Match algorithm description

Bruno Crotman

04/05/2020

1 Introduction

This document is part of a larger research project about software degradation caused by careless developers' behavior and strategies to deal with such undesired behavior. The strategies to deal with this problem will possibly be inspired by concepts from game theory. At this moment, we assume that software degradation can be measured by the number and the types of kludges made by software developers in the code. So, one of the goals of this project is to study how software projects evolve in terms of number and kinds of kludges. Right now, we are trying to identify kludges by looking at alerts generated by the PMD source code analyzer.

To evaluate how the number of alerts evolves throughout the history of a software project, we must be able to analyze two different versions of a source code module (an old and a new version) and categorize each alert contained in the new version as either **new**, **fixed** or **open**.

A PMD alert generated for the old version is either **open** or **fixed** in the new version. An **open** alert remains in the new version of the code. A **fixed** alert does not exist in the new version.

A PMD alert generated for the new version is either **open** or **new**. An **open** alert indicates that the same alert was identified in the old version. A **new** alert implies that the same alert cannot be identified in the old version.

The alerts identified as **open** are equivalent in both new and old versions. To decide whether an alert is **open**, **fixed** or **new**, one has to identify if an alert in the old version is equivalent to an alert in the new version. The intersection between **fixed** alerts, **new** alerts and **open** alerts is empty.

In order to decide if an alert is **open**, **fixed** or **new**, we have to identify if an alert in the old version is equivalent to an alert in the new version. This document describes the algorithms we are using to make this classification.

In Section 2, I describe how I use PMD source code analyzer in two tasks. The first task is to list the alerts that represent possible kludges. PMD receives a source code (Prof Márcio, ele pode gerar alertas de um software inteiro, não só um conjunto, por isso tirei "module") as input and generates a list of bad programming practices contained in the code. The process we follow to generate the alerts using PMD source code analyzer is discussed in Section 2.1. The other task for which we use PMD is in the creation of an Abstract Syntax Tree (AST) from a source code with selected nodes. This will help us in one of the algorithms described in Section 3. The creation of the AST using PMD is described in Section 2.2.

In Section 3, I describe two algorithms that categorize the alerts as **new**, **fixed** or **open**. The first one, described in Section 3.1, is a naive algorithm based on matches by lines of code and some key features of the alerts. The second is a more sophisticated algorithm, based on matches by blocks of code, using the Abstract Syntax Tree.

2 PMD Source Code Analyzer

PMD is static source code analyzer that is commonly used to find possible programming flaws.

2.1 Using PMD to generate alerts

PMD traverses the AST of a source code searching for violations of rules that are configured by the user. PMD comes with a default rule set for Java language. The default rule set finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It´s possible configure a different set of rules creating a custom XML file. At this point we use the default rule set to generate the alerts that we interpret as kludges and try to categorize in Section 3

In Figure 1, we can see an example of a simple code and the alerts that were generated by the default rule set of PMD alerts tool.

```
/*  1-                                   */package pack_x;
/*  2-                                   */
/*  3-                                   */import importX.function;
/*  4-                                   */
/*  5-                                   */class ClassX extends ClassY implements InterfX {
/*  6-                                   */    private long fieldX;
/*  7-                                   */
/*  8-                                   */    ClassX(int paramX, double paramY) {
/*  9-                                   */        int varX = function(paramX, paramY);
/* 10-                                   */        if (varX == 0)
/* 11-ControlStatementBraces             */            this.fieldX = 1;
/* 12-                                   */        else{
/* 13-                                   */            this.fieldX = 0;
/* 14-                                   */     }
/* 15-                                   */    }
/* 16-                                   */    @Override
/* 17-                                   */    public int methodX(int paramW, Boolean paramZ)
/* 18-                                   */    {
/* 19-                                   */        if (paramZ)
/* 20-ControlStatementBraces             */            fieldX = paramW;
/* 21-                                   */        else{
/* 22-                                   */            fieldX = 0;
/* 23-                                   */     }
/* 24-                                   */        return paramW + this.fieldX;
/* 25-                                   */     }
/* 26-                                   */}
```

Figure 1: Simple code with its alerts

2.2 Using PMD to generate an Abstract Syntax Tree

3 Algorithms to categorize alerts

3.1 Matches by line of code

In this first algorithm, I match the lines of code of the old version with the lines of code of the new version using information from the output of git's diff command. When an alert with the same features occurs in both matched lines, this alert is declared **open**. The alerts that occur in a not matched line of the old version are declared *fixed* and the alerts located in a unmatched line of the new version are declared as **new**.

These are the steps of the algorithm:

1. Generate a list of alerts from each version (old and new) using PMD Alert (Section 3.2);
2. Generate the git diff between the two versions (Section 3.2.1);
3. Using information from git diff, create a map between the lines (Section 3.3);
4. Categorize the alerts (Section 3.4).

3.2 Generate a list of alerts for each version

The two codes presented in this Section, named "new version" and "old version", are used in Sections 3.2.1, 3.3 and 3.4 to describe the algorithm.

The old version, with the alerts generated by PMI, is shown in Figure 2.

```
/*  1-                                   */package twitter4j;
/*  2-                                   */
/*  3-UnusedImports                      */import java.util.concurrent.ConcurrentHashMap;
/*  4-                                   */
/*  5-                                   */class TwitterImpl extends TwitterBaseImpl implements Twitter {
/*  6-                                   */    private static final long serialVersionUID = 9170943084096085770L;
/*  7-UnusedPrivateField                 */    private static final Logger logger = Logger.getLogger(TwitterBaseImpl.class);
/*  8-                                   */
/*  9-                                   */    /*package*/
/* 10-                                   */    TwitterImpl(Configuration conf, Authorization auth) {
/* 11-                                   *//* ...  */
/* 45-                                   *//* ...  */
/* 46-                                   */            if (conf.isTweetModeExtended()) {
/* 47-                                   */                params.add(new HttpParameter("tweet_mode", "extended"));
/* 48-                                   */            }
/* 49-OptimizableToArrayCall             */            HttpParameter[] implicitParams = params.toArray(new HttpParameter[params.size()]);
/* 50-                                   */
/* 51-                                   */            // implicitParamsMap.containsKey() is evaluated in the above if clause.
/* 52-                                   */            // thus implicitParamsStrMap needs to be initialized first
/* 53-                                   *//* ...  */
/* 59-                                   *//* ...  */
/* 60-                                   */
/* 61-                                   */
/* 62-                                   */    @Override
/* 63-FormalParameterNamingConventions   */    public AccountSettings updateAccountSettings(Integer trend_locationWoeid,
/* 64-FormalParameterNamingConventions(2)*/                                                 Boolean sleep_timeEnabled, String start_sleepTime,
/* 65-FormalParameterNamingConventions(2)*/                                                 String end_sleepTime, String time_zone, String lang)
/* 66-                                   */            throws TwitterException {
/* 67-                                   */        List<HttpParameter> profile = new ArrayList<HttpParameter>(6);
/* 68-                                   */        if (trend_locationWoeid != null) {
/* 69-                                   *//* ...  */
/* 83-                                   *//* ...  */
/* 84-                                   */            profile.add(new HttpParameter("lang", lang));
/* 85-                                   */        }
/* 86-                                   */        return factory.createAccountSettings(post(conf.getRestBaseURL() + "account/settings.json"
/* 87-OptimizableToArrayCall             */                , profile.toArray(new HttpParameter[profile.size()])));
/* 88-                                   */
/* 89-                                   */    }
/* 90-                                   */
/* 91-                                   */}
```

Figure 2: Example: old version

Table 1 lists the alerts found in the old version.

Table 1: Alerts in the old version

id	beginline	ruleset	rule	package	class	method	variable
1	3	Best Practices	UnusedImports	twitter4j	TwitterImpl	No method	No variable
2	7	Best Practices	UnusedPrivateField	twitter4j	TwitterImpl	No method	logger
3	49	Performance	OptimizableToArrayCall	twitter4j	TwitterImpl	TwitterImpl	No variable
4	63	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	trend__locationWoeid
5	64	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	sleep__timeEnabled
6	64	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	start__sleepTime
7	65	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	end__sleepTime
8	65	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	time__zone
9	87	Performance	OptimizableToArrayCall	twitter4j	TwitterImpl	updateAccountSettings	No variable

The new version, shown in Figure 3, has the alerts listed in Table 2.

Table 2: Alerts in the new version

id	beginline	ruleset	rule	package	class	method	variable
1	5	Best Practices	UnusedPrivateField	twitter4j	TwitterImpl	No method	logger
2	47	Performance	OptimizableToArrayCall	twitter4j	TwitterImpl	TwitterImpl	No variable
3	62	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	sleep__timeEnabled
4	62	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	start__sleepTime
5	63	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	end__sleepTime
6	63	Code Style	FormalParameterNamingConventions	twitter4j	TwitterImpl	updateAccountSettings	time__zone
7	85	Performance	OptimizableToArrayCall	twitter4j	TwitterImpl	updateAccountSettings	No variable
8	89	Best Practices	UnusedPrivateField	twitter4j	TwitterImpl	No method	not__used

By mapping the alerts in both version, we observe the following:

- line 3 in the old version had a “Unused Import” which was removed in the new version. So, the alert related to this line must be declared **fixed**;
- line 63 in the old version (61 in the new version) was fixed by changing the name of the parameter. This must be classified as another **fixed** alert;
- line 89 was added to the new version and contains a unused private field. Such must be categorized as a **new** alert.

```
/*  1-                                   */package twitter4j;
/*  2-                                   */
/*  3-                                   */class TwitterImpl extends TwitterBaseImpl implements Twitter {
/*  4-                                   */    private static final long serialVersionUID = 9170943084096085770L;
/*  5-UnusedPrivateField                 */    private static final Logger logger = Logger.getLogger(TwitterBaseImpl.class);
/*  6-                                   */
/*  7-                                   */    /*package*/
/*  8-                                   */    TwitterImpl(Configuration conf, Authorization auth) {
/*  9-                                   *//* ...  */
/* 43-                                   *//* ...  */
/* 44-                                   */            if (conf.isTweetModeExtended()) {
/* 45-                                   */                params.add(new HttpParameter("tweet_mode", "extended"));
/* 46-                                   */            }
/* 47-OptimizableToArrayCall             */            HttpParameter[] implicitParams = params.toArray(new HttpParameter[params.size()]);
/* 48-                                   */
/* 49-                                   */            // implicitParamsMap.containsKey() is evaluated in the above if clause.
/* 50-                                   */            // thus implicitParamsStrMap needs to be initialized first
/* 51-                                   *//* ...  */
/* 58-                                   *//* ...  */
/* 59-                                   */
/* 60-                                   */    @Override
/* 61-                                   */    public AccountSettings updateAccountSettings(Integer trendlocationWoeid,
/* 62-FormalParameterNamingConventions(2)*/                                                 Boolean sleep_timeEnabled, String start_sleepTime,
/* 63-FormalParameterNamingConventions(2)*/                                                 String end_sleepTime, String time_zone, String lang)
/* 64-                                   */            throws TwitterException {
/* 65-                                   */        List<HttpParameter> profile = new ArrayList<HttpParameter>(6);
/* 66-                                   */        if (trendlocationWoeid != null) {
/* 67-                                   *//* ...  */
/* 81-                                   *//* ...  */
/* 82-                                   */            profile.add(new HttpParameter("lang", lang));
/* 83-                                   */        }
/* 84-                                   */        return factory.createAccountSettings(post(conf.getRestBaseURL() + "account/settings.json"
/* 85-OptimizableToArrayCall             */                , profile.toArray(new HttpParameter[profile.size()])));
/* 86-                                   */
/* 87-                                   */    }
/* 88-                                   */
/* 89-UnusedPrivateField                 */    private int not_used = 0;
/* 90-                                   */
/* 91-                                   */}
```

Figure 3: Example: new version

Table 3: Relation between lines of the old version and lines of the new version

1	2	3	4	5	6	7-	-59	60	61		62	63	64	65	66		67		68	69	70	71	72-	-87	88	89		90		91
1	2			3	4	5-	-57	58	59	61	60		62	63	64	66	65	67			68	69	70-	-85	86	87	89	88	90	91

3.2.1 Generate the git diff between the two versions

The *git diff* command is executed between two subsequent versions with the option –patience. The result of the *git diff* operation between the versions presented in the last section is shown in Figure 4.

```
5   5   j/match_algorithm_description/{old => new}/code.java

diff --git a/j/match_algorithm_description/old/code.java b/j/match_algorithm_description/new/code.java
index cd61181..245a6e8 100644
--- a/j/match_algorithm_description/old/code.java
+++ b/j/match_algorithm_description/new/code.java
@@ -3,2 +2,0 @@ package twitter4j;
-import java.util.concurrent.ConcurrentHashMap;
-
@@ -63 +61 @@ class TwitterImpl extends TwitterBaseImpl implements Twitter {
-    public AccountSettings updateAccountSettings(Integer trend_locationWoeid,
+    public AccountSettings updateAccountSettings(Integer trendlocationWoeid,
@@ -68,2 +66,2 @@ class TwitterImpl extends TwitterBaseImpl implements Twitter {
-        if (trend_locationWoeid != null) {
-            profile.add(new HttpParameter("trend_location_woeid", trend_locationWoeid));
+        if (trendlocationWoeid != null) {
+            profile.add(new HttpParameter("trend_location_woeid", trendlocationWoeid));
@@ -90,0 +89,2 @@ class TwitterImpl extends TwitterBaseImpl implements Twitter {
+    private int not_used = 0;
+
```

Figure 4: Git diff between code in Figure 2 and code in Figure 3

3.3 Using information from git diff, create a relation between the lines

For each difference stated in the output (the sections of the diff file starting with "@@"), there is an indication of the number of lines removed from the old version and the number of lines added to the new one. The line in which the lines are removed from the old version and the line at which the lines are added is indicated, too. By using this information we create a relation between the lines of the old version and the equivalent lines in the new version. For the new and old versions presented in Section 3.2, the relation is shown in Table 3.

Now we can connect the two versions, as well as their alerts, as we see in Figure 5.

```
/*  1-                       */package twitter4j;                                                    /*  1-                       */package twitter4j;
/*  2-                       */                                                                      /*  2-                       */
/*  3-UnusedImports          */import java.util.concurrent.ConcurrentHashMap;                        /*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*  4-                       */                                                                      /*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*  5-                       */class TwitterImpl extends TwitterBaseImpl implements Twitter {        /*  3-                       */class TwitterImpl extends TwitterBaseImpl implements Twitter {
/*  6-                       */    private static final long serialVersionUID = 9170943084096085770L;    /*  4-                       */    private static final long serialVersionUID = 9170943084096085770L;
/*  7-UnusedPrivateField     */    private static final Logger logger = Logger.getLogger(TwitterBaseImpl.class);    /*  5-UnusedPrivateField     */    private static final Logger logger = Logger.getLogger(TwitterBaseImpl.class);
/*  8-                       */                                                                      /*  6-                       */
/*  9-                       */    /*package*/                                                       /*  7-                       */    /*package*/
/* 10-                       */    TwitterImpl(Configuration conf, Authorization auth) {             /*  8-                       */    TwitterImpl(Configuration conf, Authorization auth) {
/* 11-                       *//* ...  */                                                            /*  9-                       *//* ...  */
/* 45-                       *//* ...  */                                                            /* 43-                       *//* ...  */
/* 46-                       */            if (conf.isTweetModeExtended()) {                         /* 44-                       */            if (conf.isTweetModeExtended()) {
/* 47-                       */                params.add(new HttpParameter("tweet_mode", "extended"));    /* 45-                       */                params.add(new HttpParameter("tweet_mode", "extended"));
/* 48-                       */            }                                                         /* 46-                       */            }
/* 49-OptimizableToArrayCa   */            HttpParameter[] implicitParams = params.toArray(new HttpParameter[params.size(/* 47-OptimizableToArrayCa   */            HttpParameter[] implicitParams = params.toArray(new HttpParameter[params.size(
/* 50-                       */                                                                      /* 48-                       */
/* 51-                       */            // implicitParamsMap.containsKey() is evaluated in the above if clause.       /* 49-                      */            // implicitParamsMap.containsKey() is evaluated in the above if clause.
/* 52-                       */            // thus implicitParamsStrMap needs to be initialized first                    /* 50-                      */            // thus implicitParamsStrMap needs to be initialized first
/* 53-                       *//* ...  */                                                            /* 51-                       *//* ...  */
/* 60-                       *//* ...  */                                                            /* 58-                       *//* ...  */
/* 61-                       */                                                                      /* 59-                       */
/*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                            /* 61-                       */    public AccountSettings updateAccountSettings(Integer trendlocationWoeid,
/* 62-                       */    @Override                                                         /* 60-                       */    @Override
/* 63-FormalParameterNamin   */    public AccountSettings updateAccountSettings(Integer trend_locationWoeid,    /*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 64-FormalParameterNamin(2)*/                                        Boolean sleep_timeEnabled, String start_s/* 62-FormalParameterNamin(2)*/                                        Boolean sleep_timeEnabled, String start_s
/* 65-FormalParameterNamin(2)*/                                        String end_sleepTime, String time_zone, S/* 63-FormalParameterNamin(2)*/                                        String end_sleepTime, String time_zone, S
/* 66-                       */            throws TwitterException {                                 /* 64-                       */            throws TwitterException {
/*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                            /* 66-                       */        if (trendlocationWoeid != null) {
/* 67-                       */        List<HttpParameter> profile = new ArrayList<HttpParameter>(6);    /* 65-                       */        List<HttpParameter> profile = new ArrayList<HttpParameter>(6);
/*   -                       *//* ...  */                                                            /* 67-                       *//* ...  */
/* 83-                       *//* ...  */                                                            /* 81-                       *//* ...  */
/* 84-                       */            profile.add(new HttpParameter("lang", lang));             /* 82-                       */            profile.add(new HttpParameter("lang", lang));
/* 85-                       */        }                                                             /* 83-                       */        }
/* 86-                       */        return factory.createAccountSettings(post(conf.getRestBaseURL() + "account/setting/* 84-                       */        return factory.createAccountSettings(post(conf.getRestBaseURL() + "account/setting
/* 87-OptimizableToArrayCa   */                , profile.toArray(new HttpParameter[profile.size()])));    /* 85-OptimizableToArrayCa   */                , profile.toArray(new HttpParameter[profile.size()])));
/* 88-                       */                                                                      /* 86-                       */
/* 89-                       */    }                                                                 /* 87-                       */    }
/*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                            /* 89-UnusedPrivateField     */    private int not_used = 0;
/* 90-                       */                                                                      /* 88-                       */
/*   -                       *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                            /* 90-                       */
/* 91-                       */}                                                                     /* 91-                       */}
```

Figure 5: Comparison between old and new version

3.4 Categorize the alerts

The alert in the old version is classified as **open** if there is an alert in the new version in the corresponding line with the same rule, same method name, and same variable name. The associated rule, the name of the method and variable affected by the alert are provided by PMD as part of the description of the alert. Otherwise, the alert is categorized as **fixed**.

Table 4 shows the classification of the alerts in the old version. For alerts 1 and 4, it is not possible to find an alert with the same rule, method and variable name in the related line of code in the new version. So, these alerts are classified as **fixed**.

For the remaining alerts, it is possible to find an alert with the same rule, method and variable name in the related line of code in the new version. So, these alerts are classified as **open**.

Marcio: referenciar as tabelas 4 e 5 no corpo do texto. Bruno: Tá referenciado, não? Botei em vermelho.

Table 4: Classifications of the alerts in the old version

id	line	rule	class	method	variable	idnew	linenew	category
1	3	UnusedImports	TwitterImpl	No method	No variable	NA	NA	Fixed
4	63	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	trend_locationWoeid	NA	NA	Fixed

Table 5 shows the classification of the alerts in the new version. The alert 8 is the only one for which there is no alert with the same characteristics in a line of the old version that is mapped to the original line in the new version. So it is the only **new** alert.

Table 5: Classifications of the alerts in the new version

id	line	rule	class	method	variable	idold	lineold	category
1	5	UnusedPrivateField	TwitterImpl	No method	logger	2	7	Open
2	47	OptimizableToArrayCall	TwitterImpl	TwitterImpl	No variable	3	49	Open
3	62	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	sleep_timeEnabled	5	64	Open
4	62	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	start_sleepTime	6	64	Open
5	63	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	end_sleepTime	7	65	Open
6	63	FormalParameterNamingConventions	TwitterImpl	updateAccountSettings	time_zone	8	65	Open
7	85	OptimizableToArrayCall	TwitterImpl	updateAccountSettings	No variable	9	87	Open
8	89	UnusedPrivateField	TwitterImpl	No method	not_used	NA	NA	New

4 Matches using the Abstract Syntax Tree and the relation between lines of code of the versions

4.1 Creating an Abstract Syntax Tree

The Abstract Syntax Tree (AST) of a source code contains the elements represented in the code, such as class declarations, method declarations, statements. The AST can be used, in conjunction with the relation between the lines we saw in Section 3.3, to understand the location of an alert in a version of a code.

The PMD Alert tool lets us configure our own rules. In section 3.2 we generated alerts using the default configuration of alerts.

The alerts that PMD generates are elements that the tool captures as it traverses the code, visiting all the elements of the AST. An element is captured and becomes an alert when it is matched with a rule. The rules are defined in a XML file.

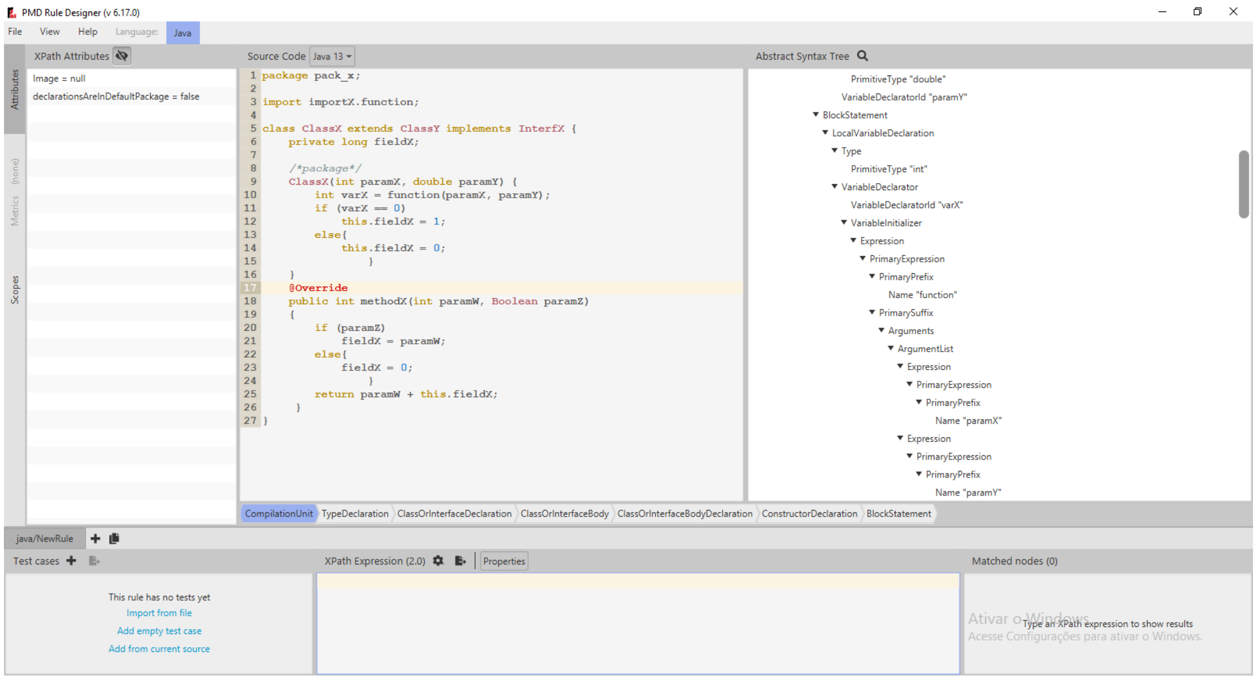
Prof. Márcio Este tipo de informação deveria vir no início do texto. Você já tratou com alertas identificados antes ...

Bruno: É que aqui estou explicando como eu uso o mecanismo de geração de alertas pra gerar a AST. Mas vou pensar na melhor forma de dividir essas coisas no texto

Marcio: mas você não usou os mesmos kinds of elements da tabela 6 para gerar os alertas no algoritmo naive? Tudo que for comum aos dois deve vir no início do texto, antes do primeiro algoritmo. Como está, você fala do primeiro algoritmo, dá uma escapada do tema, e depois retorna, deixando o leitor um pouco mais perdido sobre o que você está tratando. Tenta reorganizar esta parte do texto. Acho que vai ficar melhor.

Não. Na verdade eu rodo o PMD com o ruleset padrão pra gerar os alertas de possíveis kludges e tal, e uso esses alertas default, que já vem com algumas informações. Neste momento aqui eu rodo o PMD com um ruleset maceteado pra pegar todos os nós dos tipos escolhidos. Nesta hora ele não está pegando alertas, mas sim nós da AST. Vou explicar melhor nesta versão aqui.

It’s possible to create custom rules for PMD. Figure 6 shows the designer tool that helps a user to create custom rules.



```
Image = null
declarationsAreInDefaultPackage = false
```

```java
package pack_x;

import importX.function;

class ClassX extends ClassY implements InterfX {
    private long fieldX;

    /*package*/
    ClassX(int paramX, double paramY) {
        int varX = function(paramX, paramY);
        if (varX == 0)
            this.fieldX = 1;
        else{
            this.fieldX = 0;
                }
    }
    @Override
    public int methodX(int paramW, Boolean paramZ)
    {
        if (paramZ)
            fieldX = paramW;
        else{
            fieldX = 0;
                }
        return paramW + this.fieldX;
     }
}
```

```
        PrimitiveType "double"
      VariableDeclaratorId "paramY"
  ▼ BlockStatement
    ▼ LocalVariableDeclaration
      ▼ Type
          PrimitiveType "int"
      ▼ VariableDeclarator
          VariableDeclaratorId "varX"
        ▼ VariableInitializer
          ▼ Expression
            ▼ PrimaryExpression
              ▼ PrimaryPrefix
                  Name "function"
              ▼ PrimarySuffix
                ▼ Arguments
                  ▼ ArgumentList
                    ▼ Expression
                      ▼ PrimaryExpression
                        ▼ PrimaryPrefix
                            Name "paramX"
                    ▼ Expression
                      ▼ PrimaryExpression
                        ▼ PrimaryPrefix
                            Name "paramY"
```

CompilationUnit › TypeDeclaration › ClassOrInterfaceDeclaration › ClassOrInterfaceBody › ClassOrInterfaceBodyDeclaration › ConstructorDeclaration › BlockStatement

This rule has no tests yet

Type an XPath expression to show results

Figure 6: PMD Designer tool

We can see that the tool traverses the programming code visiting many different kinds of elements. If we build our own simple rules, aimed only to capture some kinds of elements, we will generate list of “alerts” that will contain all the elements of the chosen kinds.

In Figure 7, we show an example of a ruleset that captures all the method declarations.

```
<?xml version="1.0"?>
<ruleset name="complete"
         xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0 https://pmd.sourceforge.io/ruleset_2_0_0.xsd"
    <description>Test.</description>

    <rule name="method"
      language="java"
      message="method"
      class="net.sourceforge.pmd.lang.rule.XPathRule" >
    <description>
TODO
    </description>
    <priority>3</priority>
    <properties>
        <property name="xpath">
            <value>
<![CDATA[
//MethodDeclaration
]]>
            </value>
        </property>
    </properties>
    </rule>
]

</ruleset>
```

Figure 7: Custom ruleset for PMD alerts tool

Table 6 shows the kinds of elements that were selected for the creation of an AST for the code in Figure 1.

Table 6: Kinds of elements selected

rule_number	rule
1	annotation
2	block
3	class_or_interface_body
4	class_or_interface_declaration
5	class_or_interface_type
6	compilation_unit
7	constructor_declaration
8	extends_list
9	field_declaration
10	formal_parameter
11	formal_parameters
12	if_statement
13	implements_list
14	import_declaration
15	method
16	name
17	package
18	statement
19	type_declaration
20	variable_id

If we select the list in Table 6, the simple code shown in this Section captures the elements shown in Table 7

Table 7: Elements captured in code

line	endline	col	endcol	rule	method	code
1	26	1	3	compilation_unit	No method	package pack_x; import i...
1	1	1	15	package	No method	package pack_x;
1	1	9	14	name	No method	pack_x
3	3	1	24	import_declaration	No method	import importX.function;
3	3	8	23	name	No method	importX.function
5	26	1	1	class_or_interface_declaration	No method	class ClassX extends ClassY...
5	5	14	27	extends_list	No method	extends ClassY
5	5	22	27	class_or_interface_type	No method	ClassY
5	5	29	46	implements_list	No method	implements InterfX
5	5	40	46	class_or_interface_type	No method	InterfX
5	26	48	1	class_or_interface_body	No method	{ private long fieldX; ...
6	6	13	24	field_declaration	No method	long fieldX;
6	6	18	23	variable_id	No method	fieldX
8	15	5	5	constructor_declaration	ClassX	ClassX(int paramX, double p...
8	8	11	37	formal_parameters	ClassX	(int paramX, double paramY)
8	8	12	21	formal_parameter	ClassX	int paramX
8	8	16	21	variable_id	ClassX	paramX
8	8	24	36	formal_parameter	ClassX	double paramY
8	8	31	36	variable_id	ClassX	paramY
9	9	13	16	variable_id	ClassX	varX
9	9	20	27	name	ClassX	function
9	9	29	34	name	ClassX	paramX
9	9	37	42	name	ClassX	paramY
10	14	9	17	statement	ClassX	if (varX == 0) ...
10	14	9	17	if_statement	ClassX	if (varX == 0) ...
10	10	13	16	name	ClassX	varX
11	11	13	28	statement	ClassX	this.fieldX = 1;
12	14	13	17	block	ClassX	{ this.fieldX =...
12	14	13	17	statement	ClassX	{ this.fieldX =...
13	13	13	28	statement	ClassX	this.fieldX = 0;
16	16	5	13	annotation	No method	@Override
16	16	6	13	name	No method	Override
17	25	12	6	method	methodX	int methodX(int paramW, Boo...
17	17	23	50	formal_parameters	methodX	(int paramW, Boolean paramZ)
17	17	24	33	formal_parameter	methodX	int paramW
17	17	28	33	variable_id	methodX	paramW
17	17	36	42	class_or_interface_type	methodX	Boolean
17	17	36	49	formal_parameter	methodX	Boolean paramZ
17	17	44	49	variable_id	methodX	paramZ
18	25	5	6	block	methodX	{ if (paramZ) ...
19	23	9	17	statement	methodX	if (paramZ) fie...
19	23	9	17	if_statement	methodX	if (paramZ) fie...
19	19	13	18	name	methodX	paramZ
20	20	13	28	statement	methodX	fieldX = paramW;
20	20	13	18	name	methodX	fieldX
20	20	22	27	name	methodX	paramW
21	23	13	17	block	methodX	{ fieldX = 0; }
21	23	13	17	statement	methodX	{ fieldX = 0; }
22	22	13	23	statement	methodX	fieldX = 0;
22	22	13	18	name	methodX	fieldX
24	24	9	36	statement	methodX	return paramW + this.fieldX;
24	24	16	21	name	methodX	paramW

Table 6 contains the list and location of the elements of the AST. In order to recreate the AST, we must follow three steps:

1. Link each element a to the set of elements X that are fully located between the begin line / begin

column and end line / end column of element a. We can construct a directed graph in which the elements are the nodes and the links are the edges. This is not a tree yet, because each node will have edges directed to all its descendants and not only its children in the AST.

2. Sort the nodes in the decreasing order of its number of children. The objective is to establish that, in a search through this graph, the first child chosen will be the one that is a child in the AST, and not only on this graph.
3. Proceed a deep-first search starting from the compilation unit node.

After we follow these steps, we come up with the AST as we se in Figure 8.

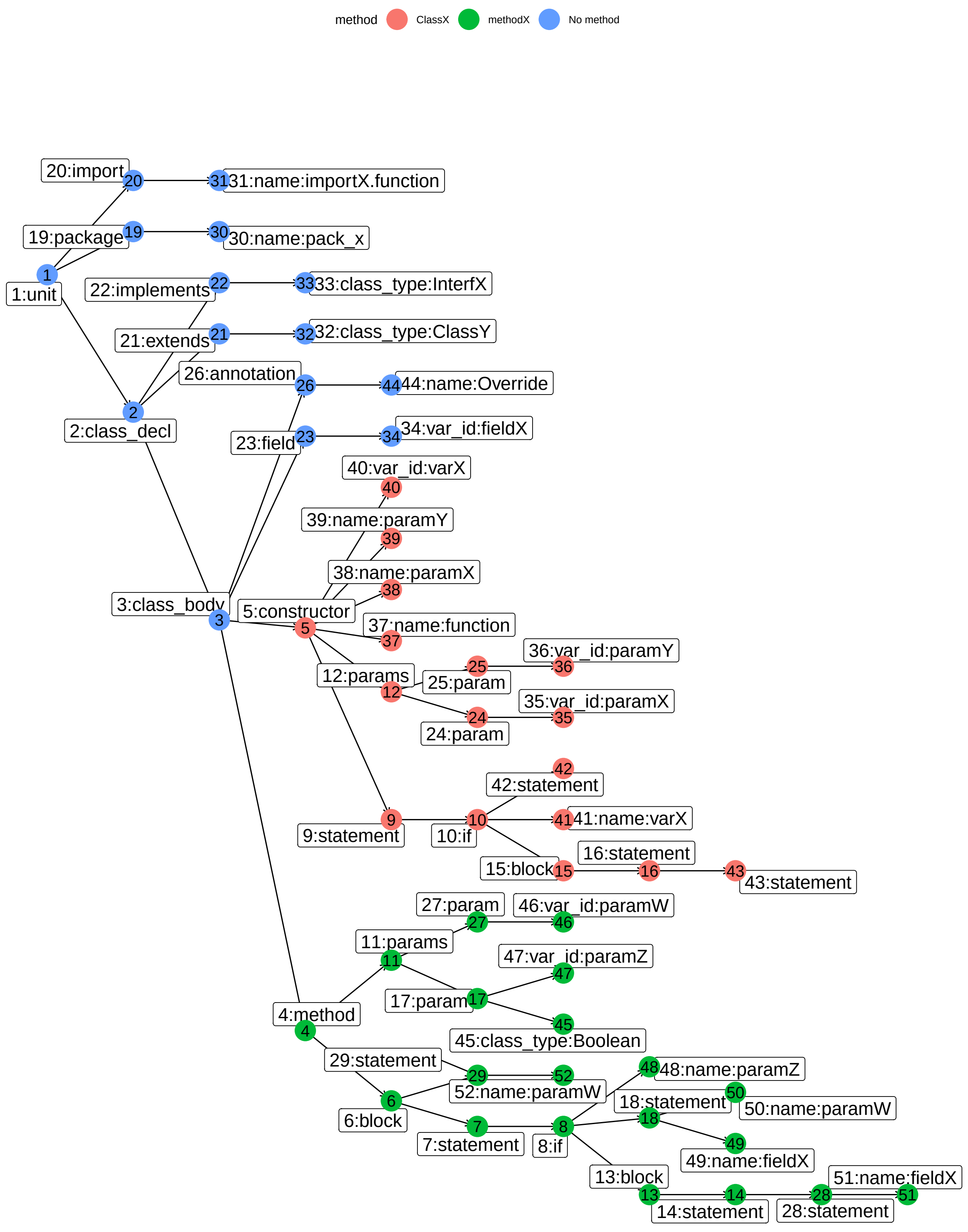


Figure 8: Abstract Syntax Tree

Prof. Márcio: OK, mas cadê o algoritmo para classificar os alertas

Bruno: eu ainda não fiz isso, mesmo. Mas vai ser mais ou menos o seguinte, só preciso fazer umas experiências e organizar melhor: vou partir do que faço no naive, mas, para os alertas que não baterem, vou tentar subir na AST em direção à raiz e ver se num nó num nível acima as coisas batem (o nó da árvore tem que bater com o nó da outra árvore via match de linha e tipo de nó, e tem que bater também as características do alerta), depois posso subir de novo, se não der... Só tem que pensar bem porque tem duas árvores, e alertas subindo pelas duas árvores. Tenho que experimentar agora que já consigo montar as árvores e tal. testar uns exemplos e ver o que acontece. Consegui me familiarizar com umas bibliotecas do R que ajudam a manipular e plotar grafos (tidygraph e ggraph). Tem um trade-off entre complexidade do algoritmo e a quantidade de falsos negativos que ele vai dar por não bater alertas que deveriam bater. Não vou esgotar isso agora, mas quero organizar UM POUCO essas ideias e mostrar esses recursos da AST pra ajudar o Earl e vc a pensarem comigo

Marcio: beleza. me avisa quando tiver uma nova versão que eu analiso. Note que você não precisa entrar no nível de detalhes de descrever uma AST, mostrar um exemplo de AST ou dizer como se gera o código a partir de uma AST. Eu e Earl, assim como os seus leitores, sabemos disso. O importante é você mostrar uma limitação do algoritmo anterior, de preferência com um exemplo, dizer como vai usar as informações da AST para resolver as limitações do caso anterior e descrever o novo algoritmo.

Bruno: vou condensar, mas considero importante mostrar que elementos eu tô pegando nesta AST específica que estou montando, já que não pego TODOs os nós, apenas os selecionados .