**CS/SE 6301 Software Analysis and Comprehension**

**Spring 2016**

INTERIM REPORT

**Common coupling detection in Java: srcML and Eclipse JDT/AST approach**

**Ronaldo Goncalves Junior**

**Sungsoo Ahn**

**Bennilyn Quek**

April 12, 2016

Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| 0.1 | Ronaldo | Preliminary Report | 4/12/2016 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction](#h.2et92p0)

[1.1 Common coupling](#h.g18wg6q8yb0w)

[1.2 Java context](#h.2v650lsorla9)

[1.3 Other types of coupling](#h.k9miocthqi1s)

[2 System](#h.lexlw6w0x76w)

[3 Baseline](#h.vz7p82a1rbe8)

[4 Eclipse JDT/AST](#h.8doa3vdb7zpv)

[5 References](#h.75t6ebu0k9cx)

## 1 Introduction

This document is the proposal for the project Common coupling detection in Java. It presents a plan to achieve the project goal, which is detect common coupling in Java source codes, calculate coupling degree, and provide common coupling information to developers and maintainers for high quality system.

### 1.1 Common coupling

Common coupling (also known as Global coupling) occurs when two modules are associated to the same global data (e.g., usage of a global variable) [1]. In other words, any changes to this data could potentially imply changes to all the modules that uses it. One additional aspect worth assessing is the set of constraints that might limit how java files (for instance) could be coupled. According to [4], there are two distinct categories of common coupling: strong and weak. Strong common coupling is when it is possible to access a global variable from any files of the system. The last one, weak common coupling, is when the global variable can only be accessed within files of the same package.

### 1.2 Java context

Some languages like Java do not have global variables [2]. In Java, all variables that are not declared as local variables are declared as fields of a class. That is, all variables are declared within the scope of either a class or a method. In Java, static fields (also known as class variables) exist independently of any instances of the class and one copy is shared among all instances; hence public static fields are used for many of the same purposes as global variables in other languages because of their similar "sharing" behavior [2].

### 1.3 Other types of coupling

Coupling is not limited to shared global data (i.e., common coupling). There are five other types of coupling [3]: data coupling, stamp coupling, control coupling, external coupling, content coupling. These types of coupling refer to other categories of source code connection, which include parameters, data structure, logic control, and so on and so forth. However, please note that none of these coupling types are considered in this project.

### 

### 

## 2 System

For the interim report contained in this document, a single system is considered: OpenNLP, a machine learning based toolkit for processing natural language text. The reasons for the selection of this system as the first system to be analyzed are:

* Confirmed degree of common coupling to some extent
  + Contains multiple global variables
* Java
* Small system
  + Slightly over 1k files
  + 640+ java files
* Well-organized
  + Apache (github/jira/etc.)
* Related to class content

### 

## 3 Baseline

In order to validate the results of the automatic approaches for common coupling detection, it is necessary to establish a starting point for comparison. First, a manual test is performed and its results are used as the baseline for the automatic approaches. This baseline was defined as follows:

1. Select a single class for assessment.
   1. The class must contain at least one global variable.
2. For every file in the system, manually detect the usage of global variables that was declared in the previously selected class.
3. Manage the coupling degree
   1. Register the number of classes coupled (common coupling)
4. Store the results in a text file

To accomplish the definition described above, the Apache OpenNLP project was imported into Eclipse. The IDE fulfills the manual search for usage of global variables requirement (step 2). The results of this manual test, in other words, the baseline, are:

**Class:** SimpleTokenizer (in: opennlp.tools.tokenize.SimpleTokenizer.java)

**Number of global variables:** 1 (public static final SimpleTokenizer INSTANCE)

**Number of coupled classes (common coupling):** 4

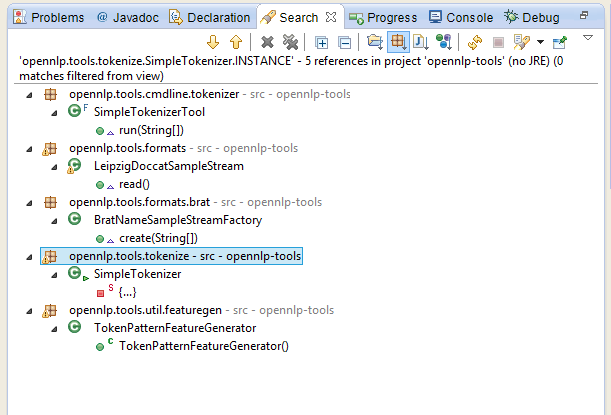


Image 1: Classes that uses the SimpleTokenizer.INSTANCE global variable.

Note that Image 1 shows the classes that contains any kind of usage (in this case, only read because the variable is *final*) of a global variable declared within the baseline class. It is possible to notice in this manual test that 4 references to the global variable were found (excluding the class itself, i.e., SimpleTokenizer).

## 4 Eclipse JDT/AST

The first iteration of the implementation of the Eclipse JDT/AST approach for common coupling detection is represented by a prototype. This first step consists of running the prototype on a single class, the same class selected for the baseline, and compare both results. The goal is to validate the implementation using ASTVisitor, and finally execute the next step, which is the execution of the prototype for the whole system (Apache OpenNLP).

### 

### 

## 5 References

1. Wikipedia. **Coupling (computer programming).** Accessed on: March, 29. Available at: <<https://en.wikipedia.org/wiki/Coupling_(computer_programming)>>.
2. Wikipedia. **Global variable.** Accessed on: March, 29. Available at: <<https://en.wikipedia.org/wiki/Global_variable>>.
3. Boukari Souley and Baba Bata. **A Class Coupling Analyzer for Java Programs.** West African Journal of Industrial and Academic Research Vol.7 No. 1 June 2013.
4. Michel Chaudron. **Design Heuristics and Architectural Styles (LL Chapter 9).** Accessed on: April, 5. Available at: <https://rickvanderzwet.nl/trac/personal/export/3/liacs/se/slides/09\_Design\_Heuristics\_and\_Styles\_part2.pdf>.