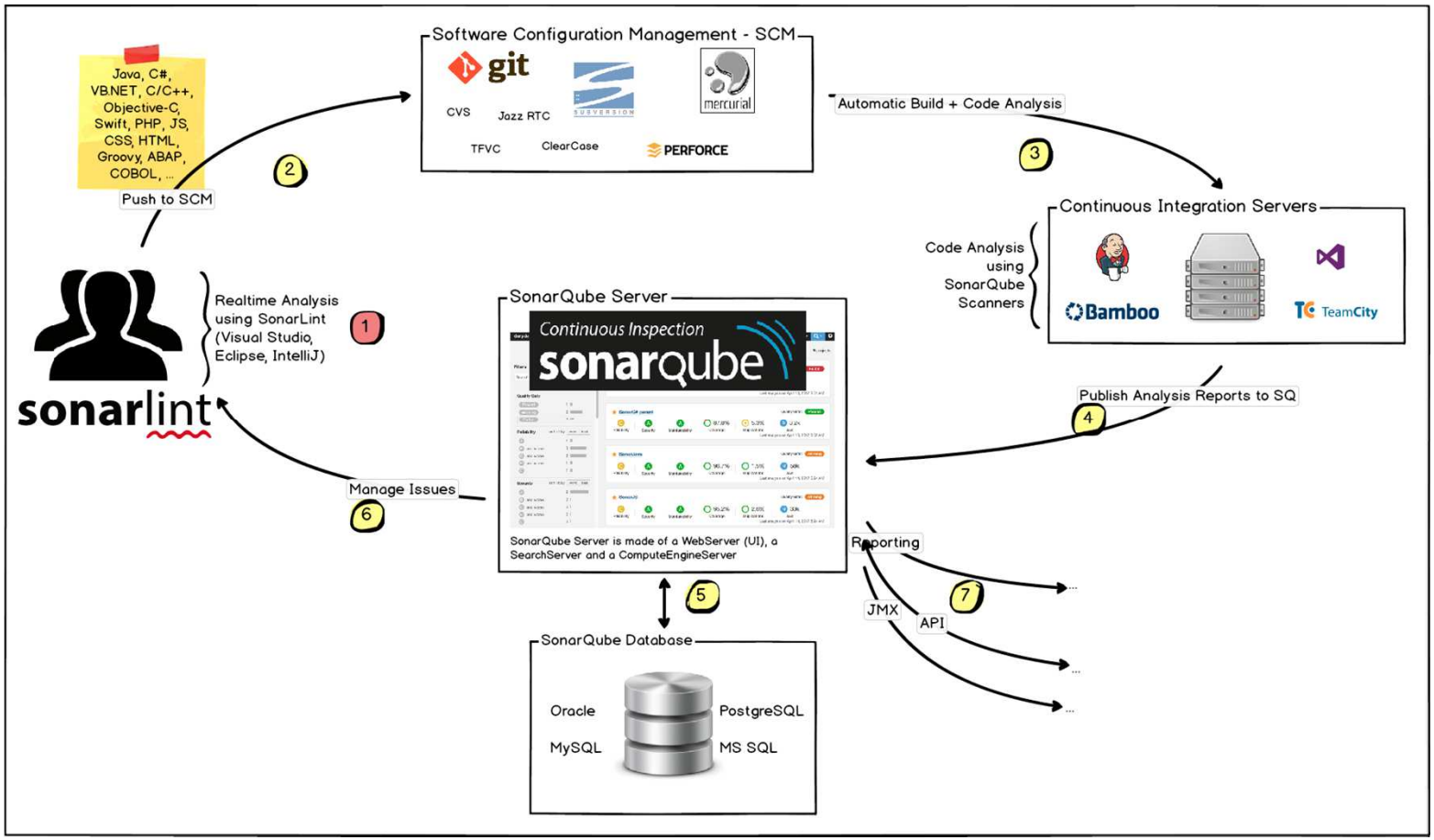
SonarQube

SonarQube is the leading Open Source product for continuous Code Qualtity Inspection

- Very popular in the Open Source Community
- Maintained by SonarSource, Geneva, Switzerland
- IDE Plugins (SonarLint) for Eclipse, IntelliJ and Netbeans for instant feedback
- Batch-scans are often integrated as ant, Maven or Gradle Tasks and run by a build server like Jenkins

SonarQube - Architecture

The batch-scan reads the sourcefiles from the SCM and feeds its results to the SonarQube-Web-UI



SonarQube

Demo

SonarQube

The commercial version supports more languages and offers enterprise features

- Open Source version:
  - Supports Java, C#, PHP, Python,…
  - Suitable for lightweight, homogenous projects
- Commercial version:
  - Enterprise Features (Reporting, LDAP-Integration, …)
  - Additional languages: C/C++, COBOL, ***PL/I***, …

So, why this session?

SonarQube and PL/I

SonarQube's PL/I support is somewhat … limited

- PL/I support requires SonarQube Enterprise + PLI/ Plugin. => 59.500 € / year
- Apart from the fact that SonarQube has only 21 rules for the PL/I language the scanner also often fails for valid Enterprise PL/I code
- As is the case with most custom PL/I scanners

Why is it so difficult to analyze PL/I code?

Most custom scanners are based on tools like lex and yacc which require the language to be strongly structured

- In 1956 famous linguist Naom Chomsky published his hierarchy of formal languages
- Java, C# etc. reside on Level-2 („Context-free“)
- This makes it comparatively easy to write scanners based on tools like lex and yacc
- Fortran and PL/I are known to reside on Level-1 („Context-sensitive“)
- This means no lex-based scanner will ever support PL/I in it‘s full strength

In 25 years I came to the conclusion:

The only reliable PL/I scanner is IBM‘s Enterprise PL/I compiler

Enterprise PL/I

The Enterprise PL/I Compiler provides 2 highly useful features: RULES and XINFO

- The RULES directive supports ~40 suboptions espescially for QA purposes
- The XINFO(XML) directive exports the compiler messages in XML format

```
<MESSAGE>
  <MSGNUMBER>IBM2418I E</MSGNUMBER>
  <MSGLINE>22</MSGLINE>
  <MSGFILE>21</MSGFILE>
  <MSGTEXT>Variable FORCE is unreferenced.</MSGTEXT>
</MESSAGE>
```

SonarQube

With its custom language support you may „teach“ SonarQube additional languages

- SonarQube is extensible through its plugin concept
- Writing your own plugin is not trivial but there are some good examples
- Documentation is provided here: https://docs.sonarqube.org/display/DEV/Extension+Guide

The Xinfo plugin

The Xinfo plugin combines Enterprise PL/I with SonarQube‘s custom language support

- The plugin parses the compilers XML output and feeds it to SonarQube‘s REST interface
- This works for every language that supports a XINFO-like directive: PL/I, COBOL, C/C++, Assembler
- Scans may run on every kind of server: Windows, Unix, z/OS.
- Restriction on z/OS: PL/I source and XML output must reside on OMVS

The Xinfo plugin: Executing scans

A simple shell or batch script executes the scan. You may even invoke it by JCL.

- Scan is invoked via SonarQube commandline client
- Successfully tested on Windows, Unix
- Prereq: Sonar Server 6.3 or higher
- On z/OS scans can be invoked using the IBM utility BPXBATCH:

```
//BPXBATCH   EXEC  PGM=BPXBATCH,
//           PARM='sh /u/y08577/sonar-scanner-2.8/bin/sonar-scanner'
//STDOUT     DD SYSOUT=*
//STDERR     DD SYSOUT=*
//STDENV     DD *
/*
```

The Xinfo plugin

Demo

Exploiting more SonarQube APIs

Integration for Debug Tool and CPD (Copy/Paste Detection)

- A simple converter transforms Debug Tool Code Coverage files into the corresponding SonarQube format.
- Split the sourcecode into tokens and SonarQube computes duplications.

Wanna try it?

The Xinfo plugin is Open Source

- Xinfo plugin source code is published under Eclipse Public License (EPL) v. 1.0
- Clone-URL on last slide
- Every comment is honestly appreciated

Whats next?

There is still a lot of work to do

- Remmediation costs per rule to compute the technical debt
- Custom measures (for example refactor programs with too many INCLUDEs)
- Automatically assign a finding to its author by questioning the SCM („blame“)

Thank you very much for your attention!

Thomas Zierer
E-Mail: thomas.zierer@muenchen-mail.de
Fork me on GitHub: https://github.com/tgmz/sonar-xinfo-plugin