Dynamic Patch Generation for Null Pointer Exceptions Using Metaprogramming

Thomas Durieux, Benoit Cornu, Lionel Seinturier and Martin Monperrus
February 24, 2017

Inria & University of Lille
Table of contents

1. Motivation

2. Test Based Automatic Repair Approach

3. NPEfix

4. Evaluation
Motivation
Motivation

Why fixing Null Pointer Exception?

Top 10 Exception Types by Frequency in 1,000+ Applications
Test Based Automatic Repair Approach
Test Based Automatic Repair Approach

A buggy program
A test suite
A repair strategy
Automatic Program Repair: Patch Generation and Validation

1. Fault Localization
Automatic Program Repair: Patch Generation and Validation

1. Fault Localization → 2. Generate Patch
Automatic Program Repair: Patch Generation and Validation

1. Fault Localization → 2. Generate Patch → 3. Compile
Automatic Program Repair: Patch Generation and Validation

1. Fault Localization ➔ 2. Generate Patch ➔ 3. Compile

4. Execute tests
Automatic Program Repair: Patch Generation and Validation


Is Valid?
Automatic Program Repair: Patch Generation and Validation

1. Fault Localization
2. Generate Patch
3. Compile

4. Execute tests

Is Valid?

No

Yes

5. Patch
NPEfix
NPEfix

A buggy program  A test suite  A repair strategy
Replace the null expression

```java
+ if (r == null) {
+   anotherVar.foo(p);
+ } else {
+   r.foo(p);
+ }
```

```java
+ if (r == null) {
+   new Foo().foo(p);
+ } else {
+   r.foo(p);
+ }
```
NPEfix: Repair Strategies

Skip the null expression

```java
+ if (r == null) {
+     return anotherVar;
+ }
    r.foo(p);
```

```java
+ if (r == null) {
+     return new Bar();
+ }
    r.foo(p);
```
Object m(A p) {
    field.inv();
    return p;
}
Object m(A p) {
    try {
        call(field).inv()
        return p;
    } catch (SkipMethod e) {
        if (returnVar) return getVar()
        if (returnNew)
            return createObject();
    }
}
1. Code transformation
NPEfix

1. Code transformation → 2. Compile Metaprogram
NPEfix


4. Execute tests → 5. Runtime Patch

Is Valid? Yes → No
NPEfix


Is Valid?

4. Execute tests → 5. Runtime Patch
NPEfix


Is Valid? → No

4. Execute tests → 5. Runtime Patch → Yes

5. Patch
--- pdfbox/pdmodel/interactive/form/
PDAcroForm.java
+++ pdfbox/pdmodel/interactive/form/
PDAcroForm.java
@@ -250,2 +250,5 @@

+     if (fields == null) {
+         return retval; // retval is null
+     }

for (int i = 0; i < fields.size() &&
     retval == null; i++)
Evaluation
Research Question

What is the impact of the meta programming approach compare to a template based approach on the number of patches?
Evaluation Protocol

Compare metaprogramming to template based repair approach

16 real null dereference bugs

Collect the number of generated patches
The number of generated patches

![Bar chart showing the number of generated patches for different projects. The x-axis represents various projects such as Collections-360, Felix-4960, Lang-304, Lang-587, Lang-703, Lang-1115, Math-1117, Math-290, Math-305, Math-369, Math-988a, Math-988b, PDFBox-2812, PDFBox-2965, PDFBox-2995, Sling-4982. The y-axis represents the number of patches ranging from 0 to 80. The chart compares two categories: Template and NPEFix.]
Conclusion

**Take way**
It is possible to explore the embedded search space at runtime

**Future work**
Uses the metaprogramming approach for multi-points patches

https://github.com/Spirals-Team/npefix
Why do we use complex techniques when simple techniques work almost as well?