Precise Program Repair

Yingfei Xiong

Peking University

Background of Yingfei Xiong

- 2000~2004, UESTC, Undergraduate
- 2004~2006, Peking University, Graduate
 - Adivsor: Hong Mei, Fuqing Yang
- 2006~2009, The University of Tokyo, Ph.D.
 - Advisor: Zhenjiang Hu, Masato Takeichi
- 2009~2011, University of Waterloo, PostDoc
 - Supervisor : Krzysztof Czarnecki
- 2012~, Peking University, Assistant Professor under Young Talents Plan
- Research Interests
 - Software Analysis, Program Language Design

Origin

- "War. War never changes" Fallout series
 - The war between developers and bugs never changes
- Fault Detection: Is there a bug?
 - Since 60s
 - Example Techniques : Testing, Verification
- Fault Localization: Where is the bug?
 - Since 90s
 - Example Techniques : Spectrum-based fault localization
- Fault Repair: How to fix the program?
 - Since 00s
 - Example Techniques : Test-based Program Repair

"Generate-Validate" Program Repair

Input: a program and a set of tests, where the program fails at least one test

Output: a patch that makes the program pass all the tests



Existing Work

- GenProg
 - [Weimer et al.: ICSE'09, GECCO'09, CACM'10, ICSE'12]
 - Approach
 - Replace the potentially faulty code with code pieces elsewhere
 - Use search algorithm to find an optimal combination
 - Results: 55/105, 8\$/bug
- Inspired a wave of program repair research
 - AutoFix, Nopol, RSRepair, MintHint, AutoRepair, SemFix, DirectFix, SPR...

A Turning Point

- [Qi-ISSTA'15]
 - Only 2 among the 55 defects were correctly fixed by GenProg
 - Reason: passing the test does not guarantee correctness
- [Le Goues-FSE'15]
 - Extensive experiments on more methods, datasets, test suites
 - The finding still holds
- Other work
 - Prophet, Angelix
 - The precision (proportion of correct patches) is lower than 40%

Our Work

High Precision Defect Repair

Learn from QA sites [ASE15]



Precise Condition Repair [ICSE17]



[ASE15] Qing Gao, Hansheng Zhang, Jie Wang, Yingfei Xiong, Lu Zhang, Hong Mei. Fixing Recurring Crash Bugs via Analyzing Q&A Sites. ASE'15

[ICSE17] Yingfei Xiong, Jie Wang, Runfa Yan, Jiachen Zhang, Shi Han, Gang Huang, Lu Zhang. Precise Condition Synthesis for Program Repair. ICSE'17

Fixing from QA Sites

- How do developers get their experience?
- 29 public void onReceive (final Context context, final Intent intent) {
- 30 final int action = intent.getExtras().getInt(KEY_ACTION, -1);
- 31 final float bl = BatteryHelper.level(context);
- 32 LOG.i("AlarmReceiver invoked: action=%s bl=%s.", action, bl);
- 33 switch (action) {

... 51 }

52 }

java.lang.RuntimeException: Unable to start receiver com.vaguehope.onosendai.update.AlarmReceiver:

Fixing from QA Sites

Search tools

java.lang.RuntimeException: Unable to start receiver : android.conten

Web Videos News Images More -

8 results (0.52 seconds)

android - "IntentReceiver components are not allowed to ... stackoverflow.com/.../intentreceiver-components-are-not-allowed-to-regi... Jul 24, 2014 - "IntentReceiver components are not allowed to register to receive ... ACTION_BATTERY_CHANGED); Intent batteryStatus = c. ... RuntimeException: Unable to start receiver ... ActivityThread.main(ActivityThread.java:4627) at java. Iang.reflect. ... NativeStart.main(Native Method) Caused by: android.content.

android - Battery changed broadcast receiver crashing app ...

stackoverflow.com/.../battery-changed-broadcast-**receiver**-crashing-app-... ▼ Feb 27, 2013 - Battery changed broadcast receiver crashing app on some phones. No ... PowerConnectionReceiver"> <intent-filter> <action android:name="android.intent .action. ... RuntimeException: Unable to start receiver com.doublep.wakey. ReceiverCallNotAllowedException: IntentReceiver components are not ...

android - Want app to execute some code when phone is ... stackoverflow.com/.../want-app-to-execute-some-code-when-phone-is-pl... Jun 29, 2012 - ACTION_BATTERY_CHANGED)); int plugged = intent. ... The code errors out with: *FATAL EXCEPTION: main:: java.lang.RuntimeException: Unable to start receiver com.example.ChargingOnReceiver: android.content. ... IntentReceiver Stack Overflow is a community of 4.7 million programmers, just like you, helping each othe only takes a minute:

"IntentReceiver components are not allowed to register to receive inter determine Battery level



 \star

2

stack overflow

Test your app **on real Android devices** in the cloud. Keynote MOBILE TESTING PRO **START YOUR FREE TRIAL**

 I am trying to get Battery info from my Application following the guidelines at http://developer.android.com/training/monitoring-device-state/batterymonitoring.html

This is the method is came up with to check the battery level:

public void sendBatteryInfoMessage(){

IntentFilter iFilter = new IntentFilter(Intent.ACTION_BATTERY_ Intent batteryStatus = c.registerReceiver(null, iFilter);

Challenge of Analyzing QA Sites

- It is hard to understand natural languages
- Instead of:
- 4 context.registerReceiver(null, new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
 - use:

```
context.getApplicationContext().registerReceiver(null, new IntentFilter(Intent.ACTION_BATTER
```

This is annoying -- registerReceiver() should be smarter than this -- but it's the workaround for this particular case.

- Observation: programmers communicates in programming languages
- Solution: Directly compare the code pieces

Approach Overview



Experiments

- 24 Android crash bugs that have answers on StackOverflow
 - Selected out of 161 Android crash bugs

- Correctly Fixed : 8
- Wrongly Fixed : 2
- Precision : 80%
- Recall : 33% (5% among Android crash bugs)

Precise Condition Synthesis

Targeted defect class: condition bugs

lcm = Math.abs(a+b);

+ if (lcm == Integer.MIN_Value)

+ throw new ArithmeticException();

Missing boundary checks

- if (hours <= 24) + if (hours < 24)

withinOneDay=true;

Conditions too weak or too strong

Condition bugs are common

ACS System

- ACS = Accurate Condition Synthesis
- Two sets of templates for repair



Challenge – Many incorrect conditions pass the tests

```
int lcm=Math.abs(
    mulAndCheck(a/gdc(a,b),b));
+if (lcm == Integer.MIN_VALUE) {
    throw new ArithmeticException();
+}
    return lcm;
```

Test 1 (Passed): Input: a = 1, b = 50 Oracle: Icm = 50

Correct condition: lcm == Integer.MIN_VALUE Test 2 (Failed): Input: a = Integer.MIN_VALUE, b = 1 Oracle: Expected(ArithmeticException)

Incorrect conditions:

• a != 1

- b == 1
- lcm != 50

```
• ...
```

Idea: Rank the Conditions

- Rank potential conditions by their probabilities of being correct
- Validate the conditions one by one
- Stop validating when the probability is too low



Idea: Rank the Conditions

- Rank potential conditions by their probabilities of being correct
- Validate the conditions one by one
- Stop validating when the probability is too low



Ranking Conditions is Difficult

- The number of potential conditions is large
 - Cannot enumerate the conditions
 - Difficult to perform statistics: not enough samples for each condition

Solution: Divide-and-Conquer



Step 1: Rank variables Step 2: Rank predicates for each variable

Ranking Method 1: Rank Variables by Data-Dependency

- Locality of variable uses: recently assigned variables are more likely to be used
- Rank variables by data-dependency
 - lcm = Math.abs(mulAndCheck(a/gdc(a, b), b))



• Consider only variables in the first two levels

Ranking Method 2: Filter Variables by JavaDoc

/** ...

* @throws IllegalArgumentException if initial is not between * min and max (even if it is a root) **/

Only variable "initial" is considered when throwing IllegalArgumentException

Ranking Method 3: Rank Predicates by Context

• The predicates tested on the variables are related to its context

```
Variable TypeVector v = ...;<br/>if (v == null) return 0;Variable Nameint hours = ...;<br/>if (hours < 24)<br/>withinOneDay=true;Method Nameint factorial() {<br/>...<br/>if (n < 21) {<br/>...
```

- Approximate the conditional probabilities by querying GitHub
- Consider only the predicates whose probabilities are larger than a threshold

Evaluation: Performance of ACS

Dataset: Four projects from Defects4J benchmark:

- Time, Lang, Math, Chart
- In total 224 defects

Approach	Correct	Incorrect	Precision	Recall
ACS	18	5	78.3%	8.0%
jGenProg	5	22	18.5%	2.2%
Nopol	5	30	14.3%	2.2%
xPAR	3	_4	_4	$1.3\%^{2}$
$HistoricalFix^1$	$10(16)^3$	_4	_4	$4.5\%(7.1\%)^{2,3}$

Vision

- Long-term Goal: automate programming
- Roadmap: deal with more and more difficult issues
 - Issue = bug report + feature requests

Conclusion

- Will program repair be useful in practice?
 - Increasing precision is the key
- Can we improve precision?
 - Yes, at least for incorrect conditions and crashes
- How can we improve precision?
 - By learning from existing resources