Fixing Recurring Crash Bugs via Analyzing Q&A Sites

Author: Qing Gao, Hansheng Zhang, Jie Wang, Yingfei Xiong, Lu Zhang, Hong Mei
Presenter: Nhat Minh Le – 40323412 – Concordia University

Agenda

- I. Overview
- II. Related work
- III. Main novelty
- IV. Technical contribution
- V. Experiment
- VI. Evaluation

I. Overview

Recurring bugs: bugs that occur often in different projects, and are found common

These bugs can affect various aspects of software and have a range of implications across different industries.

Some type of recurring bugs:
+ Configuration Errors
+ Input Validation and Parsing Errors
+ UI/UX Issues

II. Related work

1. GenProg: copies code pieces from other parts of the software project to fix the current bug

2. PAR: uses human-written templates to generate patches

3. RSRepair: assume that patches exist in the current project, and use search-based techniques to find the patches

4. SPR: instantiates transformation schemas to repair program defects by using condition synthesis

III. Main novelty

1. First new approach: fixing recurring crash bugs via analyzing Q&A sites (internet resource).

2. This new approach in this paper focuses only on one type of bug: crash bug

3. This new approach has the potential for further research and applying in real-worlds

IV. Technical contribution

1. Extracting queries from crash traces
2. Retrieving a list of Q&A pages
3. Analyze the pages
4. Generate edit scripts
5. Apply these scripts to target source code
6. Filter out the incorrect patches

1) Page extraction

2) Edit script extraction

3) Patch generation

4) Patch filtering

IV. Technical contribution

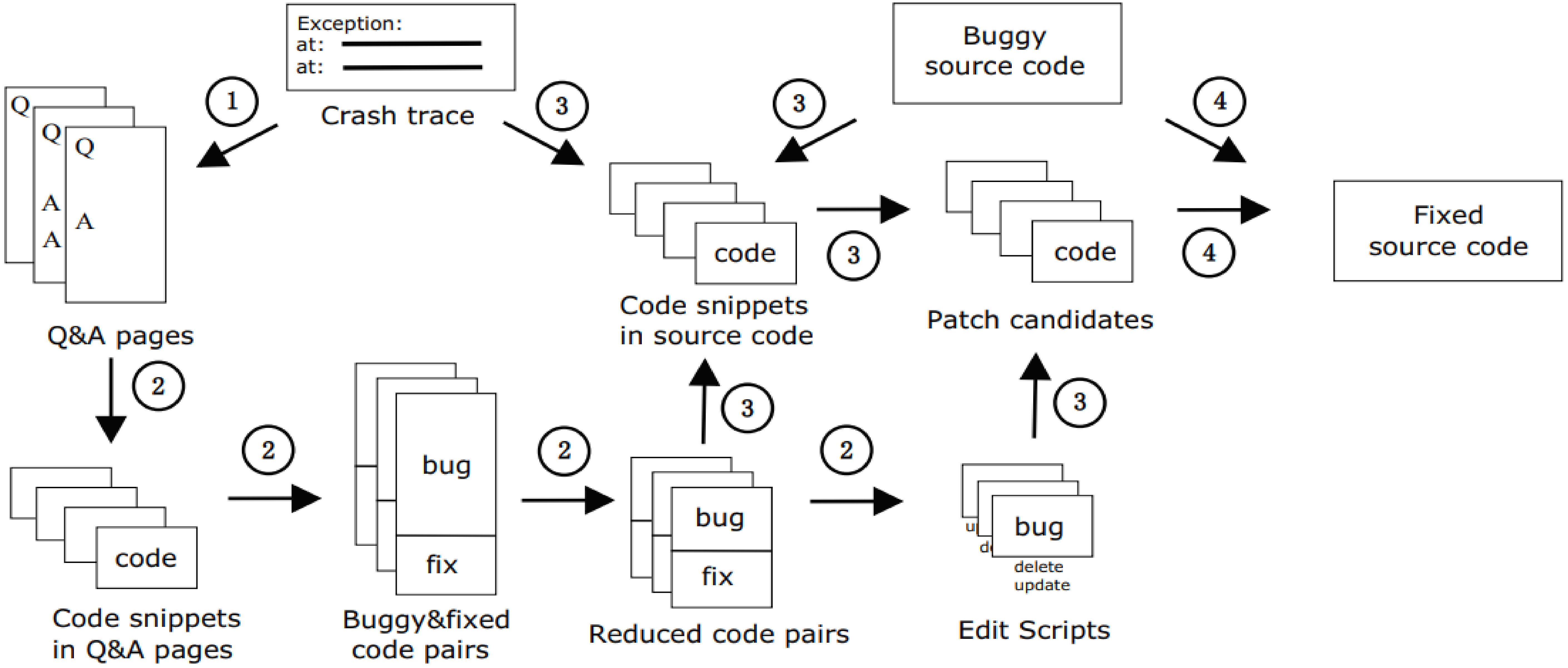

Fig. 1: Overview of our approach

IV.1 Q&A Page Extraction

```
java.lang.RuntimeException: Unable to start receiver com.vaguehope.onosendai.update.AlarmReceiver:
android.content.ReceiverCallNotAllowedException: IntentReceiver components are not allowed to register to receive intents
at android.app.ActivityThread.handleReceiver(ActivityThread.java:2126)
at android.app.ActivityThread.access$1500(ActivityThread.java:123)
at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1197)
at android.os.Handler.dispatchMessage(Handler.java:99)
at android.os.Looper.loop(Looper.java:137)
at android.app.ActivityThread.main(ActivityThread.java:4424)
at java.lang.reflect.Method.invokeNative(Native Method)
at java.lang.reflect.Method.invoke(Method.java:511)
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:784)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:551)
at dalvik.system.NativeStart.main(Native Method)
Caused by: android.content.ReceiverCallNotAllowedException: IntentReceiver components are not allowed to register to receive intents
at android.app.ReceiverRestrictedContext.registerReceiver(ContextImpl.java:118)
at android.app.ReceiverRestrictedContext.registerReceiver(ContextImpl.java:112)
at com.vaguehope.onosendai.update.AlarmReceiver.onReceive(AlarmReceiver.java:31)
at android.app.ActivityThread.handleReceiver(ActivityThread.java:2119)
... 10 more
```

Fig. 2: An example of a crash trace

IV.1 Q&A Page Extraction

The query: *java.lang.RuntimeException: Unable to start receiver IntentReceiver components are not allowed to register to receive intents*

IV.2 Edit Script Extraction

Consist 3 steps:

1. Buggy & fixed code pair extraction

2. Buggy & fixed code reduction

3. Edit script generation

IV.2.1 Buggy & Fixed code pair extraction

- Isolate code snippet from Q&A page by getting the content inside HTML tag pair <code> and </code>
- Combine different code snippet to buggy & fixed code pairs

IV.2.1 Buggy & Fixed code pair extraction

Buggy & fixed code pair:

1) Both buggy code and fixed code are in the same answer post.

2) Buggy code is in the question post, and fixed code is in the answer post

Question : How to know which is question, which is answer posts ?

1) Use keyword matching to distinguish the buggy code and the fixed code (instead of, change to...)

2) Take each code snippet in each type post

→ Rank 1) is higher than 2)

IV.2.2 Buggy & Fixed code reduction

Reduce the size of both buggy code and fixed code according to their similarities

1. Parse a buggy & fixed code pair and get two Abstract Syntax Trees (ASTs): BuggyAST and FixedAST

AST:

- Each node has a label – type of the node (eg: method invocation)
- Each leaf node has a value (eg: the name of a variable)

IV.2.2 Buggy & Fixed code reduction

2. Use partial parsing techniques to parse the code snippets into AST

3. Calculate the similarities between each statements in the code pair

Consider two types of similarities:

a) Text similarity

$$Sim(Text) = 1 - \frac{edit\ distance}{\sqrt{len\_buggy * len\_fixed}}$$

b) Structure similarity

$$Sim(Structure) = \frac{num\_common}{num\_total}$$

IV.2.2 Buggy & Fixed code reduction

```
context.registerReceiver(...);
------------------------------------------------------
context.getApplicationContext().registerReceiver(...);
```

(a) Code pair from the same answer post

```
Intent intent = context.registerReceiver(...);
------------------------------------------------------
context.registerReceiver(...);
```

```
Intent intent = context.registerReceiver(...);
------------------------------------------------------
context.getApplicationContext().registerReceiver(...);
```

(b) Code pairs from both the question and answer post

4. Filter out statements which has low similarity score

IV.2.3 Edit Script Generation

Use GumTree (edit script generation technique) to generate edit scripts for buggy code snippets.

1. Build mappings between the nodes of BuggyAST vs FixedAST
2. The edit script contains four type of operations on a node: add, delete, update, move
3. To overcome the problem of change position and rename variable of BuggyAST, the authors add 2 more operations in edit scripts: replace, copy

IV.2.3 Edit Script Generation

Algorithm 1 Generating a *replace* operation

n': the non-root node of $fixedAST$
p': the parent node of n'
i: the index of n' in p'
$newNode$: a newly added node
$N.mappedNode$: the mapped node of N in $buggyAST$

```
if n'.hasMapping then
    if n'.label ≠ n'.mappedNode.label then
        return replace(newNode, n'.mappedNode, n')
else
    if p'.hasMapping then
        p := p'.mappedNode
        if p.childNum>i then
            n := p.getChild(i)
            for each e' ← p'.children do
                if e'.mappedNode == n then
                    return NULL
            if n.label ≠ n'.label then
                return replace(newNode, n, n')
return NULL
```

IV.3 Patch Generation

Including 3 steps:

1. Extracting source code snippets from the target projects
2. Combining them with buggy code snippets from Q&A pages to obtain a list of buggy & source code pairs
3. Applying each edit script of the buggy code snippet to the corresponding source code snippet to obtain patches – Edit script application

IV.3.1 Extracting source code

Note: If the buggy code snippets is a block, the algorithm consists of 3 steps below:

1. Use call stack and buggy code to find fault locations
2. Expand each faulty location inside the candidate files, and combine them to obtain a buggy & source code pair
3. Choose the previous location and the next location with the same block size as two additional source code snippets.

IV.3.3 Edit script application

srcAST: source code snippet

1. Use GumTree to build mappings between buggyAST and srcAST

2. If there is unmapped node in the edit script, do not generate a fix.

3. For each buggy & source code pair in order, apply each transformed edit script to srcAST, and transform the edited AST back to code

4. Obtain a ranking list of generated patches

IV.3.3 Edit script application

Factors to sort generated patches

1. Q&A page ranking

2. Code pairs in the same answer post is ranked higher than those in both question and answer post

3. Faulty locations identified by the call stack is ranked higher than those identified by the buggy code

IV.4 Patch Filtering

Using the following two rules:

1. Merging. The approach may generate multiple patches that are equivalent. → Check the equivalence at the AST level, and merge them as one patch.

2. Compiling. If there is a compilation failure, filter out the patch.

Report the first k patches in the list to the programmer. In the experiment, k=1 because the author sees that there are high accuracy in generating the first patch.

V. Experiment

Product:

- Language: Java, open source tool: QACrashFix

Tools and Calculation score used:

- Search engine: Google
- Q&A pages: StackOverflow
- Eclipse AST parser (parse code to AST)
- GumTree (re-implement to build mapping and generate edit scripts)
- Sim(Text) = 0.8 and Sim(Structure) = 0.3
- Window 7, dual-core 2.50GHz Intel Core5 processor and 8GB memory

V. Experiment

- 24 issues as the final benchmark

- Generate a patch for each bug and comparing with developer’s patch

VI. Evaluation

TABLE I: Details of generated fixes

Project	Issue No.	Loc	#Edit Scripts	#Patches					Time (sec)		
				Initial	Equivalent	Compile Error	Remaining	Correct	First	Total	Compilation
Calligraphy	41	406	0	0	0	0	0	–	0.001	0.001	0
screen-notifications	23	846	6	1	0	1	0	–	30.205	30.205	12.187
TuCanMobile	27	2,849	8	20	2	12	6	Y	10.619	83.447	54.866
OpenIAB	62	7,053	8	1	0	0	1	Y	37.106	53.433	35.905
Android-Universal-Image-Loader	660	11,829	8	0	0	0	0	–	12.629	12.629	0
couchbase-lite-android	292	12,004	5	9	0	9	0	–	71.361	71.361	52.914
Onosendai	100	17,821	6	12	2	3	7	Y	6.845	70.080	62.945
LNReader-Android	62	21,276	3	1	0	0	1	Y	13.136	25.987	10.496
the-blue-alliance-android	252	24,094	5	1	0	1	0	–	15.949	15.949	7.099
open-keychain	217	31,038	9	9	1	6	2	Y	9.409	106.799	65.869
Ushahidi_Android	100	33,574	9	2	0	2	0	–	54.665	54.665	29.888
cgeo	457	36,963	8	11	1	3	7	N	15.500	93.372	62.235
cgeo	887	42,814	8	13	5	6	2	Y	5.729	43.697	34.343
TextSecure	1397	46,469	9	40	0	40	0	–	229.263	229.263	211.488
cgeo	2537	54,765	6	0	0	0	0	–	24.537	24.537	0
WordPress-Android	688	62,344	9	8	0	8	0	–	106.533	106.533	66.409
WordPress-Android	780	62,455	0	0	0	0	0	–	0.001	0.001	0
WordPress-Android	1320	62,895	9	5	1	3	1	Y	18.209	74.008	36.374
WordPress-Android	1484	65,307	1	0	0	0	0	–	9.133	9.133	0
WordPress-Android	1122	65,539	6	0	0	0	0	–	27.392	27.392	0
gnucash-android	221	68,158	11	0	0	0	0	–	7.146	7.146	0
cgeo	3991	68,202	12	8	0	3	5	Y	18.411	155.640	122.389
WordPress-Android	1928	71,485	8	1	0	0	1	N	14.122	35.444	12.891
calabash-android	149	93,146	10	30	0	30	0	–	161.855	161.855	143.842
Total	–	963,332	164	172	12	127	33	8	899.756	1492.577	1022.140

VI. Evaluation

RQ1: Effectiveness. How effective is our approach in fixing real-world recurring crash bugs?

1. For 7 of the 10 bugs, the tool generated correct patches. Among them, patches for 3 bugs are identical to those written by humans, and patches for 4 bugs are not identical, but are still correct.

2. For 1 of 10 bugs, the tool generated a patch using try and catch blocks as suggested in the Stack Overflow page,

3. For the rest 2 of the 10 bugs, the tool did not generate correct patches.

→ correctly fix 8 out of 24 bugs with only 2 potential false positives. (where 7 can be directly accepted)

VI. Evaluation

TABLE II: Performance of each step

Step	#Bugs unable to handle	#Total bugs in this step	Ratio
Edit script extraction	9	24	37.5%
Patch generation	3	15	20%
Patch filtering	2	12	16.7%

1. Fail to generate an edit script for 9 bugs, because there are no appropriate code pairs

2. Fail to generate a patch because cannot locate the buggy code as a result of incomplete crash trace

3. The remaining 2 bugs cannot be fixed because of compilation errors

VI. Evaluation

RQ1: Usefullness

Approach can complement existing bug-fixing approaches

TABLE III: Keyword matching in source code

Issue	Grep Command	Result
TuCanMobile #27	grep ”isShowing” -R .	N
OpenIAB #62	grep ”super.onDestroy” -R .	N
Onosendai #100	grep ”context.getApplicationContext” -R .	N
open-keychain #217	grep ”dismissAllowing” -R .	N
cgeo #887	grep ”image/jpeg” -R .	N
cgeo #887	grep ”image/*” -R .	N
LNReader-Android #62	grep ”super.onDestroy” -R .	N
Wordpress-Android #1320	grep ”commitAllowingStateLoss” -R .	N
cgeo #3991	grep ”isFinishing” -R .	N
cgeo #3991	grep ”\btry\b” -R .	Y
cgeo #3991	grep ”\bcatch\b” -R .	Y

THANK YOU