

Exhaustive Exploration of the Failure-oblivious Computing Search Space

Thomas Durieux, Youssef Hamadi, Zhongxing Yu, Benoit Baudry and Martin Monperrus

March 12, 2018

Inria & University of Lille

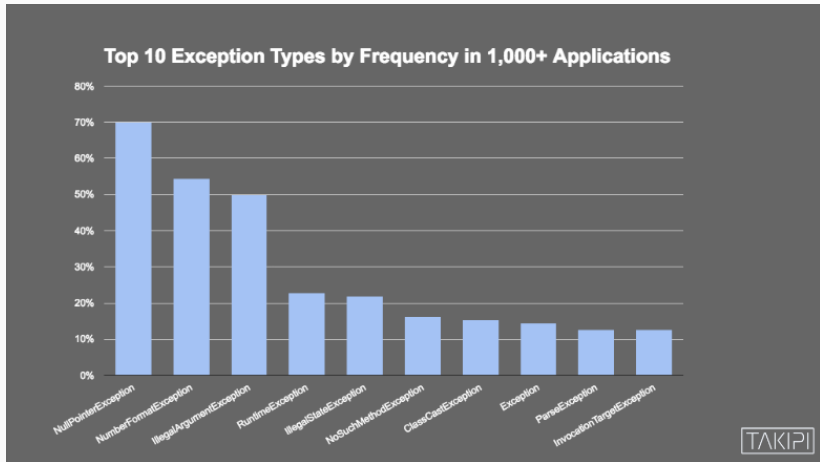
Table of contents

1. Motivation
2. Test Based Automatic Repair Approach
3. NPEfix
4. Evaluation

Motivation

Motivation

Why fixing Null Pointer Exception?



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

```
graph LR; A[1. Fault Localization] --> B[2. Generate Patch]
```


Automatic Program Repair: Patch Generation and Validation

1. Fault Localization

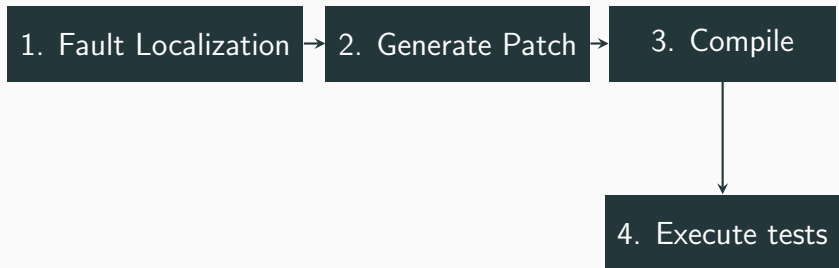


2. Generate Patch

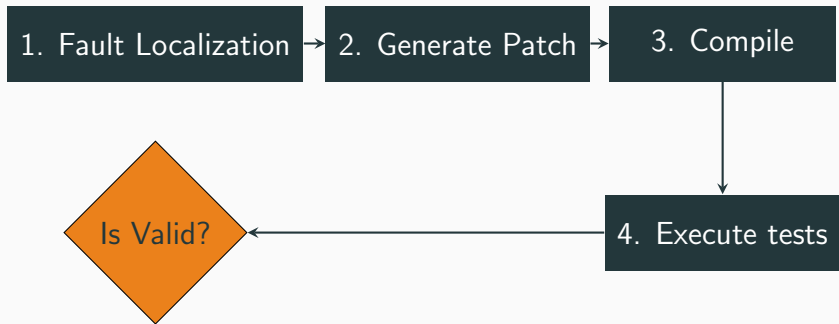


3. Compile

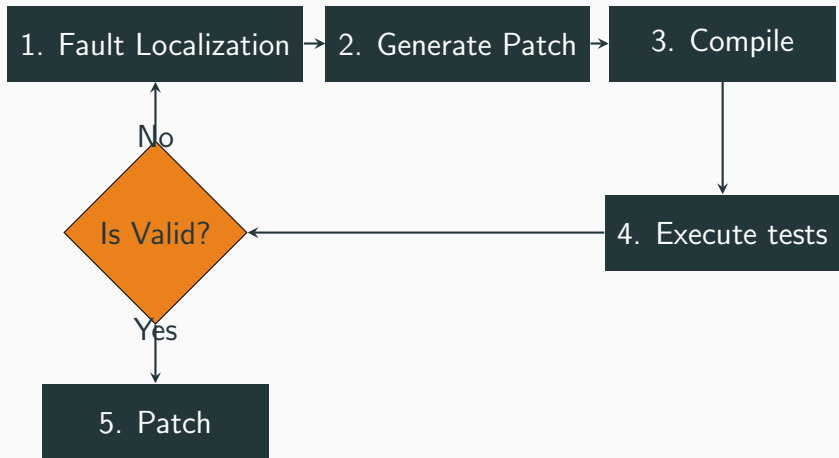
Automatic Program Repair: Patch Generation and Validation



Automatic Program Repair: Patch Generation and Validation



Automatic Program Repair: Patch Generation and Validation



NPEfix

NPEfix



A buggy program



A test suite



A repair strategy

NPEfix: Repair Strategies

Replace the null expression

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

```
+ if (r == null) {  
+   new Foo().foo(p);  
+ } else {  
+   r.foo(p);  
+ }
```

NPEfix: Repair Strategies

Skip the null expression

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

```
+ if (r == null) {  
+   return new Bar();  
+ }  
  r.foo(p);
```


NPEfix Code Transformation

```
Object m(A p) {  
    field.inv();  
    return p;  
}
```

NPEfix Code Transformation

```
Object m(A p) {  
    try {  
        call(field).inv()  
        return p;  
    } catch (SkipMethod e){  
        if (returnVar) return getVar()  
        if (returnNew)  
            return createObject()  
    }  
}
```

1. Code transformation

1. Code transformation → 2. Compile Metaprogram

1. Code transformation

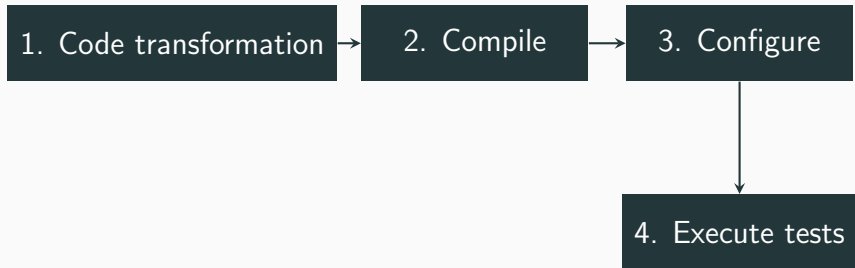


2. Compile

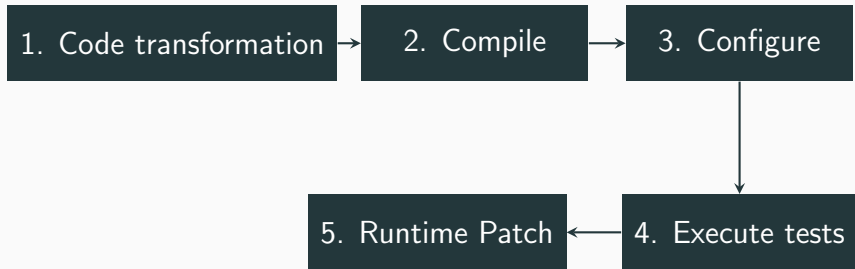


3. Configure

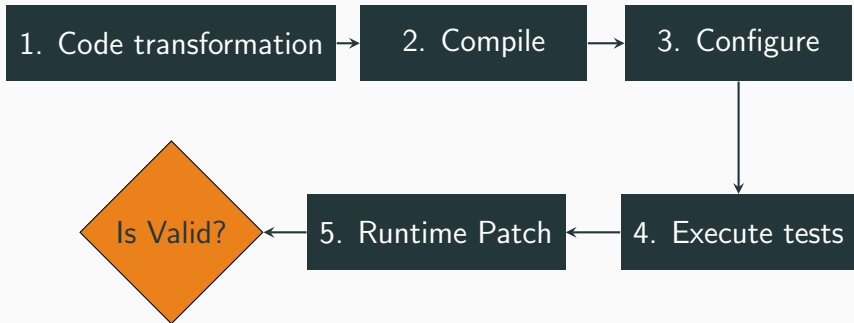
NPEfix



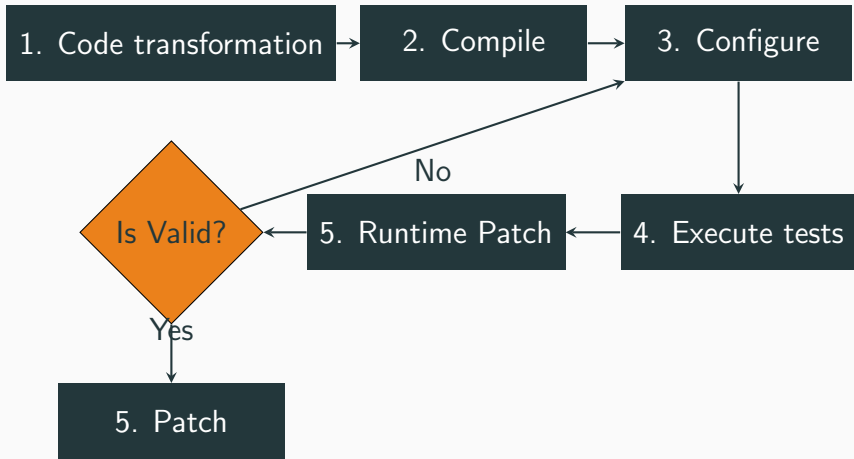
NPEfix



NPEfix



NPEfix



Patch Example

```
--- pdfbox/pdmodel/interactive/form/  
    PDAcroForm.java  
+++ pdfbox/pdmodel/interactive/form/  
    PDAcroForm.java  
@@ -250,2 +250,5 @@  
  
+   if (fields == null) {  
+       return retval; // reval 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?

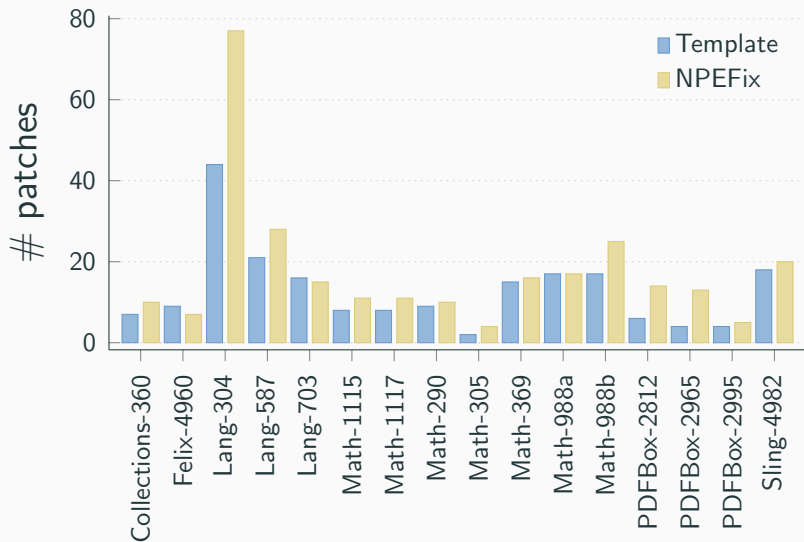
Evaluation Protocol

Compare metaprogramming to template based repair approach

16 real null dereference bugs

Collect the number of generated patches

The number of generated patches



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`