Detecting **Overly Strong Preconditions** in Refactoring Engines

Melina Mongiovi

Federal University of Campina Grande

---

It’s difficult to implement refactorings!

```
program → program

preconditions
```

---

**Overly Strong Preconditions**

![Diagram of program transformations](image)

---

Pull up Method B.m

![Diagram showing pull up method](image)
We found >40 bugs related to overly strong preconditions in Eclipse Bugzilla.

“A high rate of false alarms may lead to mistrust of warnings. Thus, reducing the number of false alarms might mitigate the misuse of automated refactoring when they report warnings.”

“Programmers prefer flexible refactorings, often ignore precondition failures, and manually fix the problems afterwards.”

Vakilian et al. ICSE 2012
Detecting overly strong preconditions in refactoring engines by disabling some preconditions.

How do we generate test inputs?

Additional constraints to guide the program generation

Well-formed Rules

```alloy
one sig C1, C2 extends Class {}
one sig M extends Method {}
pred generate() {
M in C1.methods && C2 in C1."extend
}
```

```alloy
fact ClassCannotExtendItself {
all c: Class | c ! in c."extend
}
```
How do we disable preconditions?

```java
public void unlockOverriding() {
    if (!old_overridden.equals(overriddenMethods()))
        throw new RefactoringException("overriding has changed");
}
```

```java
ConditionsMoveMethod.cond1.enabled = false;
```

Move Method

```java
public void unlockOverriding() {
    if (!old_overridden.equals(overriddenMethods()))
        throw new RefactoringException("overriding has changed");
}
```

DP changes

```java
public void unlockOverriding() {
    if (!old_overridden.equals(overriddenMethods()))
        throw new RefactoringException("overriding has changed");
}
```
DP changes

How do we evaluate transformation correctness?

AO implementation

AP implementation
AO implementation JRRT

```java
public aspect DPreconditions { JRRT extends DisablePreconditions {
    pointcut methodMsg(String msg):
    call (void AST.RefactoringException.throwException(String...)) &&
    args(msg);
    if (!old_overridden.equals(overriddenMethods()))
        throw new RefactoringException("overriding has changed");
}
```

**Overly Strong Preconditions**

<table>
<thead>
<tr>
<th>Refactoring</th>
<th>Evaluated Preconditions</th>
<th>Overly Strong Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JRRT</td>
<td>Eclipse</td>
</tr>
<tr>
<td>Move Method</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Push Down Field</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Rename Method</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Push Down Method</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Pull Up Field</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Add Parameter</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Encapsulate Field</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Rename Field</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Rename Type</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>24</td>
<td>25</td>
</tr>
</tbody>
</table>

**Disabling Precondition (DP) X Differential Testing (DT)**

<table>
<thead>
<tr>
<th>Refactoring</th>
<th>Eclipse</th>
<th>JRRT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DP</td>
<td>DT</td>
</tr>
<tr>
<td>Move Method</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Pull Up Method</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Push Down Field</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Rename Method</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Push Down Method</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Pull Up Field</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Add Parameter</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Encapsulate Field</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Rename Field</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Rename Type</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>15</td>
<td>18</td>
</tr>
</tbody>
</table>

**Only DP technique detects**

- Push Down Field A to C

**Only DT technique detects**

- Rename Field C.f2 to f1

Eclipse does not add the "cast" when we disable the condition.
Detecting Overly Strong Preconditions in Refactoring Engines

Melina Mongiovi

M. Mongiovi, R. Gheyi, G. Soares, M. Ribeiro, P. Borba, and L. Teixeira,
"Detecting overly strong preconditions in refactoring engines," IEEE Transactions on Software Engineering, 2017