PMD alerts as possible kludges: open, fixed or new?

Bruno Crotman

Contents

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 (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, we describe how we 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 as input and generates a list of bad programming practices contained in the code, i.e., the alerts. The process we follow to generate the alerts using PMD source code analyzer is discussed in Section 2.1.
- The second 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 the algorithm described in Section 3.2. The creation of the AST using PMD is described in Section 2.2.

In Section 3, we describe two algorithms that help to 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. In this work we use PMD for two tasks: to generate alerts that we interpret as clues about kludges and to create an AST from the source code with selected kinds of node.

2.1 Using PMD to generate alerts

PMD traverses the AST of a source code searching for violations of rules which 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 to 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 by the algorithms described 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

The AST is used, in conjunction with the relation between the lines we will see in Section 3.1.3, to understand the location of an alert in a version of a code. We use this information in the algorithm described in Section 3.2

We can see that the tool traverses the source 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 contained in the AST.

We don´t use all the types of nodes recognized by PMD Alert to generate the AST. The kinds of elements that were selected are the following ones:

- **Annotation**: a syntatic metadata added to a source code.
- **Block**: a block of statements enclosed by braces.
- **ClassOrInterfaceBody**: the body of an interface or a class, excluding the declaration.
- **ClassOrInterfaceDeclaration**: class or interface, including the declaration and the body.
- **ClassOrInterfaceType**: declaration of a type.
- **CompilationUnit**: the root of an AST tree.
- **ConstructorDecaration**: class or interface, including the declaration and the body.
- **ExtendsList**: list of extensions of a class.
- **FieldDeclaration**: field declaration, including the type, the name and the possible assignment.
- **FormalParameter**: a parameter of a method or constructor.
- **FormalParameters**: list of formal parameters of a method or constructor.
- **IfStatement**: an if statements including its blocks and condition.
- **ImplementsList**: list of implementations of an interface.
- **Import**: a package imported.
- **Method**: a method, including body and declaration.
- **Name**: a named value, like a variable, a class or a package referenced in the code.
- **package**: the indication of the package to which the compilation unit belongs.
- **statement**: any statement, like an if statement or an assignment.
- **TypeDeclaration**: any type declaration.
- **VariableId**: the name of a variable in a variable declaration.

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 contained 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 described 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 a descendant.
3. Proceed a deep-first search starting from the compilation unit node.

3 Algorithms to categorize alerts

3.1 Algorithm 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 3.1.1);
2. Generate the git diff between the two versions (Section 3.1.2);
3. Using information from git diff, create a = map between the lines (Section 3.1.3);
4. Categorize the alerts (Section 3.1.4).

3.1.1 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 3.1.2, 3.1.3 and 3.1.4 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

3.1.2 Generate the git diff between the two versions

The *git diff* command is executed between two subsequent versions with the option –patience. The patience algorithm works better when the changes are big. The result of the *git diff* operation between the versions presented in the last section is shown in Figure 4.

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

diff --git a/old/code.java b/new/code.java
index cd61181..245a6e8 100644
--- a/old/code.java
+++ b/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

3.1.3 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 3.1.1, 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 = 91709430840960/*  4-                */    private static final long serialVersionUID = 91709430840960
/*  7-UnusedPrivateF   */    private static final Logger logger = Logger.getLogger(Twitt/*  5-UnusedPrivateF  */    private static final Logger logger = Logger.getLogger(Twitt
/*  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", "ext/* 45-                */                params.add(new HttpParameter("tweet_mode", "ext
/* 48-                 */            }                                                  /* 46-                */            }
/* 49-OptimizableToA   */            HttpParameter[] implicitParams = params.toArray(new/* 47-OptimizableToA  */            HttpParameter[] implicitParams = params.toArray(new
/* 50-                 */                                                               /* 48-                */
/* 51-                 */            // implicitParamsMap.containsKey() is evaluated in /* 49-                */            // implicitParamsMap.containsKey() is evaluated in
/* 52-                 */            // thus implicitParamsStrMap needs to be initialize/* 50-                */            // thus implicitParamsStrMap needs to be initialize
/* 53-                 *//* ...  */                                                     /* 51-                *//* ...  */
/* 60-                 *//* ...  */                                                     /* 58-                *//* ...  */
/* 61-                 */                                                               /* 59-                */
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                     /* 61-                */    public AccountSettings updateAccountSettings(Integer trendl
/* 62-                 */    @Override                                                  /* 60-                */    @Override
/* 63-FormalParamete   */    public AccountSettings updateAccountSettings(Integer trend_/*   -                *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 64-FormalParamete(2)*/                                                 Boolean sleep_/* 62-FormalParamete(2)*/                                                 Boolean sleep_
/* 65-FormalParamete(2)*/                                                 String end_sle/* 63-FormalParamete(2)*/                                                 String end_sle
/* 66-                 */            throws TwitterException {                          /* 64-                */            throws TwitterException {
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                     /* 66-                */        if (trendlocationWoeid != null) {
/* 67-                 */        List<HttpParameter> profile = new ArrayList<HttpParamet/* 65-                */        List<HttpParameter> profile = new ArrayList<HttpParamet
/*   -                 *//* ...  */                                                     /* 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.getRestB/* 84-                */        return factory.createAccountSettings(post(conf.getRestB
/* 87-OptimizableToA   */                , profile.toArray(new HttpParameter[profile.siz/* 85-OptimizableToA  */                , profile.toArray(new HttpParameter[profile.siz
/* 88-                 */                                                               /* 86-                */
/* 89-                 */    }                                                          /* 87-                */    }
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                     /* 89-UnusedPrivateF  */    private int not_used = 0;
/* 90-                 */                                                               /* 88-                */
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/                     /* 90-                */
/* 91-                 */}                                                              /* 91-                */}
```

Figure 5: Comparison between old and new version

3.1.4 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 Source Code Analyzer 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 name and variable name in the related line of code in the new version. So, these alerts are classified as **fixed**. It´s easy to compare the rule, the variable name related to the alert and the method name, because all of them are outputs from the execution of PMD Source Code Analyzer. They are presented in the output as characteristics of the alert.

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**.

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
8	89	UnusedPrivateField	TwitterImpl	No method	not_used	NA	NA	New

3.2 Algorithm 2: Matches using the Abstract Syntax Tree and the relation between lines of code of the versions

This Section discusses the Algorithm 2, which uses the AST to create features that help to infer if the alerts in different versions must be considered the same. Nevertheless this algorithm does not return a single categorical answer for each alert (new, open or fixed), but features, it considers many cases that are ignored by the algorithm described in Section 3.1.

The Algorithm 1 considers only the method name, the variable name, the lines of code (using git´s diff) and the type of alert. All these characteristics must be equivalent in order to two alerts be considered the same. For instance, if the developer changes only the name of the method but everything stays the same, the Algorithm 1 will not consider that an alert located in this renamed method is still open.

In the Algorithm 2, many features other than "Same Method Name" will indicate that the alerts in the two versions could be the same if the only thing that changed was the name of the method.

In the Section 3.2.1, we show how the features are created. Even though it's not possible to be sure if a pair alerts in different versions are same alert, we think that these features can provide some clues. The features can be an input to a heuristic or a machine learning tool which can decide if the alerts are the same.

3.2.1 Feature engineering

In this Section, we will consider the old version in Figure 6 and the new version in Figure 7. In the new version, the alert generated by the line 11 of the old version was fixed.

```
/*  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 6: Old version

```
/*  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-                                  */        {
/* 12-                                  */            this.fieldX = 1;
/* 13-                                  */        }
/* 14-                                  */        else{
/* 15-                                  */            this.fieldX = 0;
/* 16-                                  */     }
/* 17-                                  */    }
/* 18-                                  */    @Override
/* 19-                                  */    public int methodX(int paramW, Boolean paramZ)
/* 20-                                  */    {
/* 21-                                  */        if (paramZ)
/* 22-ControlStatementBraces            */            fieldX = paramW;
/* 23-                                  */        else{
/* 24-                                  */            fieldX = 0;
/* 25-                                  */     }
/* 26-                                  */        return paramW + this.fieldX;
/* 27-                                  */     }
/* 28-                                  */}
```

Figure 7: New version

In Figure 8 we can see both ASTs, from the old and the new version. In this figure, the number of the nodes are meaningless. We can see the alerts linked to their nodes. The Figure 9 shows the kinds of nodes located near the alert that exists in both versions.

Prof. Márcio você perguntou o significado dos números. Não é nenhum, como está dito na frase acima. Eles estão aí só para ajudar na identificação dos labels

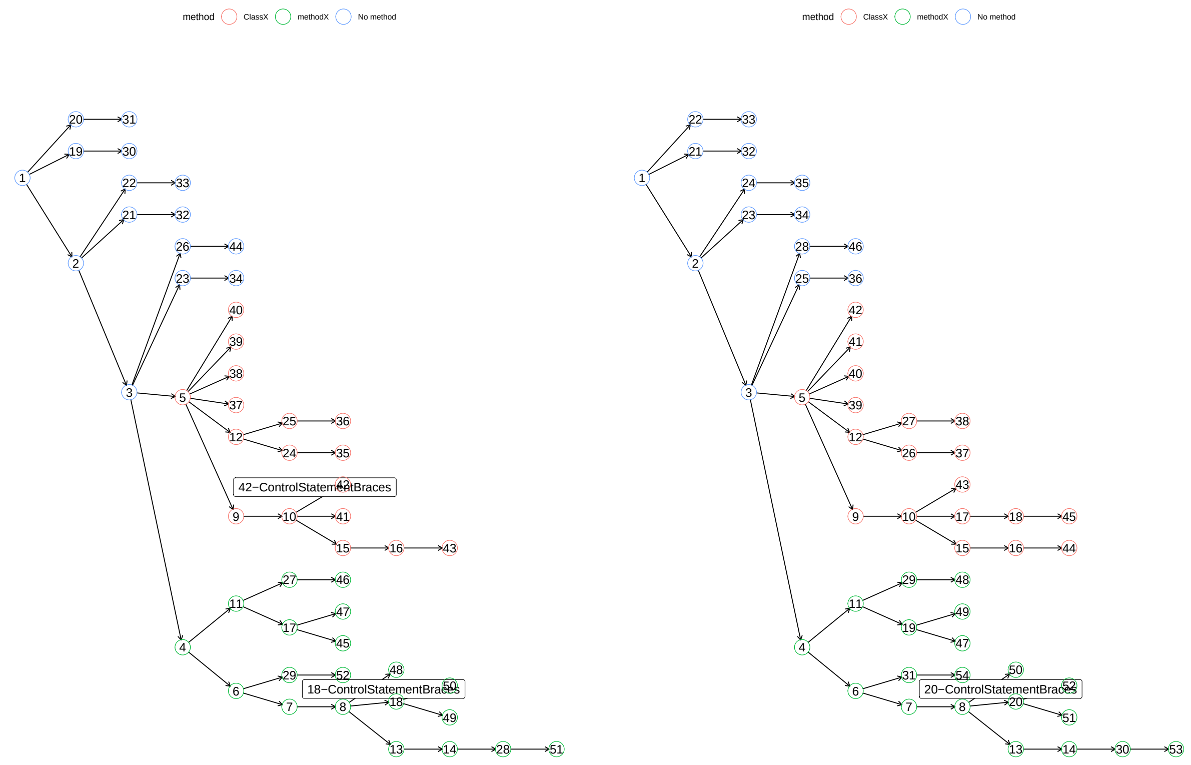
method
ClassX
methodX
No method
42−ControlStatementBraces
18−ControlStatementBraces
method
ClassX
methodX
No method
20−ControlStatementBraces

Figure 8: Abstract Syntax Trees. New and old versions, with alerts

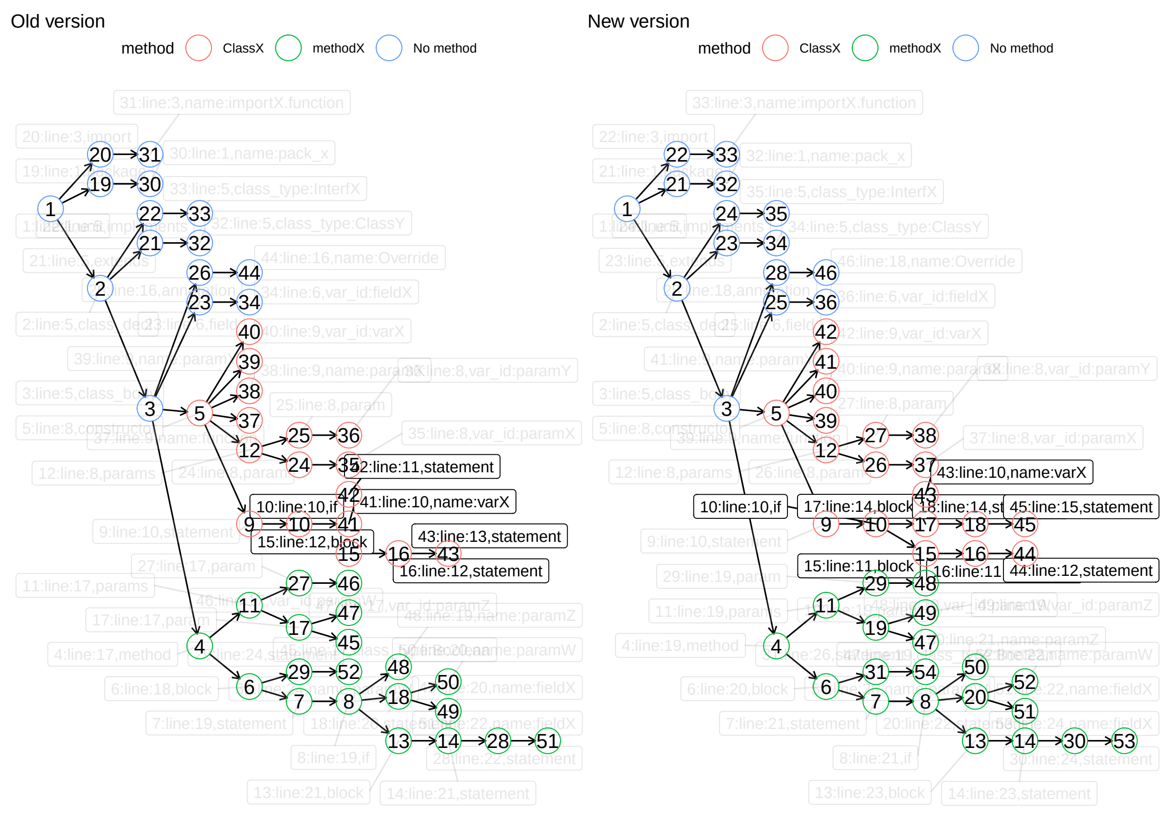

Figure 9: Abstract Syntax Trees compared. New and old versions, with the type of some AST nodes

Prof. Márcio, a diferença entre as árvores 8 e 9, que coloquei na legenda agora, é que uma mostra os alertas e a outra mostra os tipos de nós da AST localizados perto do alerta que ficou nas duas versões. Se colocasse tudo junto ia embolar

In Figure 10 the numbers inside the nodes are the number of the group: nodes with the same number are equivalent. We define that a node from an AST is equivalent to a node of the AST of the subsequent version if:

- Considering the relation between the lines of the two versions constructed as we saw in Section 3.1.3, the nodes in both trees begin and end in related lines.
- The nodes are of the same kind

Prof. Márcio, sim, agora ser o mesmo nó significa apenas estar nas mesmas linhas (levando em conta o diff) e ser do mesmo tipo. As outras carascterísticas vão ser indicadas nas features que serão construídas mais pra frente

Prof. Márcio, eu fui colocando várias árvores pra tentar fazer as coisas passo a passo. Apresentei primeiro os nós equivalentes e depois coloquei os alertas. Mas agora eu apresentei só uma dessas duas árvores, com uma legenda melhor

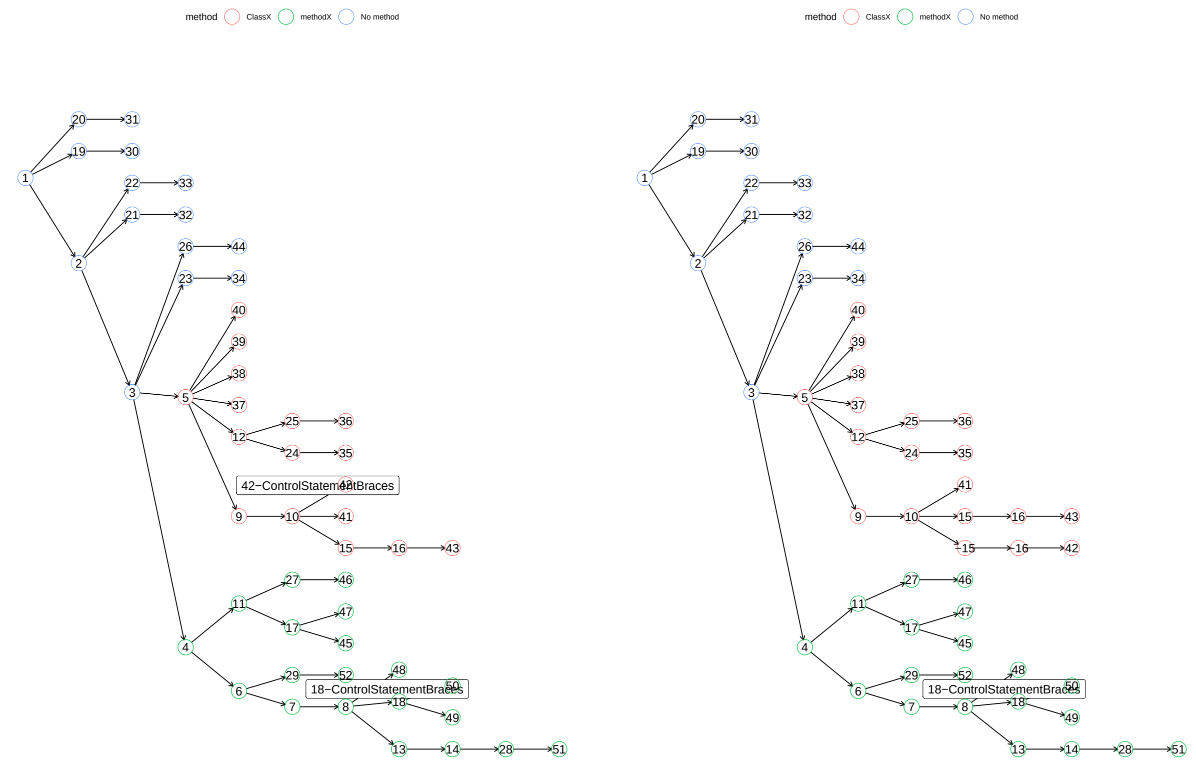
method
ClassX
methodX
No method
42−ControlStatementBraces
18−ControlStatementBraces
method
ClassX
methodX
No method
18−ControlStatementBraces

Figure 10: Abstract Syntax Tree. Nodes with the same number are equivalent

The path between the alert and the root of the AST can be seen in 11

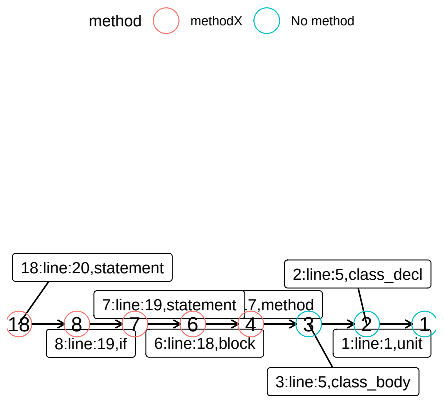

Figure 11: Abstract Syntax Tree

The algorithm generates a set of features for each pair of alerts (n, o) with one element n coming from the old version and one element o coming from the new version. The features do not lead to a direct conclusion. It´s necessary to create a heuristic or statistical learning algorithm that will decide the final verdict based on the features.

We propose the following list of features:

- Same Rule: a boolean indicator that tells if the alerts are of the same type
- Same Group ID: a boolean indicator that tells if the alerts are equivalent as defined in Figure **??**
- Same Method Group ID: a boolean indicator that tells if the alerts belong to the same method. We know the alert's method following the path from the alert´s node to the root. The first node of the kind "method" found in this path defines the alert's method. If this is the same for o and for n, then they belong to the same method.
- Same Method Name: a boolean indicator that tells if the alerts were found in a method with the same name.
- Same Block: a boolean indicator that shows if the o and n belong to the same block. It is defined the same way the "Same method" indicator is defined.
- Same Code: a boolean indicator that shows the nodes that generate the alert have the same programming code.
- Same Method Code: a boolean indicator that shows that the methods that contain the nodes that generate the alert have the same programming code.
- Line distance: o and n have a begin line $b(o)$ and $b(n)$ and an end line $e(n)$ and $e(n)$. Line distance is $abs(mean(b(o), e(o)) - mean(b(n), e(n)))$
- Normalized line distance (block size): this is the line distance but normalized by the size of the last common node.

- Normalized line distance (method size): this is the line distance but normalized by the size of the last common method (if there is no common method, it´s normalized by the side of the compilation unit).
- Normalized line distance (compilation unit size): this is the line distance but normalized by the size of the compilation unit.

Table 6 shows the combinations (n, o) in the example. There are $2 \cdot 1 = 2$ combinations whereas we have two alerts in the old version and one alert in the new one.

Table 6: Resulting features

Alert combination	Feature	Value
Line (Old version):20, Line (New version):22	Same Rule	TRUE
	Same Group ID	TRUE
	Same Method Group ID	TRUE
	Same Method Name	TRUE
	Same Block	TRUE
	Same Code	TRUE
	Same Method Code	TRUE
	Line Distance	0.00
	Line Distance Normalized by Block Size	0.00
	Line Distance Normalized by Method Size	0.00
	Line Distance Normalized by Compilation Unit Size	0.00
Line (Old version):11, Line (New version):22	Same Rule	TRUE
	Same Group ID	FALSE
	Same Method Group ID	FALSE
	Same Method Name	FALSE
	Same Block	FALSE
	Same Code	FALSE
	Same Method Code	FALSE
	Line Distance	10.00
	Line Distance Normalized by Block Size	0.43
	Line Distance Normalized by Method Size	0.37
	Line Distance Normalized by Compilation Unit Size	0.37

3.2.2 Example: Renaming method

In this example, the new and old versions have only one alert. The method in which the alert happens is renamed from MethodX to methodZ.

```java
/*  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);
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 10-                 */        if (varX == 0){
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 11-                 */            this.fieldX = 1;
/* 12-                 */        }
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 13-                 */        else{
/* 14-                 */            this.fieldX = 0;
/* 15-                 */        }
/* 16-                 */    }
/* 17-                 */    @Override
/* 18-                 */    public int methodX(int paramW, Boolean paramZ)
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 19-                 */    {
/* 20-                 */        if (paramZ)
/* 21-ControlStateme   */            fieldX = paramW;
/* 22-                 */        else{
/* 23-                 */            fieldX = 0;
/* 24-                 */        }
/* 25-                 */        return paramW + this.fieldX;
/* 26-                 */     }
/* 27-                 */}
```

```java
/*  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)
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 11-                 */        {
/* 12-                 */            this.fieldX = 1;
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 13-                 */        }
/* 14-                 */        else{
/* 15-                 */            this.fieldX = 0;
/* 16-                 */        }
/* 17-                 */    }
/* 18-                 */    @Override
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 19-                 */    public int methodZ(int paramW, Boolean paramZ)
/* 20-                 */    {
/* 21-                 */        if (paramZ)
/* 22-ControlStateme   */            fieldX = paramW;
/* 23-                 */        else{
/* 24-                 */            fieldX = 0;
/* 25-                 */        }
/* 26-                 */        return paramW + this.fieldX;
/* 27-                 */     }
/* 28-                 */}
```

Figure 12: Comparison between old and new version

In the Table 7 we can see that the features “Same Method Name” and “Same Method Group ID” are now FALSE.

Table 7: Resulting features: rename method example

Alert combination	Feature	Value
Line (Old version):21, Line (New version):22	Same Rule	TRUE
	Same Group ID	TRUE
	Same Method Group ID	FALSE
	Same Method Name	FALSE
	Same Block	TRUE
	Same Code	TRUE
	Same Method Code	FALSE
	Line Distance	0.00
	Line Distance Normalized by Block Size	0.00
	Line Distance Normalized by Method Size	0.00
	Line Distance Normalized by Compilation Unit Size	0.00

3.2.3 Example: including a statement before

In this example, a new statement is included before the alert.

```
/*  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-                 */        {
/* 12-                 */            this.fieldX = 1;
/* 13-                 */        }
/* 14-                 */        else{
/* 15-                 */            this.fieldX = 0;
/* 16-                 */        }
/* 17-                 */    }
/* 18-                 */    @Override
/* 19-                 */    public int methodZ(int paramW, Boolean paramZ)
/* 20-                 */    {
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 21-                 */        if (paramZ)
/* 22-ControlStateme   */            fieldX = paramW;
/* 23-                 */        else{
/* 24-                 */            fieldX = 0;
/* 25-                 */        }
/* 26-                 */        return paramW + this.fieldX;
/* 27-                 */     }
/* 28-                 */}
```

```
/*  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-                 */        {
/* 12-                 */            this.fieldX = 1;
          /* 13-                 */        }
/* 14-                 */        else{
/* 15-                 */            this.fieldX = 0;
/* 16-                 */        }
/* 17-                 */    }
/* 18-                 */    @Override
/* 19-                 */    public int methodZ(int paramW, Boolean paramZ)
/* 20-                 */    {
/* 21-                 */      paramT = 0;
/* 22-                 */        if (paramZ)
/* 23-ControlStateme   */            fieldX = paramW;
/* 24-                 */        else{
/* 25-                 */            fieldX = 0;
/* 26-                 */        }
/* 27-                 */        return paramW + this.fieldX;
/* 28-                 */     }
/* 29-                 */}
```

Figure 13: Comparison between old and new version

In the Table 8 we can see that the features are not affected by this new statement.

Table 8: Resulting features: statement included before

Alert combination	Feature	Value
Line (Old version):22, Line (New version):23	Same Rule	TRUE
	Same Group ID	TRUE
	Same Method Group ID	TRUE
	Same Method Name	TRUE
	Same Block	TRUE
	Same Code	TRUE
	Same Method Code	FALSE
	Line Distance	0.00
	Line Distance Normalized by Block Size	0.00
	Line Distance Normalized by Method Size	0.00
	Line Distance Normalized by Compilation Unit Size	0.00

3.2.4 Example: nesting the alert in an if statement

```
/*  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-                  */        {
/* 12-                  */            this.fieldX = 1;
/* 13-                  */        }
/* 14-                  */        else{
/* 15-                  */            this.fieldX = 0;
/* 16-                  */        }
/* 17-                  */    }
/* 18-                  */    @Override
/* 19-                  */    public int methodZ(int paramW, Boolean paramZ)
/* 20-                  */    {
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 21-                  */        if (paramZ)
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 22-ControlStateme    */            fieldX = paramW;
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 23-                  */        else{
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 24-                  */            fieldX = 0;
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 25-                  */        }
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 26-                  */        return paramW + this.fieldX;
/* 27-                  */     }
/* 28-                  */}
```

```
/*  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-                  */        {
/* 12-                  */            this.fieldX = 1;
          /* 13-                  */        }
/* 14-                  */        else{
/* 15-                  */            this.fieldX = 0;
/* 16-                  */        }
/* 17-                  */    }
/* 18-                  */    @Override
/* 19-                  */    public int methodZ(int paramW, Boolean paramZ)
/* 20-                  */    {
/* 21-                  */       if(paramZ == 0){
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 22-                  */            if (paramZ)
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 23-ControlStateme    */                fieldX = paramW;
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 24-                  */            else{
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 25-                  */                fieldX = 0;
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 26-                  */            }
/* 27-                  */       }
/* 28-                  */       else{
/* 29-                  */           fieldX = 1;
/* 30-                  */       }
/* 31-                  */        return paramW + this.fieldX;
/* 32-                  */     }
/* 33-                  */}
```

Figure 14: Comparison between old and new version

In the Table 9 we can see that the algorithm does not recognize the two nodes as equivalent, but other features can lead us to the conclusion that the alert is still open.

Table 9: Resulting features: statement included before

Alert combination	Feature	Value
Line (Old version):22, Line (New version):23	Same Rule	TRUE
	Same Group ID	FALSE
	Same Method Group ID	TRUE
	Same Method Name	TRUE
	Same Block	FALSE
	Same Code	TRUE
	Same Method Code	FALSE
	Line Distance	1.00
	Line Distance Normalized by Block Size	0.06
	Line Distance Normalized by Method Size	0.06
	Line Distance Normalized by Compilation Unit Size	0.03

3.2.5 Example: editing the line that generates the alert

```
/*  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-                 */        {
/* 12-                 */            this.fieldX = 1;
/* 13-                 */        }
/* 14-                 */        else{
/* 15-                 */            this.fieldX = 0;
/* 16-                 */        }
/* 17-                 */    }
/* 18-                 */    @Override
/* 19-                 */    public int methodZ(int paramW, Boolean paramZ)
/* 20-                 */    {
/* 21-                 */        if (paramZ)
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 22-ControlStateme   */            fieldX = paramW;
/* 23-                 */        else{
/* 24-                 */            fieldX = 0;
/* 25-                 */        }
/* 26-                 */        return paramW + this.fieldX;
/* 27-                 */     }
/* 28-                 */}
```

```
/*  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-                 */        {
/* 12-                 */            this.fieldX = 1;
          /* 13-                 */        }
/* 14-                 */        else{
/* 15-                 */            this.fieldX = 0;
/* 16-                 */        }
/* 17-                 */    }
/* 18-                 */    @Override
/* 19-                 */    public int methodZ(int paramW, Boolean paramZ)
/* 20-                 */    {
/* 21-                 */        if (paramZ)
/* 22-ControlStateme   */            fieldX = paramZ;
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 23-                 */        else{
/* 24-                 */            fieldX = 0;
/* 25-                 */        }
/* 26-                 */        return paramW + this.fieldX;
/* 27-                 */     }
/* 28-                 */}
```

Figure 15: Comparison between old and new version

In the Table 10 the nodes are not recognized as equivalent, but other features can lead us to the conclusion that the alert is still open.

Table 10: Resulting features: alert line edited

Alert combination	Feature	Value
Line (Old version):22, Line (New version):22	Same Rule	TRUE
	Same Group ID	FALSE
	Same Method Group ID	TRUE
	Same Method Name	TRUE
	Same Block	TRUE
	Same Code	FALSE
	Same Method Code	FALSE
	Line Distance	1.00
	Line Distance Normalized by Block Size	0.20
	Line Distance Normalized by Method Size	0.11
	Line Distance Normalized by Compilation Unit Size	0.04

3.2.6 Example: changing the order of the methods

We changed the order of the methods in two ways

```
/*  1-                 */package pack_x;
/*  2-                 */
/*  3-                 */import importX.function;
/*  4-                 */
/*  5-                 */class ClassX extends ClassY implements InterfX {
/*  6-                 */    private long fieldX;
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*  7-                 */
/* 18-                 */    @Override
/*  8-                 */    ClassX(int paramX, double paramY) {
/* 19-                 */    public int methodZ(int paramW, Boolean paramZ)
/*  9-                 */        int varX = function(paramX, paramY);
/* 20-                 */    {
/* 10-                 */        if (varX == 0)
/* 21-                 */        if (paramZ)
/* 11-                 */        {
/* 22-ControlStateme   */            fieldX = paramW;
/* 12-                 */            this.fieldX = 1;
/* 23-                 */        else{
/* 13-                 */        }
/* 24-                 */            fieldX = 0;
/* 14-                 */        else{
/* 25-                 */        }
/* 15-                 */            this.fieldX = 0;
/* 26-                 */        return paramW + this.fieldX;
/* 16-                 */        }
/* 27-                 */     }
/* 17-                 */    }
/* 28-                 */}
```

```
/*  1-                 */package pack_x;
/*  2-                 */
/*  3-                 */import importX.function;
/*  4-                 */
/*  5-                 */class ClassX extends ClassY implements InterfX {
/*  6-                 */    private long fieldX;
/*  7-                 */
/*  8-                 */    @Override
/*  9-                 */    public int methodZ(int paramW, Boolean paramZ)
/* 10-                 */    {
/* 11-                 */        if (paramZ)
/* 12-ControlStateme   */            fieldX = paramW;
/* 13-                 */        else{
/* 14-                 */            fieldX = 0;
/* 15-                 */        }
/* 16-                 */        return paramW + this.fieldX;
/* 17-                 */     }
/* 18-                 */
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
        /* 19-                 */    ClassX(int paramX, double paramY) {
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
       /* 20-                 */        int varX = function(paramX, paramY);
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 21-                 */        if (varX == 0)
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 22-                 */        {
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 23-                 */            this.fieldX = 1;
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
          /* 24-                 */        }
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 25-                 */        else{
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 26-                 */            this.fieldX = 0;
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 27-                 */        }
/*   -                 *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 28-                 */    }
/* 29-                 */}
```

Figure 16: Comparison between old and new version

In the Table 11 the nodes are not recognized as equivalent, but other features can lead us to the conclusion that the alert is still open.

Table 11: Resulting features: changed methods order

Alert combination	Feature	Value
Line (Old version):22, Line (New version):12	Same Rule	TRUE
	Same Group ID	FALSE
	Same Method Group ID	FALSE
	Same Method Name	TRUE
	Same Block	FALSE
	Same Code	TRUE
	Same Method Code	TRUE
	Line Distance	15.00
	Line Distance Normalized by Block Size	0.44
	Line Distance Normalized by Method Size	0.39
	Line Distance Normalized by Compilation Unit Size	0.39

```
/*  1-                  */package pack_x;
/*  2-                  */
/*  3-                  */import importX.function;
/*  4-                  */
/*  5-                  */class ClassX extends ClassY implements InterfX {
/*  6-                  */    private long fieldX;
/*  7-                  */
/*  8-                  */    @Override
/*  9-                  */    public int methodZ(int paramW, Boolean paramZ)
/* 10-                  */    {
/* 11-                  */        if (paramZ)
/* 12-ControlStateme    */            fieldX = paramW;
/* 13-                  */        else{
/* 14-                  */            fieldX = 0;
/* 15-                  */        }
/* 16-                  */        return paramW + this.fieldX;
/* 17-                  */     }
/* 18-                  */
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 19-                  */    ClassX(int paramX, double paramY) {
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 20-                  */        int varX = function(paramX, paramY);
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 21-                  */        if (varX == 0)
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 22-                  */        {
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 23-                  */            this.fieldX = 1;
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 24-                  */        }
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 25-                  */        else{
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 26-                  */            this.fieldX = 0;
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 27-                  */        }
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/* 28-                  */    }
/* 29-                  */}
```

```
/*  1-                  */package pack_x;
/*  2-                  */
/*  3-                  */import importX.function;
/*  4-                  */
/*  5-                  */class ClassX extends ClassY implements InterfX {
/*  6-                  */    private long fieldX;
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*   -                  *//*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*/
/*  7-                  */
/* 18-                  */    @Override
        /*  8-                  */    ClassX(int paramX, double paramY) {
/* 19-                  */    public int methodZ(int paramW, Boolean paramZ)
       /*  9-                  */        int varX = function(paramX, paramY);
/* 20-                  */    {
 /* 10-                  */        if (varX == 0)
/* 21-                  */        if (paramZ)
 /* 11-                  */        {
/* 22-ControlStateme    */            fieldX = paramW;
 /* 12-                  */            this.fieldX = 1;
/* 23-                  */        else{
          /* 13-                  */        }
/* 24-                  */            fieldX = 0;
 /* 14-                  */        else{
/* 25-                  */        }
 /* 15-                  */            this.fieldX = 0;
/* 26-                  */        return paramW + this.fieldX;
 /* 16-                  */        }
/* 27-                  */     }
 /* 17-                  */    }
 /* 28-                  */}
```

Figure 17: Comparison between old and new version

In the Table 12 the nodes are not recognized as equivalent, but other features can lead us to the conclusion that the alert is still open.

Table 12: Resulting features: changed methods order 2

Alert combination	Feature	Value
Line (Old version):12, Line (New version):22	Same Rule	TRUE
	Same Group ID	FALSE
	Same Method Group ID	FALSE
	Same Method Name	TRUE
	Same Block	FALSE
	Same Code	TRUE
	Same Method Code	TRUE
	Line Distance	15.00
	Line Distance Normalized by Block Size	0.44
	Line Distance Normalized by Method Size	0.39
	Line Distance Normalized by Compilation Unit Size	0.39