Operating System Change Impact Analysis on Android Applications

Jeffrey Jones, Austin Moninger, and Guowei Yang, PhD
Department of Computer Science, Texas State University, San Marcos, TX

Abstract

People are incredibly dependent on mobile devices for computation, storing private information, and navigation among others. Mobile operating systems, particularly Android, face frequent updates that can affect the functionality of apps. Yet, there are few tools out there for Android for automating test selection to cope with the frequent updates. We seek to develop a framework that can inform Android developers which portions of their app's code has been affected by the OS change.

Type	Additions	Changes	Removals	Total
Packages	18	82	0	100
Classes and *Interfaces*	145	349	4	498
Constructors	23	16	0	39
Methods	795	139	18	952
Fields	572	69	0	641
Total	**1553**	**655**	**22**	**2230**

Figure 1. Android provides change statistics for their API updates. This is information based on API 26.

Problem

- Android mobile devices are nearly ubiquitous, but since the products are relatively young, there are fewer testing tools for developers
- Application functionality may be affected by updates, yet constantly testing after frequent updates is very expensive
- Developers are "blind" to what is being updated within the operating system
- Android developers must manually search the API Difference Report in order to determine which sections of their application's code may be affected by an OS update (or they may be unaware of the Difference Report altogether)

```
TipCalculatorActivity.java
   <--package--> android.app
                  <--class--> Activity
                              <--method--> findViewById(int)
                              <--method--> onVisibleBehindCanceled()
                              <--method--> requestVisibleBehind(boolean)
                              <--method--> enterPictureInPictureMode()
                              <--method--> onMultiWindowModeChanged(boolean)
                              <--method--> onPictureInPictureModeChanged(boolean)
   <--package--> android.content
                  <--class--> Intent
                              <--field--> ACTION_DEVICE_STORAGE_LOW
                              <--field--> ACTION_DEVICE_STORAGE_OK
                              <--field--> EXTRA_SHORTCUT_ICON
                              <--field--> EXTRA_SHORTCUT_ICON_RESOURCE
                              <--field--> EXTRA_SHORTCUT_INTENT
                              <--field--> EXTRA_SHORTCUT_NAME
   <--package--> android.view
                  <--class--> View
                              <--method--> findViewById(int)
                              <--method--> findViewWithTag(java.lang.Object)
```

Figure 2. Example output of the affected user defined classes using our framework.

Approach / Framework

- Develop a framework to provide developers with an OS change impact analysis on their application
- Compare libraries being used within the application to the updated libraries in the Android Difference Report
- Report the impacted classes within the updated libraries to show the affected methods/fields within the application
- Report statistics to quantify the level of change impact on their application
- Minimize the amount of testing needed to be done

```
OS CHANGE IMPACT:

Imported libraries in the app:                      24

Imported packages in the app that were updated:     11 out of 24
PACKAGE LEVEL IMPACT:                               45.83%

Imported classes in the app that were updated:      8 out of 24
CLASS LEVEL IMPACT:                                 33.33%
```

Figure 3. Example output of OS Change Impact statistics using our framework.

Results

- Framework quantifies the operating system change impact for developers
- The more fine-grain the analysis is, the more accurate the change impact statistics will be (i.e., the class level impact is more accurate than the package level impact)
- Our framework allows developers to see what has been impacted within their application and can run tests accordingly

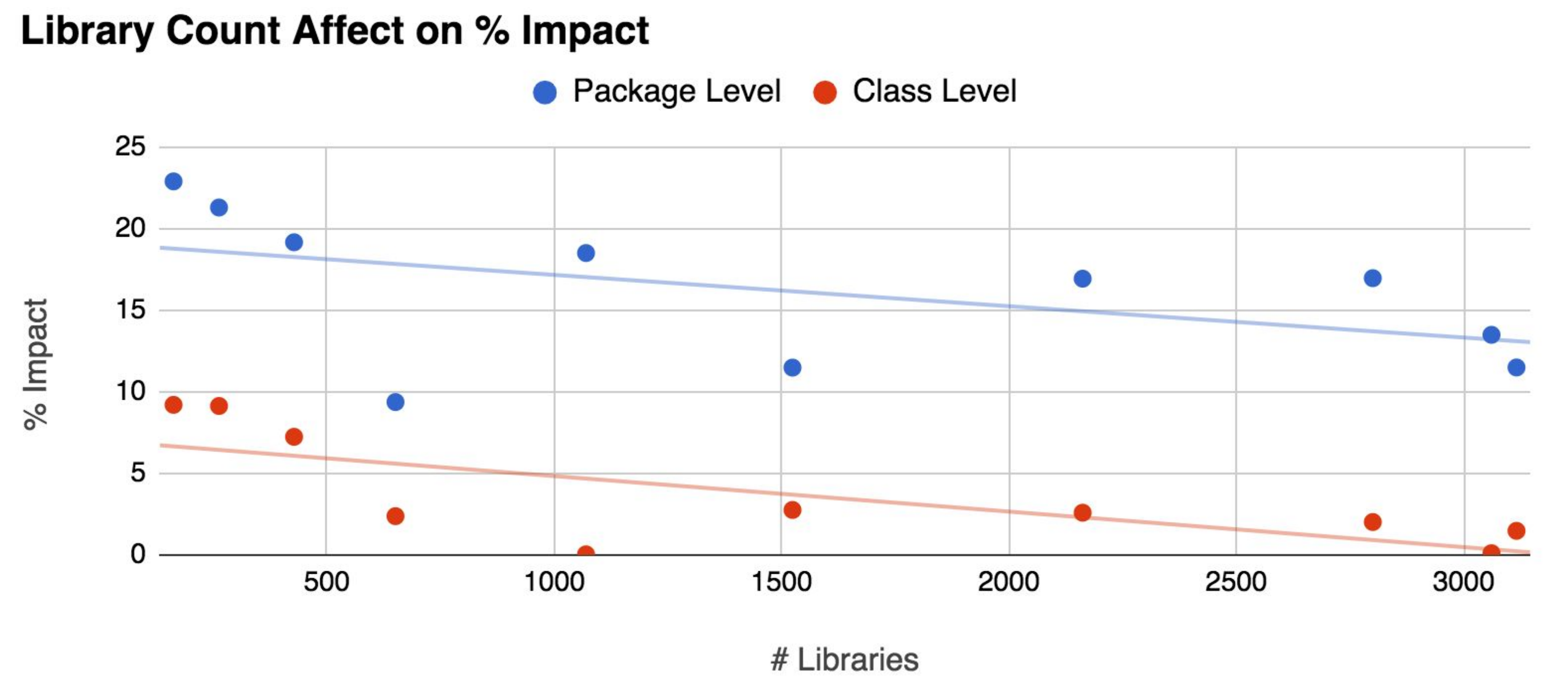

Figure 4. Graph of percentage change impacts for ten miscellaneous applications that we have analyzed.

Conclusions and Future Work

- Overall change impact is low and larger scale applications are impacted less by each update than smaller scale applications
- Use our framework to automate test case selection for Android applications
- Perform finer-grain analysis of the change impact

Acknowledgements

We thank the National Science Foundation for funding the research under the Research Experiences for Undergraduates site programs (CNS-1358939, CNS-1659807) at Texas State University to perform this piece of work and the infrastructure provided by a NSF-CRI 1305302 award.

Contact

Jeffrey Jones:	jeffjones2994@gmail.com
Austin Moninger:	austinmoninger@gmail.com
Dr. Guowei Yang:	gyang@txstate.edu

References

1. O. Alessandro and G. Rothermel. Software Testing: A Research Travelogue (2000-2014). In *Proceedings of the on Future of Software Engineering - FOSE*, pages 117-132, 2014.
2. C. Hu and I. Neamtiu. Automating GUI Testing for Android Applications. In *Proceedings of The 6th International Workshop on Automation of Software Test - AST*, 2011.
3. Choudhary, S. R., Gorla, A., & Orso, A. (2015). Automated Test Input Generation for Android: Are We There Yet? (E). *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*.
4. Q. Do, G. Yang, M. Che, D. Hui and J. Ridgeway. Redroid: A Regression Test Selection Approach for Android Applications. In *Proceedings of The 28th International Conference on Software Engineering and Knowledge Engineering, 2016.*