Automated Static Analysis Tool research

Team Members:

Name:	StudentID:	E-mail:
Boning Gong	4367308	boninggong@yahoo.com
Clinton Cao	4349024	C.S.Cao@student.tudelft.nl
Michiel Doesburg	4343875	M.S.Doesburg@student.tudelft.nl
Sunwei Wang	4345967	S.Wang-11@student.tudelft.nl
Tim Buckers	4369459	TimBuckers@gmail.com

SE TA:
Bastiaan Reijm

TU Delft

Table of contents

1. Introduction

Before we start implementing our system, we need to do some research on automated static analysis tools. This is necessary for the development of our system. There are a few things that we would like achieve with this research:

- We hope that we could figure out which programming language to start with for our system.
- We hope to get some knowledge on what are the most frequently used automated static analysis tools(taking the popularity of the language into account).
- We hope to get some knowledge on which automated static analysis tools are available for each programming language. With this knowledge we could get an idea on how many programming languages our system will be able to support.
- We hope to get some knowledge on how each automated static analysis tools outputs the result of the analysis. With this knowledge we could get an idea on whether it is possible to unify the results of the automated static analysis tools or not.

In the rest of this document, we will be answering some questions that we've come up for this research.

In chapter 2 we will be answering the question: which automated static analysis tools are available? This question is not specific for one programming language, but in general.

In chapter 3 we will be answering the question: What are the most used automated static analysis tools? For the answer to this question, we do take the popularity of the language(s) that each automated static analysis tool is made for, into consideration.

In chapter 4 we will be answering the following question: What are the different types of automated static analysis tools and what are the differences?

In the final chapter we will answer the following question: How do automated static analysis tools output their results and how can the input be unified?

2. Which automated automated static analysis tools are available?

Because the list of available automated static analysis tools is too long to put them all in this document, we will provide a link to a wikipedia page that contains a list of automated static analysis tools that are available.
Please take a look at the references for the link [1].

3. What are the most used automated static analysis tools?

There are many tools available for automated code review. Due to the popularity of Java (according to TIOBE index [2]), it seems that ASATs is something that is more frequently used for Java. Listed below are some of the important and most often used tools:

Checkstyle: This is used more for syntactic analysis of the code like java docs, line length, naming conventions and very few design parameters like Lines of code (LOC) of a class, LOC of a method, unused parameters, variables, imports

PMD: This scans the java code and reports the potential java problems that include Optimization rules like StringBuffer, java logging rules and some overlapping rules with Checkstyle like naming conventions, unused code and other basic java design problems.

FindBugs: it uses static analysis to review the code. This is an important tool and detects critical code problems like possible null pointer exceptions, potential multithreading problems etc and some performance related problems.

The use of each of these tools would depend on individual projects size and content. However, FindBugs is suggested for each developer due to its immense power to detect high severity items. PMD is also a desirable tool. Checkstyle should be used for a clean, maintainable code [3].

Cobertura: Uses McCabe's cyclomatic code complexity algorithm to determine how complex a piece of code is. As code becomes more complex, it becomes more prone to errors. A class with high complexity might need refactoring. Or if the complexity is warranted, might need additional test coverage.

Emma: it scans the project and report how much coverage there is for the project.

4. What types of automated static analysis tools exist, and what are the differences?

There are several types of automated static analysis tools, taking the most common programming language, Java.

There are tools for:

- Finding typical bugs and ensuring code standard compliance: Sonar, Checkstyle, PMD
- Finding Class or Package Dependencies: JDepend, tattletale
- Code Coverage: Cobertura, JTest

5. How do ASATs output data? How can the output of different ASATs be unified?

ASAT	**Output format**
Checkstyle[4]	**xml**
PMD[5]	csv, emacs, html, ideaj, summaryhtml, text, textcolor, textpad, vbhtml, **xml**, xslt, yahtml
FindBugs[6]	**xml**, html, xdoc

Emma[7]	plain text, html, **xml**
Cobertura[8]	html

As we can see from the table of the previous page, 4 of the 5 ASATs output in XML. Their output can be unified into xml. Cobertura is the exception, it only outputs html. This means a translator will be required at some point. For now, it might be wise to start with XML.

6. References

[1] Link for list of ASATs: https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

[2] TIOBE index: http://www.tiobe.com/tiobe_index

[3] Enhancing java code quality through automated code review tools http://technoguider.com/2015/05/enhancing-java-code-quality-through-automated-code-review-tools/

[4] Checkstyle: http://checkstyle.sourceforge.net/cmdline.html

[5] PMD: http://pmd.sourceforge.net/pmd-5.3.2/usage/running.html

[6] FindBugs: http://findbugs.sourceforge.net/manual/running.html

[7] Emma: http://emma.sourceforge.net/

[8] Cobertura: http://cobertura.github.io/cobertura/