**Title**

Domenico Amalfitano , Anna Rita Fasolino, Porfirio Tramontana, Salvatore De Carmine, Atif M. Memon. 2012. Using GUI ripping for automated testing of Android applications. In proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering

**Keywords**

ii1 *Testing Automation* : A testing scenario where derivation of test cases for both crash and regression testing can be automated.

ii2 *Event* : A user action like a command input or a data input

ii3 *Action* : Zero or more data input events followed by single command input events.

ii4 *Task* : A couple (action, GUI state). A task is executed by reaching the GUI state and then performing the action

**iii1 Motivation**:

The motivation for this work comes from the growing sales of Android devices and an increasing demand for suitable techniques and tools to support their testing. Also authors believe that Android application testing is challenging, though developed using Java, they differ from standard Java client-server model.

**iii2 Related Work:**

A model based approach for Android GUI Testing by Takala et al.

The technique describes GUI of an application by state machine. After a detailed static or dynamic analysis a formal model is obtained. Generation algorithms then use this detailed model to produce test cases.

A GUI Crawling-Based Technique for Android Mobile Application by D. Amalfitano et al.

This approach uses a tool that explores application GUI by simulating user events and constructing a GUI tree model

**iii3 Baseline results**

Android Ripper was compared against Monkey, a random testing approach. Both the techniques were tested on an open-source Android application called “Wordpress for Android”. Release r394 of the Android application was used for testing. This release source code consists of 6 java packages, a total of 71 files containing 334 classes and 1,489 methods and total of 10017 executable lines of code. Monkey took 4.46 hours to find 3 crashes corresponding to bug B2 and reached 25.27 % line of code coverage. On the other hand Android Ripper took less than 5 hours to find 8 crashes related to 4 bugs and reached 38 % line of code coverage.

**iii4 Future Work**

AndoridRipper achieves 38% line of code coverage and it takes around 5 hours for Round 2 and Round 3. Future work can focus on achieving higher code coverage and taking lesser time.

**Scope for improvement**

**iv1** The paper states that model development is the biggest problem in using the existing work for Android application testing and hence there was a need for something like Android ripper. But authors do not clearly specify how Android ripper leverages the existing work.

**iv2** The metric used for comparing the two technique were line of code coverage and defect detection effectiveness. There can be more metrics that can be used for comparison but no other metric was mentioned by the authors. Paper could have also discussed why these particular metrics were selected from the bunch of available metrics. Also we feel it would have been better if Android Ripper and Monkey were compared on all possible metrics. This would have given more insight about weaknesses of the new automated testing framework.

**iv3** Authors failed to explain why they selected “Wordpress for Android” for testing their techniques. They mentioned that this android application is open source but apart from that no other explanation was given for choosing it. It would have been great if they could have discussed about the factors they considered while making this choice.

**iv4** Also, the authors mentioned about using Robotium Framework and the Android Instrumentation class for implementing Android Ripper. We feel the paper could have been more clear about how these frameworks were actually used in implementation.

**Connection to other papers**

An event-flow model of GUI-based applications for testing by Atif M. Memon et al., 2007.

*AndroidRipper* takes the idea of graphical object with fixed set of properties from this work. At any set of such properties constitutes the state of the GUI

Automating GUI testing for Android applications by Cuixiong Hu et al., 2011.

This Android specific testing technique that is event-based and focuses on Activity, event and dynamic type of error is used in *AndroidRipper*.