API Change-Driven Regression Test Selection on Android Applications

University of Idaho | Concordia University Texas | Texas State University

Tyler Clemens[1], Morgan Dozier[2], Jihan Rouijel[3], Guowei Yang[3]

1. University of Idaho; 2. Concordia University; 3. Texas State University

National Science Foundation WHERE DISCOVERIES BEGIN

Motivation

How does test selection on mobile apps affect development?

Mobile devices are ubiquitous, and thousands of devices run on the Android operating system. Mobile apps, which are frequently updated, need to be reliable for users. Developers need to constantly test apps to ensure high-quality products and performance.

Frequent updates to the underlying Android API (application programming interface; allows apps to access the operating system) can cause app functionality to break. Classes and methods dependent on previous API may be affected by the changes and cause issues.

Rerunning entire test suites can be time-consuming and expensive. Here we evaluate and explore test selection to select only relevant tests that cover code affected by the API change, which will ensure code functionality and prevent regression. This project will propose a novel approach to safe regression test selection to ensure reliability.

Approach

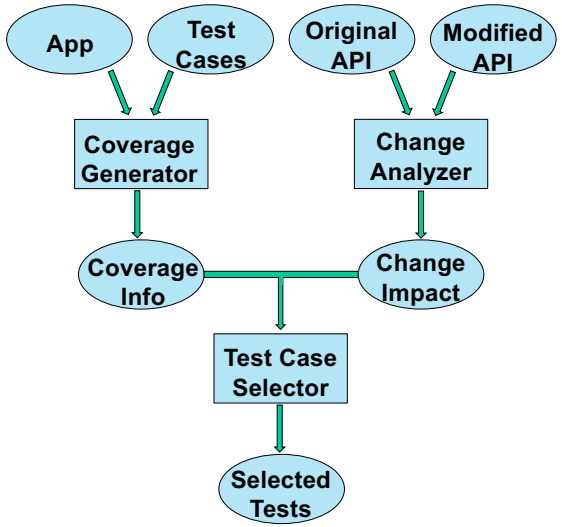

Fig. 1 Our approach to regression test selection.

- **The end goal is "safe" regression test selection**
  - Reveal same amount of bugs with fewer tests
  - Eliminate tests that don't find new bugs
- **Our proposal for this project will be to:**
  1) Design an approach to efficiently select tests
  2) Evaluate the approach for various Android applications

Evaluation

App Name	Type	Downloads
kouchat	Communication	5,000+
AnkiDroid	Education	1,000,000+
gnucash	Finance	100,000+
friendspell	Games	-
bookdash	News & Mags	100,000+
materialistic	News & Mags	10,000,000+
Launcher3	Personalization	500,000+
AmazeFileManager	Tools	1,000,000+
c:geo	Travel & Local	5,000,000+
Habitica	Productivity	1,000,000+
wikipedia	News & Mags	10,000,000+
K-9 Mail	Communication	5,000,000+
Financius	Finance	1,000+

Fig 2. Our list of apps.

Criteria for choosing apps:

- Large number of downloads to ensure reliability
- Range of app types to ensure representative mix

Apps will be built in API version 27

- Debugged the app builds
  - Built by Gradle
  - Reviewed unit tests
- Emulated built apps on Android Virtual Device
  - Ran tests using JaCoCo code coverage tool built into Gradle

```
@Test
public void sendFileShouldThrowExceptionIfUserIsNull() throws (
    expectedException.expect(IllegalArgumentException.class);
    expectedException.expectMessage( expectedExceptionMessage: "Use

    controller.sendFile( user: null, mock(FileToSend.class));
}

@Test
public void sendFileShouldThrowExceptionIfFileIsNull() throws (
    expectedException.expect(IllegalArgumentException.class);
    expectedException.expectMessage( expectedExceptionMessage: "File

    controller.sendFile(mock(User.class), file: null);
}
```

Fig 3. A set of unit tests for KouChat.

JaCoCo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
org.jacoco.examples		58%		64%	24	53	97	193	19	38	6	12
org.jacoco.core		97%		93%	102	1,324	105	3,206	21	699	2	134
org.jacoco.agent.rt		78%		86%	30	120	58	305	21	74	7	20
jacoco-maven-plugin		90%		81%	35	183	44	407	8	110	0	19
org.jacoco.cli		97%		100%	4	109	10	275	4	74	0	20
org.jacoco.report		99%		99%	4	562	2	1,331	1	369	0	64
org.jacoco.ant		98%		99%	4	163	8	428	3	111	0	19
org.jacoco.agent		86%		75%	2	10	3	27	0	6	0	1
Total	1,322 of 26,576	95%	136 of 2,022	93%	205	2,524	327	6,172	77	1,481	15	289

Fig. 4 JaCoCo coverage report.

- JaCoCo generates coverage reports
  - Covers class, method, and line-level
  - Highlights source code covered by the test(s).

```
38  public class AdBlocker {
39      private static final String AD_HOSTS_FILE = "pgl.
40      private static final Set<String> AD_HOSTS = new H
41
42      public static void init(Context context, Schedule
43          Observable.fromCallable(() -> loadFromAssets(
44                  .onErrorReturn(throwable -> null)
45                  .subscribeOn(scheduler)
46                  .subscribe();
47      }
```

Fig. 5 Code covered by JaCoCo.

- Code coverage is performed for every individual test.
- We are checking for coverage on the method-level.

- Methods affected by API change
  - Data scraped from Android website
  - Can be combined with the coverage reports to mark affected tests

Changed Packages
android
android.accessibilityservice
android.accounts
android.animation
android.app
android.app.admin
android.app.assist
android.app.backup
android.app.job

Fig. 6 Table of packages affected by API change.

Discussion

- There are several reports available from JaCoCo for debug and release builds. Exploring other code coverage tools found that JaCoCo produces the best results for tracing coverage of individual test cases.
- By comparing the methods used in our apps' test cases to the methods listed as affected by subsequent API change, we can identify "safe" tests to select to prevent app regression.
- API change can affect the functionality of the app source code (when associated classes and methods are changed) and reduce reliability.

Future Work

1. Select tests based on changed methods targeted by the JaCoCo coverage reports.
2. Apply the framework to more open-source apps to further evaluate how our framework works with any given app.
3. Clean the automation script to accept user inputs such as specified device type, API, and file location.

References

S. Yoo, & M. Harmon. Regression Testing Minimisation, Selection and Prioritization: A Survey. *Software Testing, Verification, and Reliability, 00*, 1-7. (2007).

Q. Do, G. Yang, M. Che, D. Hui, & J. Ridgeway. Redroid: A Regression Test Selection Approach for Android Application. In *Proceedings of The 28th International Conference on Software Engineering and Knowledge Engineering, 2016.*

G. Yang, A. Moninger, J. Jones, & M. Che. How Do Android Operating System Updates Impact Apps? In *5th IEEE/ACM International Conference on Mobile Software Engineering and Systems, 2017.*

M. Linares-Vasquez, C. Bernal-Cardena, K. Moran, & D. Poshyvanyk. How do Developers Test Android Applications? In *Proceedings of The IEEE International Conference on Software Maintenance and Evolution (ICSME), 2017.*

X. Li, N. Chang, Y. Wang, H. Huang, Y. Pei, L. Wang, & L. Xuandong. ATOM: Automatic Maintenance of GUI Test Scripts for Evolving Mobile Applications. In *Proceedings of The 10th IEEE International Conference on Software Testing, Verification and Validation, 2017.*

Acknowledgements

We thank the National Science Foundation for funding the research under the Research Experiences for Undergraduates site programs (CCF-1659807) at Texas State University to perform this piece of work and the infrastructure provided by an NSF-CRI 1305302 award.