

# Abschlusspräsentation

Birgit Pohl, Philipp Badenhoop

Tim Sikatzki, Daniel Bucher

Software Engineering II

Humboldt-Universität zu Berlin, Institut für Informatik



Verantwortlicher: S. Heiden

Idee

Astor - GenProg

Problemstellung

Ergebnisse

Idee

## Erster Ansatz

- ▶ Suche nach einem Opensourceprogramm
- ▶ Über [www.codetriage.com](http://www.codetriage.com)
  - ▷ Zur Vermittlung von Opensourceprojekten an Entwickler
- ▶ Auswahl von Astor als Debuggingtool

## libgdx

- ▶ Ist ein Java-Framework
- ▶ Für plattformunabhängige Spielentwicklung
- ▶ Unterstützt Windows, Mac, Linux, Android, iOS & Blackberry
- ▶ Unter Apache-2-Lizenz freigegeben

## Astor

- ▶ Automatic Software Transformations fOr program Repair
- ▶ Für automatische Reparatur von Java Programmen
- ▶ Ursprüngliche implementaiton in C, jetzt in Java
- ▶ Besteht aus 3 Programmteilen
  - ▷ jGenProg2
  - ▷ jKali
  - ▷ jMutRepair

## Probleme

- ▶ Installationsprobleme von libgdx
  - ▷ Benötigt spezielle Bibliotheken
- ▶ Probleme bei Astor
  - ▷ Kann Fehler nicht fixen, welche GenProg können soll
- ▶ Neuer Fokus auf Astor

# Astor - GenProg



## Astor

- ▶ Wendet eins der drei Modi an
  - ▷ jGenProg2
  - ▷ jKali
  - ▷ jMutRepair
- ▶ Unser Fokus wurde auf GenProg gelegt

# Kali

- ▶ Zielt auf schwache Testsuits
- ▶ Vorgehen bei der Reperatur":
  - ▷ löschen von Zeilen
  - ▷ überspringen von Zeilen

## MutRepair

- ▶ Mutiert die Konditionen von if-Statements
- ▶ Hat drei Arten der Mutation:
  - ▷ Relations Operationen
  - ▷ Logische Operationen
  - ▷ Negation

# GenProg

- ▶ Idee: Reparatur durch Evolution
  - ▷ Nutzung von generischer Programmierung
  - ▷ gezielte, zufällige Mutation

## Vorgehen

- ▶ Fehlerlokalisierung
  - ▷ Erstellung eines abstrakten Syntax Baums
  - ▷ Durchlaufen der Testfälle
  - ▷ Fehlerbestimmung anhand von Pfaden mit negativen Testfällen
- ▶ Patch-Generierung
  - ▷ Mutation von Crossovervarianten
  - ▷ Solange, bis ein optimaler Kandidat gefunden wurde
- ▶ Validierung
  - ▷ Prüfen des Testsuits

# Problemstellung

## Testprogramm (1)

```
1  public class Program {  
2      public Language getLanguage(String lang) {  
3          if (lang.equals("C"))  
4              return Language.C;  
5          else if (lang.equals("CPP"))  
6              return Language.CPP;  
7          else  
8              return Language.JAVA;  
9      }  
10  
11     public Language working(String lang) {  
12         if (lang.equals("C"))  
13             return Language.C;  
14         else if (lang.equals("CPP"))  
15             return Language.CPP;  
16         else if (lang.equals("JAVA"))  
17             return Language.JAVA;  
18         else  
19             return Language.PYTHON;  
20     }  
21 }
```

## Testprogramm (2)

```
1 public enum Language {  
2     C, CPP, JAVA, PYTHON  
3 }
```

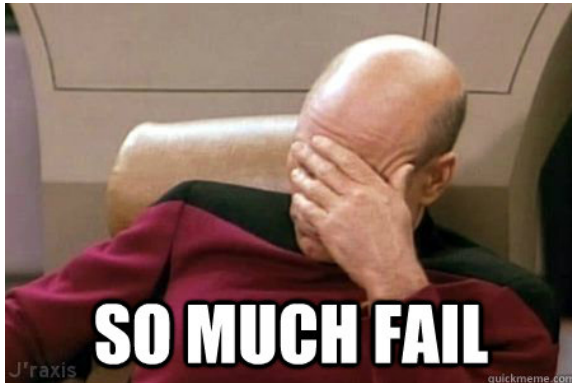


## Test-Suite

```
1  @Test
2  public void p1() {
3      assertEquals(Language.C, new Program().getLanguage("C"));
4  }
5
6  @Test
7  public void p2() {
8      assertEquals(Language.CPP, new Program().getLanguage("CPP"));
9  }
10
11 @Test
12 public void p3() {
13     assertEquals(Language.JAVA, new Program().getLanguage("JAVA"));
14 }
15
16 @Test
17 public void n1() {
18     assertEquals(Language.PYTHON, new Program().getLanguage("PYTHON"));
19 }
```

## Resultat

Astor findet keinen Bugfix...



- ▶ Haben wir etwas falsch gemacht?
- ▶ Was müsste GenProg tun, um das Programm zu reparieren?

## Analyse (Original)

```
1  public Language getLanguage(String lang) {  
2      if (lang.equals("C")) {  
3          return Language.C;  
4      } else {  
5          if (lang.equals("CPP")) {  
6              return Language.CPP;  
7          } else {  
8              return Language.JAVA;  
9          }  
10     }  
11 }
```

## Analyse (Schritt 1)

```
1  public Language getLanguage(String lang) {
2      if (lang.equals("C")) {
3          return Language.C;
4      } else {
5          if (lang.equals("CPP")) {
6              return Language.CPP;
7          } else {
8              // REMOVE STATEMENT: "return Language.JAVA;"
9          }
10     }
11 }
```

## Analyse (Schritt 2 - Fix)

```
1  public Language getLanguage(String lang) {
2      if(lang.equals("C")) {
3          return Language.C;
4      } else {
5          if(lang.equals("CPP")) {
6              return Language.CPP;
7          } else {
8              // INSERT STATEMENT:
9              if(lang.equals("JAVA")) {
10                 return Language.JAVA;
11             } else {
12                 return Language.PYTHON;
13             }
14         }
15     }
16 }
```

- ▶ Nach einer intensiven Debug-Session finden wir heraus...
- ▶ ... das Problem liegt in der Methode  
**VariableSolver.fitInContext()**

## Problem

---

**Algorithm 1:** `VariableResolver.fitInContext()`

---

**Input** : Set of context variables *varContext*.**Input** : Statement *stmt*.

```
1 varAccesses = collectVariableAccesses(stmt);  
2 forall access  $\in$  varAccesses do  
3   |   contextIndependent = stmt.contains(access.getDeclaration())  
      |    $\vee$  access.isPublicAndStatic();  
4   |   if  $\neg$ contextIndependent  $\wedge$   $\neg$ varContext.contains(access) then  
5   |   |   return false;  
6   |   end  
7 end  
8 return true;
```

---



## Problem

Zu kontextunabhängigen Variablen  
gehören auch enum-Referenzen!

## Fix

---

**Algorithm 2:** VariableResolver.fitInContext()

---

**Input** : Set of context variables *varContext*.**Input** : Statement *stmt*.

```
1 varAccesses = collectVariableAccesses(stmt);
2 forall access ∈ varAccesses do
3   | contextIndependent = stmt.contains(access.getDeclaration())
3   |   ∨ access.isPublicAndStatic() ∨
3   |   access.isPublicEnumReference() ;
4   | if ¬contextIndependent ∧ ¬varContext.contains(access) then
5   |   | return false;
6   | end
7 end
8 return true;
```

---

# Resultat nach Fix

```
1 public class Program {
2     public com.astortest.Language getLanguage(java.lang.String lang) {
3         if (lang.equals("C"))
4             return com.astortest.Language.C;
5         else
6             if (lang.equals("CPP"))
7                 return com.astortest.Language.CPP;
8             else
9                 if (lang.equals("JAVA"))
10                     return com.astortest.Language.JAVA;
11                 else
12                     return com.astortest.Language.PYTHON;
13
14
15     }
16
17
18     public static com.astortest.Language working(java.lang.String lang) {
19         if (lang.equals("C"))
20             return com.astortest.Language.C;
21         else
22             if (lang.equals("CPP"))
23                 return com.astortest.Language.CPP;
24             else
25                 if (lang.equals("JAVA"))
26                     return com.astortest.Language.JAVA;
27                 else
28                     return com.astortest.Language.PYTHON;
29
30
31     }
32 }
33 }
```

# Ergebnisse

## Die Testmenge

- ▶ Als Testmenge wurden die Fehler von defects4j verwendet.
- ▶ 395 Test stehen zur Verfügung in folgenden Bereichen
  - ▷ JFreechart (26)
  - ▷ Closure compiler (133)
  - ▷ Apache commons-lang (65)
  - ▷ Apache commons-math (106)
  - ▷ Mockito (38)
  - ▷ Joda-Time (27)

Project	jGenProg	jKali	jMutRepair
Chart	C1, C3, C5, C7, C13, C15, C25,	C1, C5, C13, C15, C25, C26	C1, C7, C25, C26
	$\sum = 7$	$\sum = 6$	$\sum = 4$
Lang	0	0	L27
	$\sum = 0$	$\sum = 0$	$\sum = 1$
Math	M2, M5, M7, M8, M28, M40, M49, M50, M53, M60, M70, M71, M73, M78, M80, M81, M82, M84, M85, M95	M2, M8, M28, M32, M40, M49, M50, M78, M80, M81, M82, M84, M85, M95	M2, 28, 40, 50, 57, 58, 81, 82, 84, 85, 88
	$\sum = 20$	$\sum = 14$	$\sum = 11$
Time	T4, T11	T4, T11	T11
	$\sum = 2$	$\sum = 2$	$\sum = 1$
Total	29	22	17

Entnommen aus [MM16]

## Erste Überlegung

- ▶ Anwendung auf das Paket Lang
- ▶ Höchste Wahrscheinlichkeit einen Fix zu finden
- ▶ Leider nicht erfolgreich gewesen

## Nächste Überlegung

- ▶ Durchsuchen aller Fehlerhaften Programme
- ▶ Überprüfung der fehlerhaften Testfälle
- ▶ Kein Testfall gefunden, welcher den Patch validieren würde



## Validierung

- ▶ Entwicklung eines eigenen Testprogramms
- ▶ Welches den Patch validieren kann

```
1 public Language getLanguage(String lang) {  
2     if (lang.equals("C"))  
3         return Language.C;  
4     else if (lang.equals("CPP"))  
5         return Language.CPP;  
6     else  
7         return Language.JAVA;  
8 }
```

```
1 public Language getLanguageWorking(String lang) {  
2     if (lang.equals("C"))  
3         return Language.C;  
4     else if (lang.equals("CPP"))  
5         return Language.CPP;  
6     else if (lang.equals("JAVA"))  
7         return Language.JAVA;  
8     else  
9         return Language.PYTHON;  
10 }
```

## Von Astor generierter Patch

```
1    public Language getLanguage(String lang) {  
2        if (lang.equals("C"))  
3            return Language.C;  
4        else if (lang.equals("CPP"))  
5            return Language.CPP;  
6        else if (lang.equals("JAVA"))  
7            return Language.JAVA;  
8        else  
9            return Language.PYTHON;  
10    }
```

## Quellen



R. Just, D. Jalali, and M. D. Ernst.

Defects4j: A database of existing faults to enable controlled testing studies for java programs, 2014.



M. Martines and M. Monperrus.

ASTOR: A Program Repair Library for Java.  
2016.