A Toolset for GUI Testing of Android Applications

Domenico Amalfitano
Anna Rita Fasolino
Porfirio Tramontana
Salvatore De Carmine
Gennaro Imparato

Dipartimento di Informatica e Sistemistica
University of Naples Federico II, Italy

Context and Motivation

Android OS is currently the most diffused platform for mobile devices [Gartner, 2012 August]

- 98 Millions of Mobile Device sales with Android OS in second quarter 2012
- Android owns 64% of Market Share

Google Play market now includes over than 500,000 apps

- Most of these apps have been developed by small programmer teams, by evolutionary life cycles with very frequent releases via Google Play

There is a strong need for fast and effective Testing automation solutions

Current Technologies and Tools for Android Testing

Testing Frameworks for Unit and GUI testing

- Robotium, MonkeyRunner, Calculon, Roboelectric, …

Stress testing Tools such as Monkey

Testing tools from research projects:

- TEMA model-based testing tools [Harty], …

Debugging tools

- DDMS debugging tool from the Android SDK

Monitoring tools

- platforms for monitoring app usages and reporting app crashes to the developers (like Crittericism)

Our Proposal: A toolset for automatic GUI Testing of Android apps

- GUI testing is an effective approach for testing an app through its GUI, firing sequences of events and providing it with user inputs

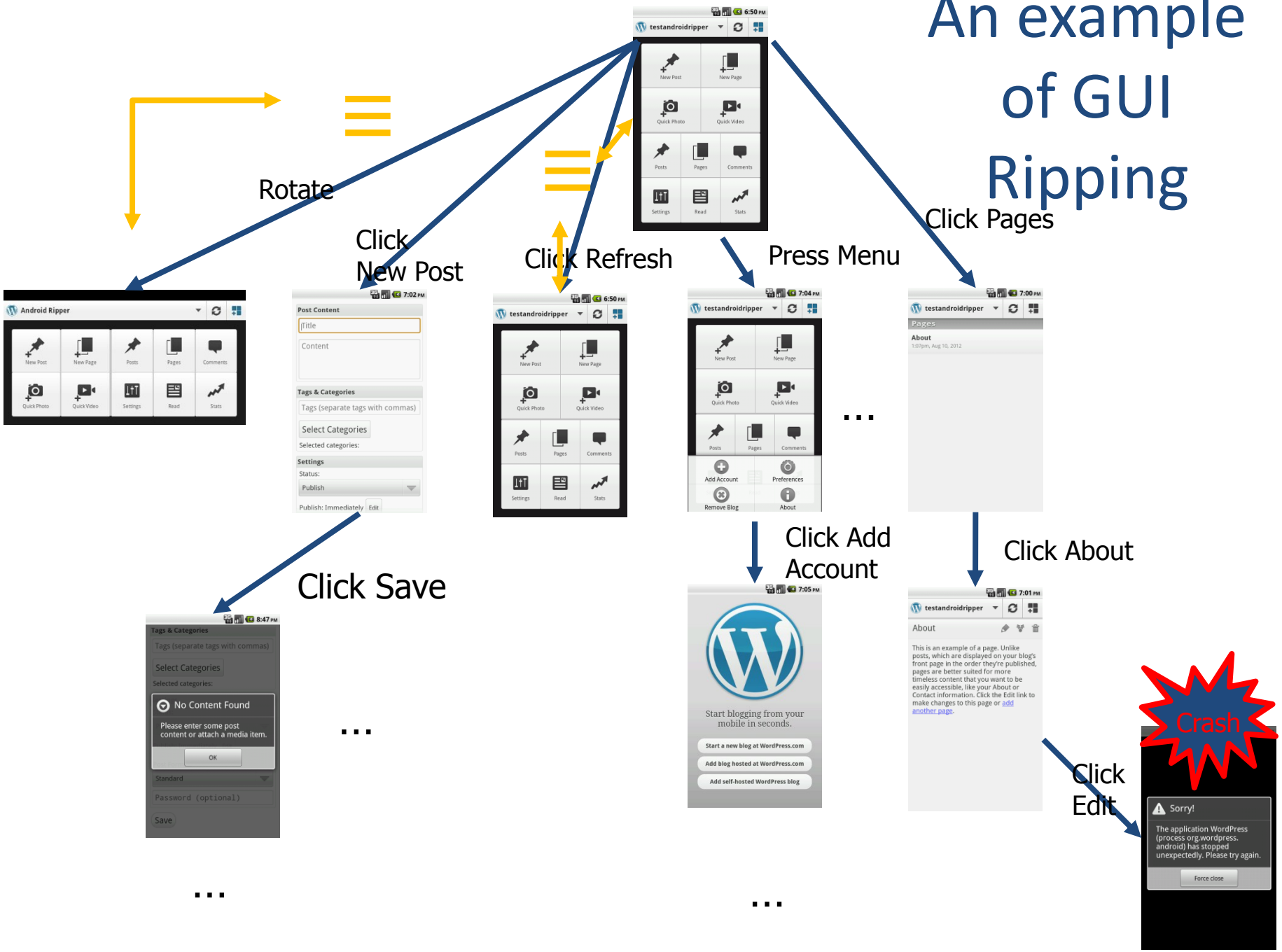
- The proposed GUI testing toolset is based on a ***configurable*** GUI Ripper (Android Ripper) that:
  - automatically traverses the GUI firing sequences of events through the UI or the device (device rotation, press back button, etc.)
  - implements a Black-Box testing approach
  - obtains automatically JUnit test cases (usable for regression testing)

Android Ripper Features

The Ripper automatically explores the GUI structure and returns a GUI Tree model

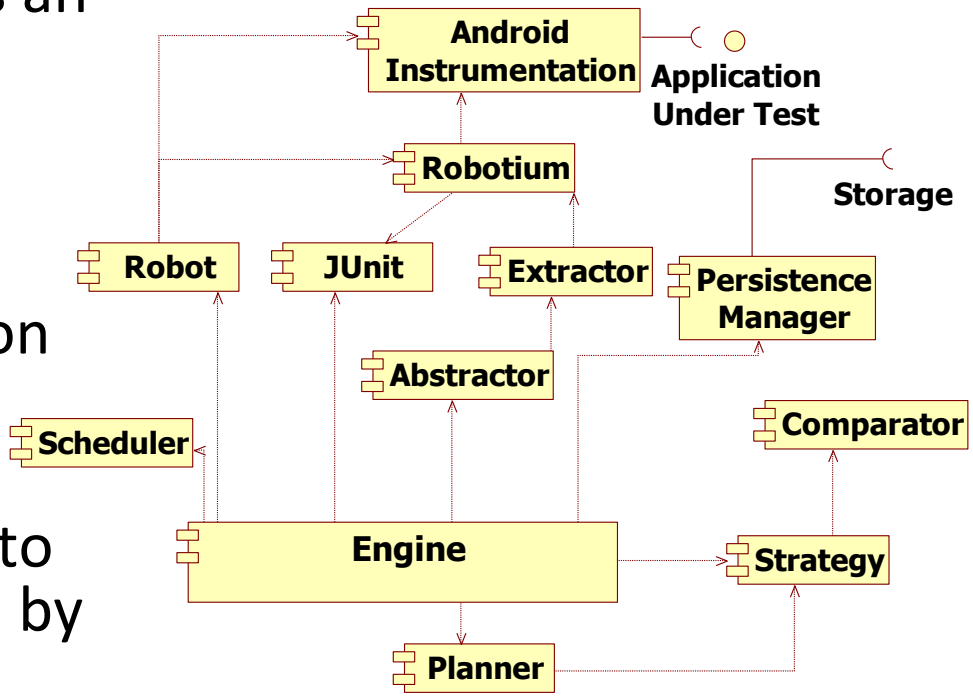
- It can be configured in order to emulate several kinds of user behaviours, choosing:
  - types of event to be fired, input values, time delays between events, GUI traversing strategy (e.g. depth first, breadth first, etc.), termination criterion to stop GUI traversal ...
- It detects runtime crashes (due to unhandled Java exceptions) and returns the sequences of events causing them

An example of GUI Ripping

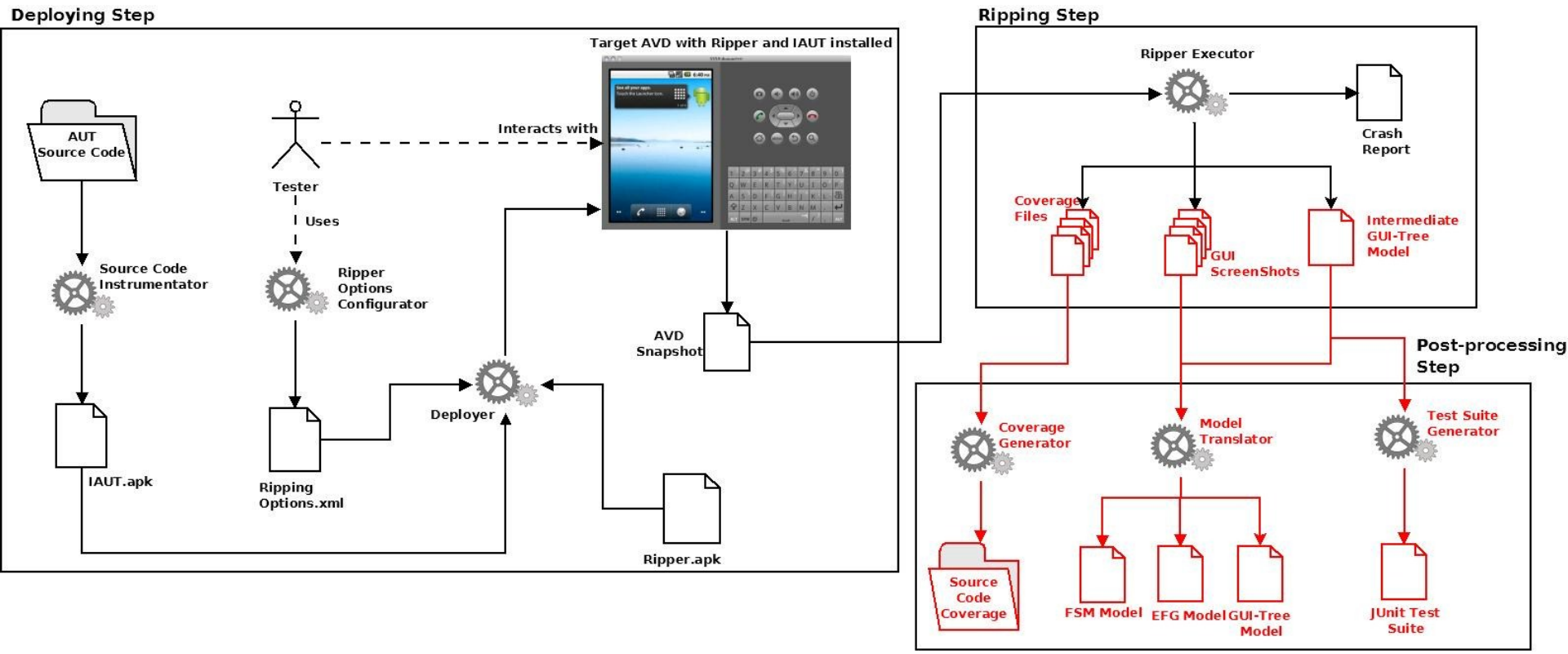


Android Ripper Technological Details

- The Ripper is implemented as an Android JUnit Test Project.
- It exploits the Robotium framework and Android Instrumentation classes to interact with the Application Under Test (AUT)
- The AUT source code can be instrumented with EMMA to calculate the code covered by the Ripper.
- The Saving SnapShot feature offers a mechanism to define the preconditions of the AUT



Toolset and Testing procedure



Experiments

We tested some open source apps from Google Play market using Android Ripper, finding several unknown bugs (reported to the developers)

A. U. T.	# Events	Ripping Time (hours)	# Bugs	# Crashes	% LOC Coverage
BookCatalogue	1,677	61	2	9	57
AarDdict	93	2.2	1	5	70
Tomdroid	86	1.25	1	13	39
Wordpress	486	6.16	6	31	43

Future Works

- We plan to extend the Ripper :
  - By implementing new event firing strategies
  - By considering new sources of events (sensors) and new input selection strategies.

- We will address the technological issues affecting the Ripper execution on real devices.

- We will be pleased to further explain the toolset usage in the Tool Bazar !