HW2: Paper from 2011

i) *Reference*:

Cuixiong Hu, Iulian Neamtiu. Automating GUI Testing for Android Applications.

In Proceedings of the 6th International Workshop on Automation of Software Test 2011.

ii) *Keywords*:

1) **GUI testing**: GUI testing tests whether the activity is performed as per the GUI specification. To perform GUI testing, automatic event generation technique called Monkey event generator was used to generate all the GUI events.
2) **Activity bugs**: Activity bugs stem from incorrect implementation of the Activity class. This results in a creation or destruction of activity in a wrong way so that it will make the application crash.
3) **Event bugs**: An Android Application should be developed taking into consideration all the possible states and transitions. If developers fail to provide proper implementations of event handlers associated with certain states, the application can either enter an incorrect state or crash outright.
4) **Type Errors**: Type errors result from runtime type exceptions. Unhandled exceptions are exceptions the user code does not catch and lead to an application crash.

iii) Paper Contents:

1) **Motivational Statements**: The adaptation of mobile devices for computational needs has increased exponentially in recent times. This has led to establish that ensuring reliability of mobile applications is an important task. However, the physical constraints of mobile devices(small memory, small display, low-power CPU etc.) and novelty of mobile platforms to developers makes it prone to new kinds of bugs. This paper addresses this issue by building a test automation approach to detect Android bugs.
2) **Baseline Results**: An Android application called *connectbot* was used for experimentation purpose and the result has been summarized in the following table:

		Errors		
		Activity	Event	Type
Results	User testing	8	21	4
	Automated testing(prev+new)	8+3=11	18+6=24	4+0=4

3) **Study instruments**: The Android bugs were detected using the proposed automation test cases and event generation tool were used combined with the analysis of log files generated during runtime. These bugs were categorized into non-overlapping categories for better gauging of efficiency.
4) **Future work**: This paper mainly deals with bugs belonging to only 3 categories viz. Activity, Event and Type. It is very desirable to have a model-based approach to identify other types of bugs such as I/O errors, API errors, Concurrency errors etc. This involves modelling the I/O and concurrency as state machines and using tools such as WALA to permit pattern and state-machine violations to obtain test results.

iv) *Needs Improvement:*

1) The bugs that are detected are corrected based on the log file analysis. This can result in overlooking of the bugs that are not logged.
2) The paper addresses the dynamic verification needs to discover bugs at the GUI level. However, it fails to address the static verification such as studying fixes and source code. The paper assumes that the number of bugs will be highest at the GUI layer without providing any empirical proof of the same.
3) The automated test model proposed in this paper does not provide 100% coverage which is evident from 3 missed event bug reports which have been already reported by users.
4) It covers only a part of the total categories of bugs and a model providing reporting of all the other bugs as well is highly desirable.