The University of Texas at Austin
SEAL - Software Evolution and Analysis Laboratory

Additional Artifacts for “Detecting Anomalies in Manual Refactoring”

Everton L. G. Alves

Myoungkyu Song
Miryung Kim
Patrícia D. L. Machado
Tiago Massoni
(Collaborators)

Austin, Texas, USA

Contents

Chapter 1

Templates for Detecting Missing edits

REFCHECKER uses templates to detect missing edits in manual refactorings that might lead to behavior changes. Tables 1.1 and 1.1 present the template rules that REFCHECKER checks with brief descriptions. The rules are presented in pseudo code. The following auxiliary functions are defined in order to simplify the rules presentation:

- `getClass(P, m)` returns the containing class of method m to be refactored.
- `getCallers(m)` returns all callers of m.
- `isAssociatedWithAField(m)` verifies whether the method m accesses any field declared in the same class.
- `checkBindingProblem (m1, m2)` verifies whether all method and variable references are identical between `m1` and `m2`.
- `verifyAccessibilityChange (m1, m2)` verifies whether method `m2` is visible to method `m1`.
- `haveDependences(m, stms)` verifies whether the remaining statements in method m after the extraction of `stmts` are dependent on any statements within extracted code `stms`.
- `getStatements(m, [beginLine;endLine])` returns the statements that are in between the range of lines specified from beginLine to endLine. In case of an empty range, it returns all statements from m.

Table 1.1: The refactoring change rules of the **RefChecker**'s templates (Part 1).

Move Method (P: original version, P_r: modified version, $m1_o$: method to be refactored, $m2_n$: newly added method)		
1	C_p ={}; c_o = getClass(P, $m1_o$); C_p = C_p ∪ {<'Remove Functionality', c_o>}	There must be a method deleting in the original version P.
2	c_o = getClass(P_r, $m1_o$); C_p = C_p ∪ {<'Add Functionality', c_o>}	There must be a new method in the modified version P_r.
3	C = getCallers(P, $m1_o$); **FOREACH** (c **in** C) **DO** **IF** (isAssociatedWithField($m1_o$)) **THEN** C_p = C_p ∪ {<'Change Attribute Type', c>}; ELSE C_p = C_p ∪ {<'Update Statement', c>};	For all callers of $m1_o$, if the method call is associated to a field, there must be an attribute type change in P, and a statement update otherwise.
4	C = getCallers(P, $m1_o$); **FOREACH** (c **in** C) **DO** m = getMethod(P_r, c); **IF** (checkBindingProblem(c, m)) **THEN** C_p = C_p ∪ {<'Binding Problem', c>};	All callers of $m1_o$ in the modified version (m from P_r) must preserve all method and variable references from the original version P.
5	**IF** (checkBindingProblem($m1_o$, $m2_n$)) **THEN** C_p = C_p ∪ {<'Binding Problem', $m1_o$>};	All method and variables references in $m1_o$ must remain the same in the modified version P_r.
6	C = getCallers(P, $m1_o$); **FOREACH** (c **in** C) **DO** m = getMethod(P_r, c); **IF** (verifyAccessibilityChange (c, m)) **THEN** C_p = C_p ∪ {<'Change Visibility', m>};	Added method $m2_n$ must be visible to the callers of the removed method $m1_o$.
Pull Up Method: rules 1, 2, 4, and 5		
Push Down Method: rules 1, 2, 4, and 5		
Extract Method (P: original version, P_r: modified version, $m1_o$: method to be refactored, $m2_n$: extracted method, $[startLine, endLine]$: portion to be extracted)		
7	C_p ={}; c_o = getClass(P, $m1_o$); C_p = C_p ∪ {<'Add Functionality', c_o>}	Modified version P_r must include a method not existing in original version P.
8	STM_o = getStatements ($m1_o$, $[startLine, endLine]$) **IF** (haveDependences (P, STM_o)) **THEN** C_p = C_p ∪ {<'Update Statement', $m1_o$>};	If any extracted statements (STM_o) modify the value of variable(s) used in the rest of the method, the modified method must have a new variable update. The updated variable must be associated to the return value of a calling to the new method.
9	**IF** (**NOT** haveDependences (P, STM_o)) **THEN** C_p = C_p ∪ {<'Insert Statement', $m1_o$>};	If Rule 8 is not applicable, there must be a new statement related to the calling of the the new method in the modified version.
10	**FOREACH** (s **in** STM_o) **DO** C_p = C_p ∪ {<'Delete Statement', s, $m1_o$>}; C_p = C_p ∪ {<'Insert Statement', s, $m2_n$>};	For each extracted statement, there must be a deleted statement in the original method $m1_o$ and an inserted statement (same statement) in the modified method $m2_n$.
11	C = getCallers(P, $m1_o$); **FOREACH** (c **in** C) **DO** m = getMethod(P_r, c); **IF** (checkBindingProblem(c, m)) **THEN** C_p = C_p ∪ {<'Binding Problem', c>};	All callers of $m1_o$ in the modified version (m from P_r) must preserve all method and variable references from the original version P.
11	C = getCallers(P, $m2_n$); **FOREACH** (c **in** C) **DO** m = getMethod(P_r, c); **IF** (checkBindingProblem(c, m)) **THEN** C_p = C_p ∪ {<'Binding Problem', c>};	All callers of with similar signatures (same name but different parameters) $m2_n$ in the modified version must preserve all method and variable references from the original version P.
12	$unionSet$ = $m1_o$ ∪ $m2_n$; **IF** (checkBindingProblem($m1_o$, $unionSet$)) **THEN** C_p = C_p ∪ {<'Binding Problem', $m1_o$>};	All methods and variable references in $m1_o$ must be the same as the combination of the original method in the new version $m1_o$ and the newly added one $m2_n$, except by the changes performed in Rules 8 or 9.

Table 1.2: The refactoring change rules of the **RefChecker**’s templates (Part 2).

Inline Method (P: original code, P_r: modified version, $m1_o$: method to be inlined)		
13	C_p ={}; c_o = getClass(P, $m1_o$); C_p = C_p ∪ {<‘Remove Functionality’, c_o>}	There must be a deleted method in the modified version P_r.
14	STM_o = getStatements ($m1_o$, []); C = getCallers(P, $m1_o$); **FOREACH** (c **in** C) **DO** $m2_n$ = getMethod(P_r, c); **IF** (isNotVoid (c)) **THEN** C_p = C_p ∪ {<‘Update Statement’, c>}; FOREACH (s in STM_o) DO C_p = C_p ∪ {<‘Insert Statement’, s, c>};	If the inlined method $m1_o$ has a return type, there must be an updated statement in each of its callers. Also, for all callers, there must exist a sequence of inserted statement inlined from $m1_o$.
-	Check rule 4	See 4

Rename Method (P: original code, P_r: modified version, $m1_o$: method to be renamed)		
15	C_p ={}; c_o = getClass(P, $m1_o$); C_p = C_p ∪ {<‘Rename Method’, c_o>}	There must be a method in P_r that had its signature changed when compared to the original version P.
16	C = getCallers(P, $m1_o$); **FOREACH** (c **in** C) **DO** C_p = C_p ∪ {<‘Update Statement’, c>};	For all callers to $m1_o$, there must be an updated statement in modified version P_r.

Chapter 2

RefSeparator Bug Conditions

Table 2.1 describes the bug conditions checked by REFSEPARATOR in order to ensure that all applied refactorings are performed properly. If any of those condition is true, REFSEPARATOR will not apply the edit and a warning message is generated to the user. The first column indicates the bug number in the Eclipse bug tracker, and the second column shows a brief description of the situation to be checked.

Table 2.1: The bug conditions checked in REFSEPARATOR.

Bug	**C1: class under refactorings; C2: target class; m: method under refactorings**
	Rename Method
313041	The method m is to be renamed, there is a method in a superclass of C with the same signature of m, but with a broader visibility.
	Push Down Method
320115	The method m is to be pushed down, which directly calls a method that is invisible from the target class.
348278	The method m is to be pushed down, which contains a method call using the keyword, this.
356698	The method m is to be pushed down, which contains a super access to a method that is overridden in class $C1$.
355322	The method m is to be pushed down, which contains a super access to a method that is overloaded in class $C1$.
290618	The method m is to be pushed down, which contains a call to a method that is overridden in the target class $C2$.
355324	The method m is to be pushed down, which contains a call to a method that is overloaded in the target class $C2$.
195003	The method m is to be pushed down, which contains a field access using the keyword, this.
195004	The method m is to be pushed down, whose callee invokes another method by the origin class (e.g., new ClassX().foo().)
	Pull Up Method
355325	The method m is to be pulled, which directly calls a method that is invisible from the target class.
319926	The method m is to be pulled, which contains a method call using the keyword, this.
355326	The method m is to be pulled, which contains an access to a method that is overloaded in class $C1$.
316831	The method m is to be pulled, which contains an access to a method that is overridden in class $C1$.
290615	The method m is to be pulled, which contains a super access to a method that is overridden in class $C1$.
195005	The method m to be pulled, which accesses a field that is not visible from the target class.
	Move Method
356689-356688	The method m is to moves, which is part of an overloading.

Chapter 3

Dataset of Missing Edits

To assess REFDISTILLER effectiveness for detecting refactoring anomalies, we use a data set that is identified in Soares et al.'s prior work [1].

In the following subsections we present this data set. Each subject is a pair of Java programs (p_1, p_2), where p_1 is an original version and the program p_2 is p_1 after a problematic refactoring. All subject programs in the data set are free of compilation errors, and p_2 contains at least one missing refactoring edit. Code insertion is marked with '+', deletion with '-'. Together with the code snippets we present a brief description of each anomaly.

3.1 Extract Method

- **EM_1**

```
public class A {
    public int m(boolean b) {
        int x = 42;
        try {
            if (b) {
                x = 23;
                throw new
    Exception();
            }
        } catch (Exception e) {
            return x;
        }
        return x;
    }
    public int test() {
        return m(true);
    }
}
```

(a) Original version.

```
  public class A {
      public int m(boolean b) {
          int x = 42;
          try {
-                 if (b) {
-                     x = 23;
-                     throw new
    Exception();
-                 }
+             x = n(b, x);
          } catch (Exception e) {
              return x;
          }
          return x;
      }
+     public int n(boolean b, int x)
    throws Exception {
+         if (b) {
+             x = 23;
+             throw new Exception();
+         }
+         return x;
+     }
      public int test() {
          return m(true);
      }
  }
```

(b) Target Version.

Error Type: Variable states modified due to change in exception handling.

Error: Incorrect dataflow analysis. There can be a state inconsistency when an exception is thrown. The *x* variable, instead of assuming the 23 value, it returns to its previous state, 42, in the target version.

3.2 Move Method

- **MM_1**

```java
package p1;
public class A {
     public C c;
     public long k(){
          return 29;
     }
}
package p1;
public class B{
    protected long k() {
     return 18;
    }
}
package p1;
public class C extends B {
     public long test(){
          return k();
     }
}
```

(a) Original version.

```java
package p1;
public class A {
     public C c;
-     public long k(){
-          return 29;
-     }
}
package p1;
public class B{
    protected long k() {
     return 18;
    }
}
package p1;
public class C extends B {
     public long test(){
          return k();
     }
+     public long k(){
+          return 29;
+     }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in C.test().

- **MM_2**

```
package p1;
public class A {
     public B b;
     protected long k(int a){
          return 4;
     }
}
package p1;
public class B{
    private long k(long a) {
     return 17;
    }
    public long test(){
     return k(2);
    }
}
```

(a) Original version.

```
  package p1;
  public class A {
       public B b;
-            protected long k(int a){
                  return 4;
-            }
  }
  package p1;
  public class B{
       private long k(long a) {
       return 17;
      }
      public long test(){
       return k(2);
      }
+      protected long k(int a){
+       return 4;
+      }
  }
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **MM_3**

```
package p1;
public class A {
      public C c;
      public long k(int a){
            return 49;
      }
      protected long k(long a){
            return 30;
      }
}
package p1;
public class B extends A {
      public long test(){
            return super.k(2);
      }
}
package p1;
public class C {
}
```

(a) Original version.

```
package p1;
public class A {
      public C c;
-            public long k(int a){
                   return 49;
-             }
      protected long k(long a){
            return 30;
      }
}
package p1;
public class B extends A {
      public long test(){
            return super.k(2);
      }
}
package p1;
public class C {
+      public long k(int a){
+            return 49;
+      }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **MM_4**

```
package p1;
public class A {
    public B b;
    protected long k(int a){
        return 32;
    }
    protected long k(long a){
        return 47;
    }
    public long test(){
        return k(2);
    }
}
package p1;
public class B {
}
```

(a) Original version.

```
package p1;
public class A {
    public B b;
-        protected long k(int a){
-            return 32;
-        }
    protected long k(long a){
        return 47;
    }
    public long test(){
        return k(2);
    }
}
package p1;
public class B{
+    protected long k(int a){
+        return 32;
+    }
}
```

(b) Target Version.

Error Type: Missing updates in callers/Modified method calls due to incorrect overloading/overriding.

Error: Missing update in A.test() / Binding change in A.test().

- **MM_5**

```
package p1;
import p2.A0;
public class ClassId0 extends
    ClassId1{
     public long methodid0(){
          return new A0().m0(2);
     }
}
package p1;
public class ClassId1 {
}
package p2;
import p1.ClassId0;
public class A0 extends ClassId0{
     public ClassId0 fieldid0 =
    null;
     protected long m0(int a){
          return 0;
     }
     public long m0(long a){
          return 1;
     }
}
```

(a) Original version.

```
  package p1;
  import p2.A0;
  public class ClassId0 extends
      ClassId1{
       public long methodid0(){
            return new A0().m0(2);
       }
+      protected long m0(int a){
+           return 0;
+      }
  }
  package p1;
  public class ClassId1 {
  }
  package p2;
  import p1.ClassId0;
  public class A0 extends ClassId0{
       public ClassId0 fieldid0 = null;
-      protected long m0(int a){
-           return 0;
-      }
       public long m0(long a){
            return 1;
       }
  }
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in ClassId0.methodid0().

- **MM_6**

```
package p1;
import p2.*;
public class A0 extends ClassId0{
     public ClassId0 fieldid0 = null;
     public long methodid0(){
          return super.m0(2);
     }
     public long m0(int a){
          return 0;
     }
}
package p2;
public class ClassId0 extends ClassId1{
}
package p2;
public class ClassId1 {
     public long m0(int a){
          return 1;
     }
}
```

(a) Original version.

```
package p1;
import p2.*;
public class A0 extends ClassId0{
     public ClassId0 fieldid0 = null;
     public long methodid0(){
          return super.m0(2);
     }
-     public long m0(int a){
-          return 0;
-     }
}
package p2;
public class ClassId0 extends ClassId1{
+     public long m0(int a){
+          return 0;
+     }
}
package p2;
public class ClassId1 {
     public long m0(int a){
          return 1;
     }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in A0.methodid0().

- **MM_7**

```java
package p1;
import p2.C;
public class B extends C{
    public long k(){
        return m(2);
    }
}
package p2;
import p1.B;
public class A {
    public B f = null;
    private long m(int a){
        return 0;
    }
}
package p2;
public class C extends A{
    protected long m(long a){
        return 1;
    }
}
```

(a) Original version.

```java
package p1;
import p2.C;
public class B extends C{
+     public long m(int a){
+         return 0;
+     }
    public long k(){
        return m(2);
    }
}
package p2;
public class C extends A{
    protected long m(long a){
        return 1;
    }
}
package p2;
import p1.B;
public class A {
    public B f = null;
-       private long m(int a){
-           return 0;
-       }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.k().

- **MM_8**

```
package p1;
public class A {
     public B b;
     protected long k(int a) {
          return 2;
     }
}
package p1;
public class B {
     private long k(long a) {
          return 1;
     }
     public long test() {
          return k(2);
     }
}
```

(a) Original version.

```
package p1;
public class A {
     public B b;
-     protected long k(int a) {
-          return 2;
-     }
}
package p1;
public class B {
+     protected long k(int a) {
+         return 2;
+     }
     private long k(long a) {
         return 1;
     }
     public long test() {
        return k(2);
     }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

3.3 Pull Up Method

- **PUM_1**

```
package p1;
public class A {
      public int k (long i){
            return 10;
      }
}
package p1;
public class B extends A{
      public int k(int i){
            return 20;
      }
      public int test(){
            return new A().k(2);
      }
}
```

(a) Original version.

```
package p1;
public class A {
      public int k (long i){
            return 10;
      }
+      public int k(int i){
+            return 20;
+      }
}
package p1;
public class B extends A{
-           public int k(int i){
-                 return 20;
-           }
      public int test(){
            return new A().k(2);
      }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PUM_2**

```
package p1;
public class A {
    public int k (long l){
        return 10;
    }
    private int k(int l){
        return 20;
    }
}
package p1;
public class B extends A{
    public int m(){
        return k(2);
    }
    public int test(){
        return m();
    }
}
```

(a) Original version.

```
package p1;
public class A {
    public int k (long l){
        return 10;
    }
    private int k(int l){
        return 20;
    }
+    public int m(){
+        return k(2);
+    }
}
package p1;
public class B extends A{
-        public int m(){
-            return k(2);
-        }
    public int test(){
        return m();
    }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PUM_3**

```
package p1;
public class A {
      public int k (){
            return 10;
      }
}
package p1;
public class B extends A{
      public int test(){
            return k();
      }
}
package p1;
public class C extends B{
      public int k(){
            return 20;
      }
}
```

(a) Original version.

```
package p1;
public class A {
      public int k (){
            return 10;
      }
}
package p1;
public class B extends A{
      public int test(){
            return k();
      }
+      public int k(){
+            return 20;
+      }
}
package p1;
public class C extends B {
-      public int k(){
-            return 20;
-      }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PUM_4**

```java
package p1;
public class A {
    public int k (){
        return 10;
    }
}
package p1;
public class B extends A{
    public int k(){
        return 20;
    }
    public int test(){
        return m();
    }
    public int m(){
        return super.k();
    }
}
```

(a) Original version.

```java
package p1;
public class A {
    public int k (){
        return 10;
    }
+    public int m(){
+        return this.k();
+    }
}
package p1;
public class B extends A{
    public int k(){
        return 20;
    }
    public int test(){
        return m();
    }
-        public int m(){
-            return super.k();
-        }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PUM_5**

```java
package p1;
public class A {
    private int k (){
        return 10;
    }
}
package p1;
public class B extends A{
    public int k(){
        return 20;
    }
    public int m(){
        return k();
    }
    public int test(){
        return m();
    }
}
```

(a) Original version.

```java
package p1;
public class A {
    private int k (){
        return 10;
    }
+    public int m(){
+        return k();
+    }
}
package p1;
public class B extends A{
    public int k(){
        return 20;
    }
-        public int m(){
-            return k();
-        }
    public int test(){
        return m();
    }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PUM_6**

```
package p1;
public class A {
}
package p1;
public class B extends A{
      protected long m(long a){
            return 1;
      }
      public long test(){
            return m(2);
      }
}
package p1;
public class C extends A{
      protected long m(int a){
            return 2;
      }
}
```

(a) Original version.

```
package p1;
public class A {
+     protected long m(int a){
+           return 2;
+     }
}
package p1;
public class B extends A{
      protected long m(long a){
            return 1;
      }
      public long test(){
            return m(2);
      }
}
package p1;
public class C extends A{
-           protected long m(int a){
-                 return 2;
-           }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PUM_7**

```
package p1;
public class A {
    protected long k(int a){
        return 10;
    }
    public long k(long a){
        return 20;
    }
}
package p1;
import p2.B;
public class C extends B{
    public long m(){
        return new A().k(2);
    }
    public long test(){
        return m();
    }
}
package p2;
public class B extends A{
}
```

(a) Original version.

```
package p1;
public class A {
    protected long k(int a){
        return 10;
    }
    public long k(long a){
        return 20;
    }
}
package p1;
import p2.B;
public class C extends B{
-    public long m(){
-        return new A().k(2);
-    }
    public long test(){
        return m();
    }
}
package p2;
import p1.A;
public class B extends A{
+    public long m(){
+        return new A().k(2);
+    }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in C.test().

- **PUM_8**

```
package p1;
public class A extends B{
     public long n(){
          return test();
     }
     public long test(){
          return k(2);
     }
}
package p1;
import p2.C;
public class B extends C{
     private long k(int a){
          return 1;
     }
}
package p2;
public class C{
     protected long k(long a){
          return 0;
     }
}
```

(a) Original version.

```
package p1;
public class A extends B{
     public long n(){
          return test();
     }
-          public long test(){
-               return k(2);
-          }
}
package p1;
import p2.C;
public class B extends C{
     private long k(int a){
          return 1;
     }
+     public long test(){
+          return k(2);
+     }
}
package p2;
public class C{
     protected long k(long a){
          return 0;
     }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in A.test().

Template Rule:

- **PUM_9**

```
package p1;
public class A {
     int m(Object o) {
         return 43;
       }
}
package p1;
public class B extends A {
     int m(String s) {
         return 23;
      }
}
package p1;
public class C {
      public int test() {
               return new
    A().m("abc");
     }
}
```

(a) Original version.

```
package p1;
public class A {
+     int m(String s) {
+         return 23;
+     }
     int m(Object o) {
         return 43;
       }
}
package p1;
public class B extends A {
-     int m(String s) {
-         return 23;
-     }
}
package p1;
public class C {
      public int test() {
              return new A().m("abc");
      }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in C.test().

- **PUM_10**

```java
package p1;
public class A {
    public int m(Object o) {
        return 42;
    }
    public int test() {
        return new B().m("abc");
     }
}
package p1;
public class B extends A {
}
package p1;
public class C extends B{
    public int m(String s) {
        return 23;
    }
}
```

(a) Original version.

```diff
package p1;
public class A {
    public int m(Object o) {
        return 42;
      }
      public int test() {
        return new B().m("abc");
      }
}
package p1;
public class B extends A {
+     public int m(String s) {
+         return 23;
+     }
}
package p1;
public class C extends B{
-         public int m(String s) {
-             return 23;
-         }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in A.test().

3.4 Push Down Method

- **PUM_1**

```
package p1;
public class A {
        public void k() {
                System.out.println(23);
        }
}
package p1;
public class B extends A {
        public void m() {
                super.k();
        }
        public void k() {
                System.out.println(42);
        }
}
package p1;
public class C extends B {
}
```

(a) Original version.

```
package p1;
public class A {
        public void k() {
                System.out.println(23);
        }
}
package p1;
public class B extends A {
-               public void m() {
-                       super.k();
-               }
      public void k() {
        System.out.println(42);
      }
}
package p1;
public class C extends B {
+       public void m() {
+               super.k();
+       }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.m().

- **PDM_2**

```java
package p1;
public class A {
      public int k() {
            return 23;
      }
}
package p1;
public class B extends A {
      public int k() {
            return 42;
      }
      public int m() {
            return super.k();
      }
}
package p1;
public class C extends B {
      public int teste(){
            return m();
      }
}
```

(a) Original version.

```diff
package p1;
public class A {
      public int k() {
            return 23;
      }
}
package p1;
public class B extends A {
      public int k() {
            return 42;
      }
-             public int m() {
-                   return super.k();
-             }
}
package p1;
public class C extends B {
      public int teste(){
            return m();
      }
+       public int m() {
+             return super.k();
+       }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.m().

- **PDM_3**

```java
package p1;
public class A {
     protected int x = 23;
     public int m() {
          return x;
     }
}
package p1;
public class B extends A {
    protected int x = 42;
    public int test() {
     return new B().m();
    }
}
```

(a) Original version.

```diff
 package p1;
 public class A {
      protected int x = 23;
-           public int m() {
-                return x;
-           }
 }
 package p1;
 public class B extends A {
      protected int x = 42;
      public int test() {
           return new B().m();
      }
+      public int m() {
+           return x;
+      }
 }
```

(b) Target Version.

Error Type: Modified field references due to incorrect field overloading/overriding.

Error: Binding change in m(), w.r.t. variable *x*.

- **PDM_4**

```java
package p1;
public class A {
    public int k(long i) {
        return 23;
    }
    public int m() {
        return k(2);
    }
}
package p1;
public class B extends A {
    public int k(int i) {
        return 42;
    }
    public int test() {
        return m();
    }
}
```

(a) Original version.

```diff
package p1;
public class A {
    public int k(long i) {
        return 23;
    }
-       public int m() {
-           return k(2);
-       }
}
package p1;
public class B extends A {
    public int k(int i) {
        return 42;
    }
    public int test() {
        return m();
    }
+    public int m() {
+        return k(2);
+    }
}
```

(b) Target Version.

Error Type: Missing updates in callers/Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PDM_5**

```
package p1;
public class A {
    public int k() {
        return 23;
    }
}
package p1;
public class B extends A {
    public int k() {
        return 42;
    }
    public int m() {
        return super.k();
    }
}
package p1;
public class C extends B{
    public int test(){
        return m();
    }
}
```

(a) Original version.

```
package p1;
public class A {
    public int k() {
        return 23;
    }
}
package p1;
public class B extends A {
    public int k() {
        return 42;
    }
-       public int m() {
-           return super.k();
-       }
}
package p1;
public class C extends B{
    public int test(){
        return m();
    }
+    public int m() {
+        return super.k();
+    }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in C.test().

- **PDM_6**

```
package p1;
public class A {
    public int k(long l) {
        return 23;
    }
}
package p1;
public class B extends A {
    public int k(int i) {
        return 42;
    }
    public int m() {
        return super.k(2);
    }
}
package p1;
public class C extends B{
    public int test(){
        return m();
    }
}
```

(a) Original version.

```
package p1;
public class A {
    public int k(long l) {
        return 23;
    }
}
package p1;
public class B extends A {
    public int k(int i) {
        return 42;
    }
-    public int m() {
-        return super.k(2);
-    }
}
package p1;
public class C extends B{
    public int test(){
        return m();
    }
+    public int m() {
+        return super.k(2);
+    }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in C.test().

- **PDM_7**

```java
package p1;
public class A {
     public int k() {
          return 23;
     }
}
package p1;
public class B extends A {
     public int k() {
          return 42;
     }
}
package p1;
public class C extends B{
     public int test() {
          return super.k();
     }
}
```

(a) Original version.

```java
package p1;
public class A {
     public int k() {
          return 23;
     }
}
package p1;
public class B extends A {
-          public int k() {
-               return 42;
-          }
}
package p1;
public class C extends B{
+      public int k() {
+           return 42;
+      }
     public int test() {
          return super.k();
     }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in C.test().

- **PDM_8**

```java
package p1;
public class A {
    public int k(int i) {
        return 23;
    }
}
package p1;
public class B extends A {
    public int k(long l) {
        return 42;
    }
}
package p2;
public class C extends B{
    public int test() {
        return super.k(0);
    }
}
```

(a) Original version.

```java
  package p1;
  public class A {
      public int k(int i) {
          return 23;
      }
  }
  package p1;
  public class B extends A {
-     public int k(long l) {
-         return 42;
-     }
  }
  package p2;
  public class C extends B{
      public int test() {
          return super.k(0);
      }
+     public int k(long l) {
+         return 42;
+     }
  }
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in C.test().

- **PDM_9**

```
package p1;
public class A {
     public int k(long l) {
          return 23;
     }
     public int m(){
          return k(2);
     }
}
package p1;
public class B extends A {
     public int k(int i){
          return 42;
     }
     public int test(){
          return m();
     }
}
```

(a) Original version.

```
package p1;
public class A {
     public int k(long l) {
          return 23;
     }
-     public int m(){
-          return k(2);
-     }
}
package p1;
public class B extends A {
     public int k(int i){
          return 42;
     }
     public int test(){
          return m();
     }
+     public int m(){
+          return k(2);
+     }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PDM_10**

```java
package p1;
public class A {
        public long m()  {
                return k();
        }
        public long k(){
                return 1;
        }
}
package p1;
public class B extends A {
        public long k(){
                return 2;
        }
        public long test(){
                return m();
        }
}
```

(a) Original version.

```java
package p1;
public class A {
-            public long m()  {
-                    return k();
-            }
        public long k(){
                return 1;
        }
}
package p1;
public class B extends A {
        public long k(){
                return 2;
        }
        public long test(){
                return m();
        }
+        public long m()  {
+                return super.k();
+        }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.test().

- **PDM_11**

```
package p1;
public class A {
     public long m() {
          return new A().k(2);
     }
     public long k(long a){
          return 2;
     }
     public long k(int a){
          return 1;
     }
}
package p2;
import p1.A;
public class B extends A {
     public long test(){
          return m();
     }
}
```

(a) Original version.

```
package p1;
public class A {
-          public long m() {
-               return new A().k(2);
-          }
     public long k(int a){
          return 1;
     }
     public long k(long a){
          return 2;
     }
}
package p2;
import p1.A;
public class B extends A {
     public long test(){
          return m();
     }
+     public long m() {
+          return new A().k(2);
+     }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in A.m().

- **PDM_12**

```java
package p1;
import p2.*;
public class ClassId1 extends ClassId0{
}
package p2;
public class ClassId0 {
    public long methodid1(){
        return 1;
    }
    public long m0(){
        return new ClassId1().methodid1();
    }
}
package p2;
public class ClassId1 extends ClassId0{
    public long methodid1(){
        return 0;
    }
    public long methodid0(){
        return m0();
    }
}
```

(a) Original version.

```java
package p1;
import p2.*;
public class ClassId1 extends ClassId0{
+    public long m0(){
+        return new ClassId1().methodid1();
+    }
}
package p2;
public class ClassId0 {
    public long methodid1(){
        return 1;
    }
-        public long m0(){
-            return new ClassId1().methodid1();
-        }
}
package p2;
public class ClassId1 extends ClassId0{
    public long methodid1(){
        return 0;
    }
    public long methodid0(){
        return m0();
    }
+    public long m0(){
+        return new ClassId1().methodid1();
+    }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in ClassId0.m0().

- **PDM_13**

```java
package p1;
import p2.A;
public class B extends A {
     protected long k(int a){
          return 0;
     }
     public long n() {
          return m();
     }
}
package p2;
public class A {
     long k(long a){
          return 1;
     }
     public long m(){
          return k(2);
     }
}
package p2;
import p2.A;
public class B extends A {
}
```

(a) Original version.

```java
package p1;
import p2.A;
public class B extends A {
     protected long k(int a){
          return 0;
     }
     public long n() {
          return m();
     }
+     public long m(){
+          return k(2);
+     }
}
package p2;
public class A{
     long k(long a){
          return 1;
     }
-     public long m(){
-          return k(2);
-     }
}
package p2;
import p2.A;
public class B extends A {
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in B.m().

- **PDM_14**

```
package p1;
import p2.A;
public class B extends A {
    public long k(long a){
        return 0;
    }
    public long n() {
        return m();
    }
}
package p2;
public class A {
    long k(int a){
        return 1;
    }
    public long m(){
        return new B().k(2);
    }
}
package p2;
public class B extends A {
}
```

(a) Original version.

```
package p1;
import p2.*;
public class B extends A {
    public long k(long a){
        return 0;
    }
    public long n() {
        return m();
    }
+    public long m(){
+        return new B().k(2);
+    }
}
package p2;
public class A {
    long k(int a){
        return 1;
    }
-        public long m(){
-            return new B().k(2);
-        }
}
package p2;
public class B extends A {
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in A.m().

- **PDM_15**

```java
package p1;
import p2.*;
public class B extends A {
}
package p2;
public class A {
    public long k(){
        return 1;
    }
    public long m(){
        return new B().k();
    }
}
package p2;
public class B extends A {
    public long k(){
        return 0;
    }
    public long n(){
        return m();
    }
}
```

(a) Original version.

```java
package p1;
import p2.*;
public class B extends A {
+     public long m(){
+         return new B().k();
+     }
}
package p2;
public class A {
    public long k(){
        return 1;
    }
-     public long m(){
-         return new B().k();
-     }
}
package p2;
public class B extends A {
    public long k(){
        return 0;
    }
    public long n(){
        return m();
    }
+     public long m(){
+         return new B().k();
+     }
}
```

(b) Target Version.

Error Type: Modified method calls due to incorrect overloading/overriding.

Error: Binding change in k().

3.5 Rename Method

- **RM_1**

```java
package p1;
public class A {
    public long k(long a){
        return 1;
    }
}
package p2;
import p1.A;
public class B extends A {
    protected long n( int a){
        return 0;
    }
    public long m(){
        return k(2);
    }
}
```

(a) Original version.

```diff
 package p1;
 public class A {
     public long k(long a){
         return 1;
     }
 }
 package p2;
 import p1.A;
 public class B extends A {
-     protected long n( int a){
-         return 0;
-     }
+     protected long k( int a){
+         return 0;
+     }
     public long m(){
         return k(2);
     }
 }
```

(b) Target Version.

Error Type: Missing updates in callers/Modified method calls due to incorrect overloading/overriding.

Error: Incomplete caller update/ Binding change

- **RM_2**

```java
package p1;
public class A {
     public long k(  long a){
          return 1;
     }
}
package p2;
import p1.*;
public class B extends A {
     protected long n(  int a){
          return 0;
     }
     public long m(){
          return this.k(2);
     }
}
```

(a) Original version.

```java
package p1;
public class A {
     public long k(  long a){
          return 1;
     }
}
package p2;
import p1.*;
public class B extends A {
-          protected long n(int a){
-               return 0;
-          }
+     protected long k(int a){
+          return 0;
+     }
     public long m(){
          return this.k(2);
     }
}
```

(b) Target Version.

Error Type: Missing updates in callers/Modified method calls due to incorrect overloading/overriding.

Error: Incomplete caller update / Binding change B.m()

- **RM_3**

```
package p1;
public class A {
  public long k(  int a){
    return 1;
  }
}
package p2;
import p1.*;
public class B extends A {
  public long m(){
    return new A().k(2);
  }
  public long n(  int a){
    return 0;
  }
}
```

(a) Original version.

```
  package p1;
  public class A {
    public long k(  int a){
      return 1;
    }
  }
  package p2;
  import p1.*;
  public class B extends A {
       public long m(){
           return new A().k(2);
         }
-              public long n(  int a){
-              return 0;
-              }
+          public long k(  int a){
+            return 0;
+          }
  }
```

(b) Target Version.

Error Type: Missing updates in callers/Modified method calls due to incorrect overloading/overriding.

Error: Incomplete caller update / Binding change B.m()

- **RM_4**

```java
package p1;
public class A {
  static String m(int i) {
    return "42";
  }
  public int test() {
    return
    Integer.parseInt(valueOf(23));
  }
}
```

(a) Original version.

```diff
package p1;
public class A {
-     static String m(int i) {
-     return "42";
- }
+  static String valueOf(int i) {
+    return "42";
+  }
  public int test() {
    return
    Integer.parseInt(valueOf(23));
  }
}
```

(b) Target Version.

Error Type: Missing updates in callers/Modified method calls due to incorrect overloading/overriding.

Error: Incomplete caller update / Binding change A.test()

Bibliography

[1] Gustavo Soares, Rohit Gheyi, and Tiago Massoni. Automated behavioral testing of refactoring engines. *Software Engineering, IEEE Transactions on*, 39(2):147–162, 2013.