3th International Workshop on TESTing Techniques & Experimentation Benchmarks for Event-Driven Software

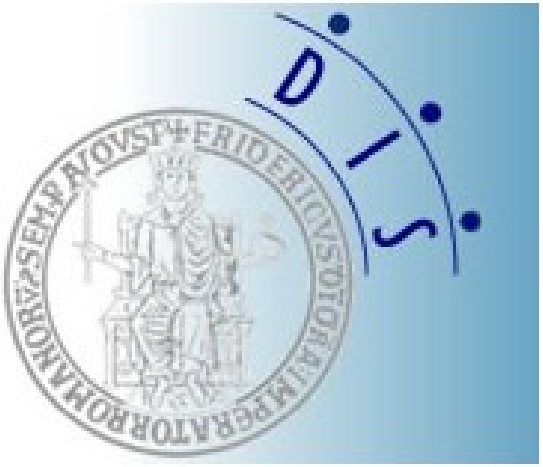
Berlin, Germany
March 21, 2011

A GUI Crawling-based technique for Android Mobile Application Testing

University of Naples "Federico II", Italy

Dipartimento di Informatica e Sistemistica

- Domenico Amalfitano
  Anna Rita Fasolino
  Porfirio Tramontana



Context and Motivations

- Context:
  - Testing of Android Applications.
- Motivations:
  - Dealing with open issues in Android testing.
  - Introducing a new GUI based testing approach.

What experts says about Android

- “There should be nothing that users can access on their desktop that they can’t access on their cell phone”; *Andy Rubin, Guru for Google's Android.*

- “Android is predicted to become the second largest mobile Operating System by 2012 and challenge Symbian for the first position by 2014”; *Gartner Newsroom.*

What is Android

- Android is a software stack for mobile devices that includes a Linux-based operating system, middleware and core applications.
  - Is possible to access the stack resources and develop applications on the Android platform using the Java programming language through the tools and the APIs provided by the Android SDK.

Android Devices

- Android operating system is often installed on smartphone devices that usually have limited hardware resources (like CPU or memory) and a small-sized screen.
  - Equipped with a large number of sensors and communication devices such as a microphone, wi-fi and Bluetooth chips, GPS receiver, single or multi touch screen, inclination sensors, camera, etc.

Android Applications

- An Android application is composed of several types of Java components instantiated at run-time:
  - Activities, Services, Intents, Broadcast Receivers and Content Providers.
- Activity component:
  - is responsible for presenting a visual user interface;
  - provides a screen with which the users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.
- An Android application usually includes one or more Activity classes that extend the base Activity class provided by the Android development framework.

Android Applications are EDS

- The processing in Android applications is event-driven.
- Two types of events can be triggered:
  - user events (such as Click, MouseOver, etc.) that can be fired on the user interface items (like Buttons, Menu, etc.);
  - events that are triggered by other input sources, such as GPS receiver, phone, network, etc.
- Events are handled by Event Handlers.
- An Event Handler is a piece of code *registered* to an Event Listener.

Open Issues in Android Testing

- Are the approaches already available for EDS testing still applicable to Android applications?
- How the techniques used for EDS testing can be adopted to carry out cost-effective testing processes in the Android platform?

EDS Testing: current approaches

- EDS testing techniques described in the literature:
  - are based on suitable models of the system or sub-system to be tested (EFG, EIG, or FSM);
  - test cases are derived:
    - exploiting the analysis of user session traces;
    - translating possible sequences of automatically generated events, i.e. generated by a GUI ripper or a Web Crawler.

EDS testing strategies in the context of Android testing

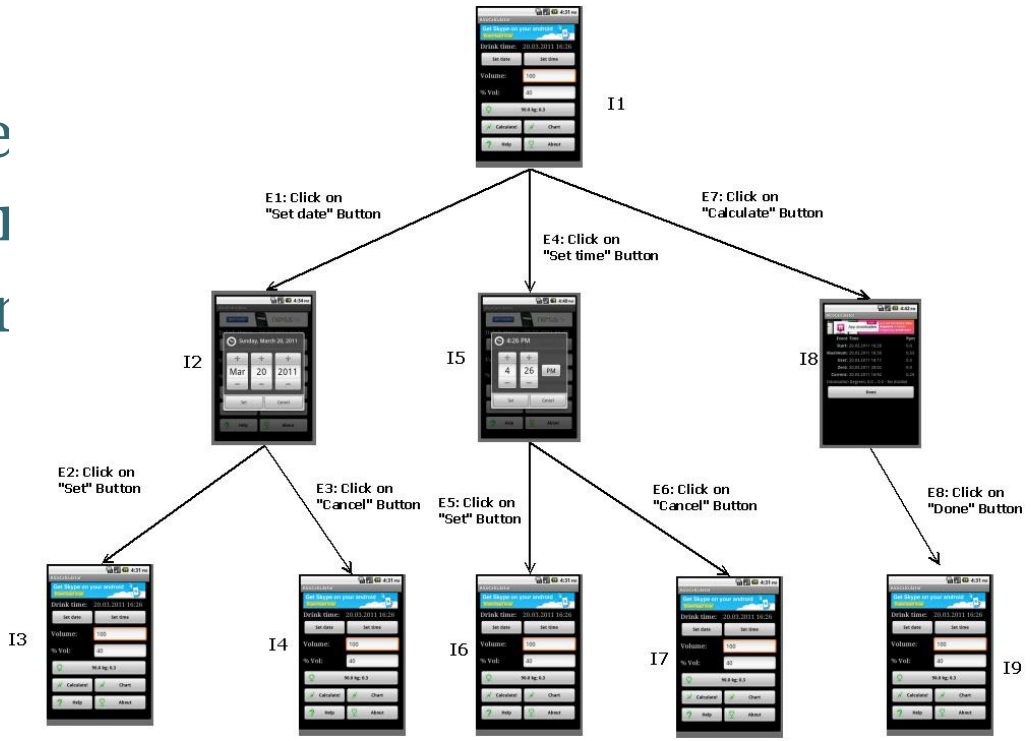
- In order to take into account the peculiar types of event and input source that are typical of Android devices, an adaptation of the models and strategies for the aims of Android testing is required. As consequence:
  - new reverse engineering and GUI ripping techniques for obtaining the necessary proposed models have to be designed;
  - platforms and tools aiding user session analysis and implementing the proposed GUI ripping techniques have to be developed.

The proposed technique for testing Android applications

- An automatic testing technique for Android applications, based on a Crawler that simulates real user events on the user interface, is proposed.
  - The Crawler infers automatically a GUI model of the application.
  - The obtained GUI model is used for deriving test cases that can be automatically executed for different aims, in particular:
    - crash testing;
    - regression testing.

The proposed model

- The model produced by the crawler is actually a GUI Tree, where:
  - the nodes represe Android applicatio
  - the edges descr between nodes.

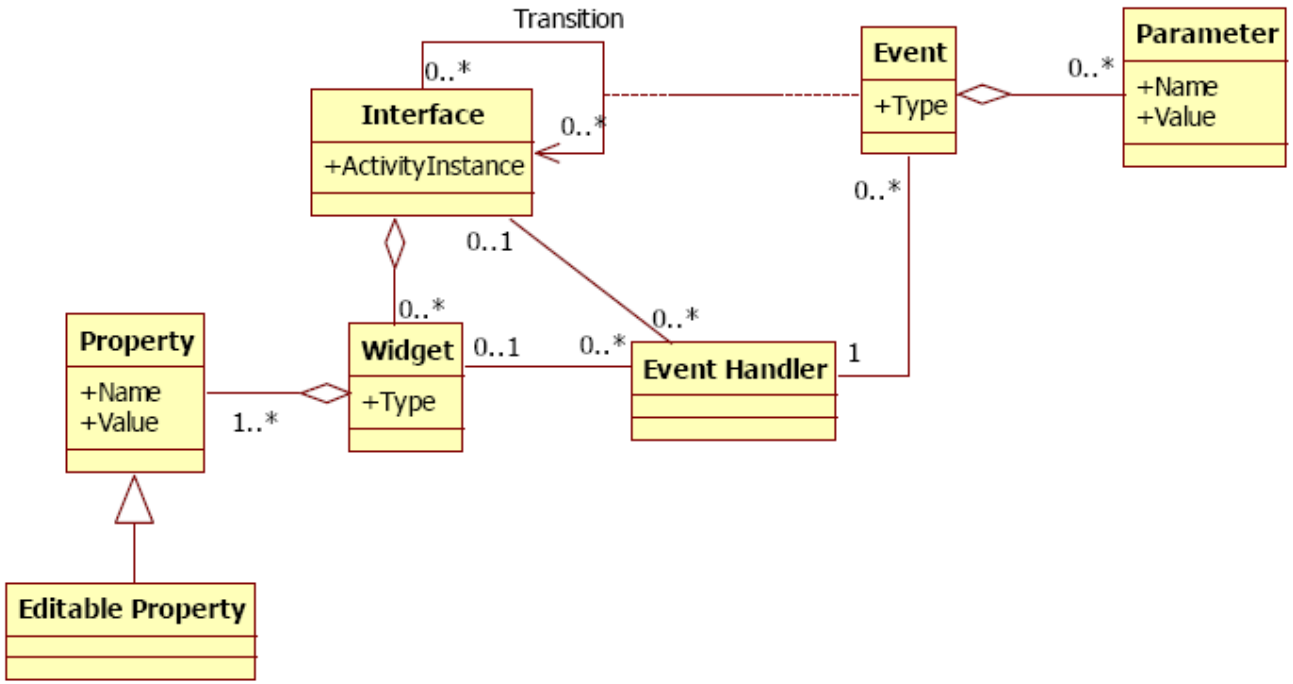


The GUI Crawler

- The crawler builds the GUI tree implementing a depth first search using an iterative algorithm.
- The crawler fires events on the application user interface capturing data about interfaces and events that will be also used to decide the further events to be fired.
- During the GUI exploration the crawler is able to perform a first crash testing.

Conceptual Model of an Android Application GUI

- The data analysed by the crawler at run time belong to the conceptual model of an Android GUI:

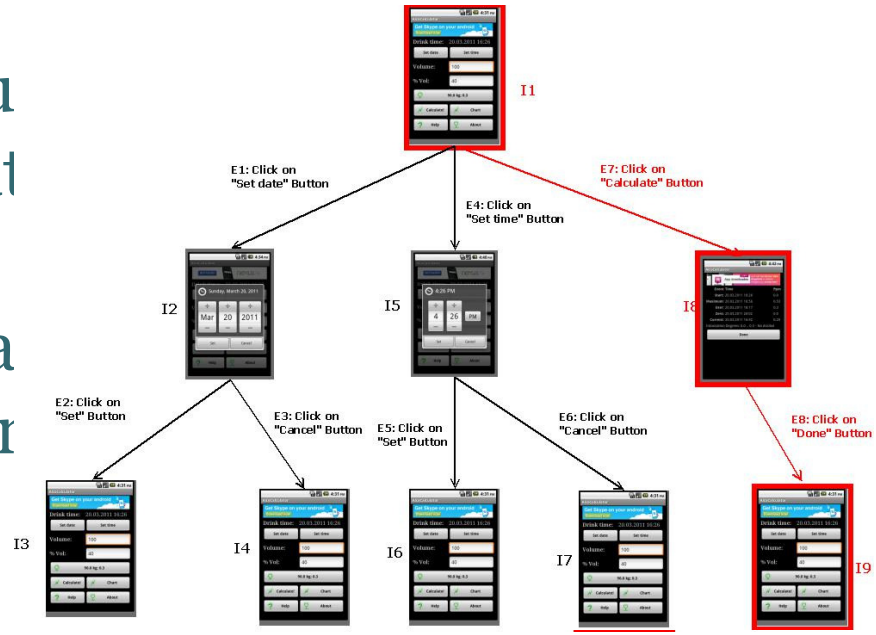


Critical aspects of a GUI Crawler

- What's the criterion used for check if two interfaces are equivalent?
  - Two interfaces are equivalent if they have the same Activity Instance attribute and they have the same set of Widgets, with the same Properties and the same Event Handlers.
- What's the approach used for defining the values of widgets properties and event parameters that must be set before firing a given event?
  - The crawler assigns them random values.

Test case definition

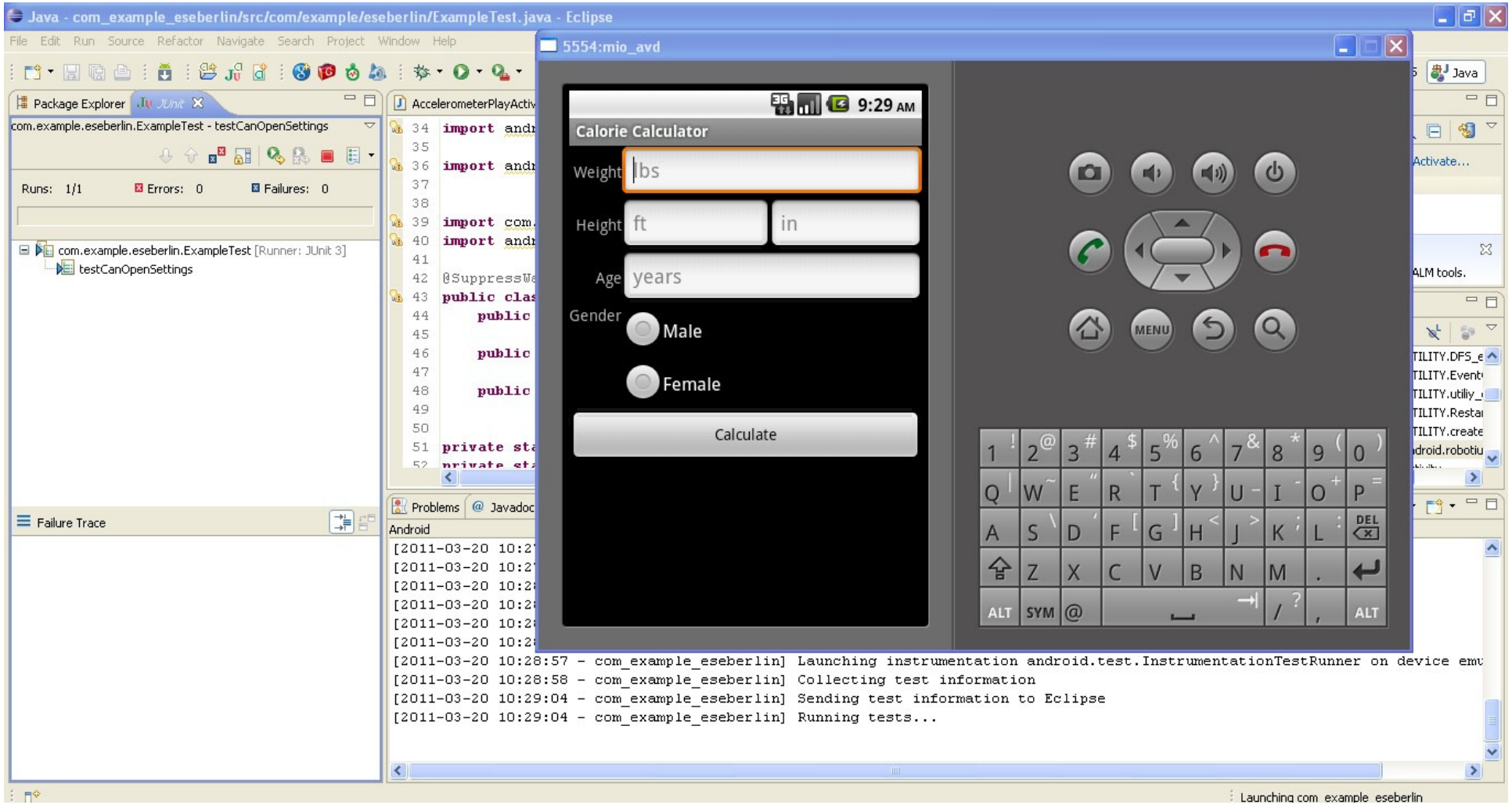
- The GUI tree is the starting point for obtaining test cases (TC).
  - TC are given by the sequ with GUI tree paths that leaves of the tree.
  - TC are used to detect cra in regression testing scer



The Testing Tool

- A tool, named A²T² (Android Automatic Testing Tool), is developed in Java technology to support the testing technique proposed.
- A²T² is composed of two main components:
  - GUI Crawler;
  - Test Case Generator.
- Both the components:
  - exploit the Robotium framework originally designed for supporting testing of Android applications;
  - are executed in the context of the Android Emulator provided by the Android SDK.
- Produced test cases are actually Junit test cases.

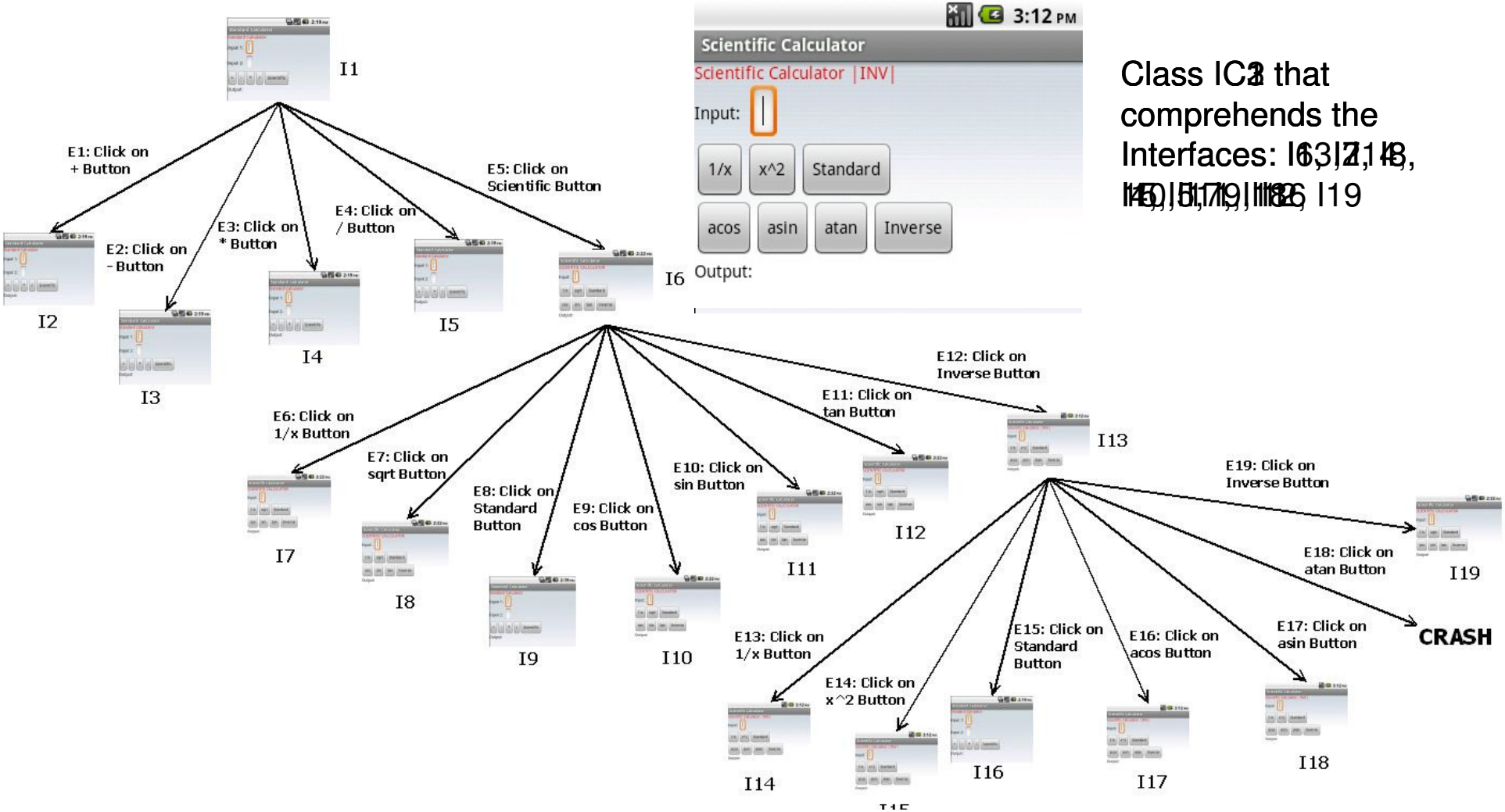
A screenshot of the tool in execution



Example

- An example of using the proposed technique and tool for testing a simple Android application that implements a mathematic calculator for the Android 2.2 platform is shown.
- A GUI tree of the application is obtained by the crawler.
- During crawling:
  - 19 Events are triggered,
  - 19 Interfaces are obtained,
  - 3 Class of Interfaces are obtained,
  - An exception causing an application crash occurs.

The obtained GUI-Tree



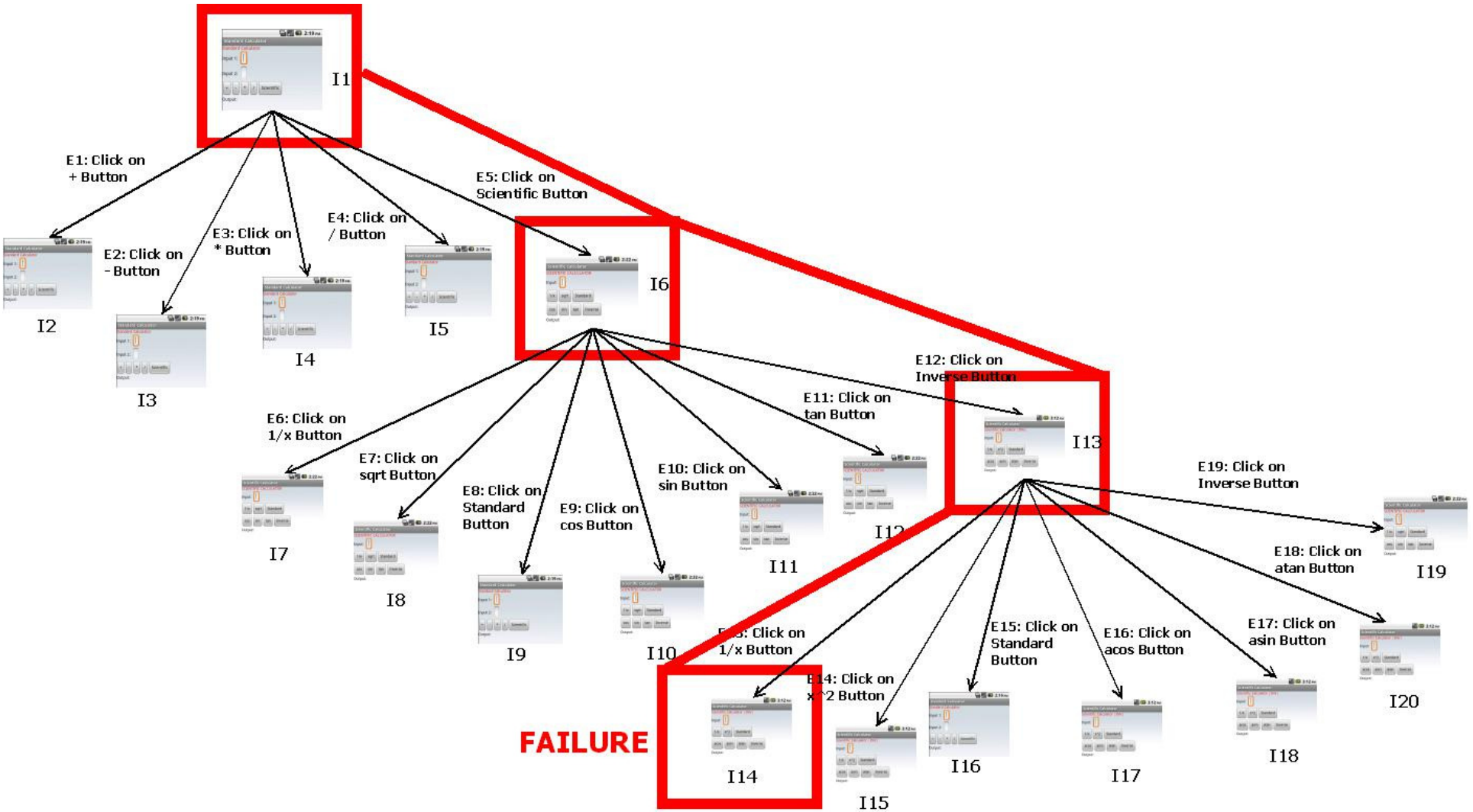
Class IC3 that comprehends the Interfaces: I6, I7, I8, I9, I10, I11, I12, I13, I14, I16, I17, I18, I19

Crash detection

- While exploring the GUI interfaces via the crawler some crashes of the application can be discovered.
- A crash occurs after firing the E18 Event:
  - click on the "*atan" Button on the Interface I13.*
- The crash is caused by the lack of a try/catch code block for handling the exception due to the input of a non-numeric value in the Input TextEdit widget.
  - This missing cause a java.lang.NumberFormatException when the application tries to convert the string in the input field into a Double value before computing the arctangent function.
- After the correction of this defect, we run the crawler again obtaining a new GUI tree and another instance of Interface (belonging to IC3 group).

Regression Test Cases

- Test Case Generator produces 17 test cases that correspond to the 17 different paths from the root to the leaves of the obtained GUI tree for the regression testing.
- In order to assess the effectiveness of the test cases for the aims of regression testing, two faults in the Android application are injected and the 17 regression test cases to find these faults are run.

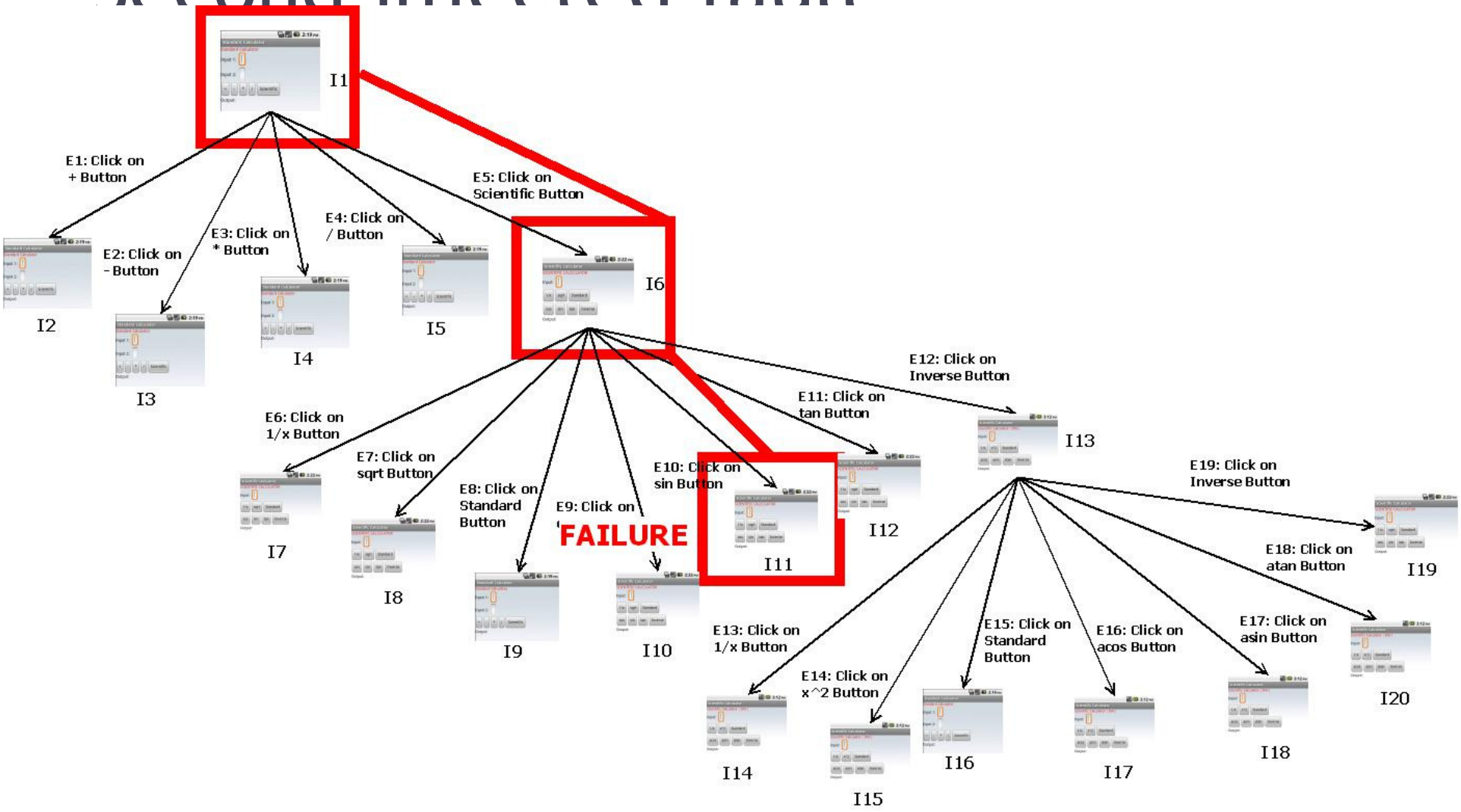

I1
E1: Click on + Button
E2: Click on - Button
E3: Click on * Button
E4: Click on / Button
E5: Click on Scientific Button
I2
I3
I4
I5
I6
E6: Click on 1/x Button
E7: Click on sqrt Button
E8: Click on Standard Button
E9: Click on cos Button
E10: Click on sin Button
E11: Click on tan Button
E12: Click on Inverse Button
I7
I8
I9
I10
I11
I12
I13
E13: Click on 1/x Button
E14: Click on x^2 Button
E15: Click on Standard Button
E16: Click on acos Button
E17: Click on asin Button
E18: Click on atan Button
E19: Click on Inverse Button
FAILURE
I14
I15
I16
I17
I18
I19
I20

Example of Test Case

```
public void testSequence11() throws Exception {
        InterfaceComparator.compare("I1");
        solo.clickOnButton("Scientific");
        InterfaceComparator.compare("I6");
        solo.clickOnButton("Inverse");
        InterfaceComparator.compare("I13");
        solo.enterText("Input", "dfghfdjg");
        solo.clickOnButton("1/x");
        InterfaceComparator.compare("I14");
}
```

- The Junit Java code of the test case corresponding to the execution of the event sequence E5-E12-E13 that detected the fault is shown.
- "`solo`" is one of the classes that Robotium provides for automatically firing events onto the application.
- "`InterfaceComparator`" is a class that we developed, having a method "`compare`" that is used to check the coincidence between interfaces.

Second injected fault



Discussion

- We are able to find a fault whose effects are visible or not visible on the GUI.
- However, the fault detection effectiveness of the technique depends considerably on the strategy used by the crawler for defining the input values needed for firing the events;
  - I.e. a possible fault in the reciprocal function due to an unmanaged exception of a division by zero might be revealed only by a test case with a zero value in the input field;
  - other input generation techniques should be considered in order to solve this problem.

Conclusions

- A technique for automatic testing of Android mobile applications has been proposed.
- The technique is inspired to other EDS testing techniques proposed in the literature and relies on a GUI crawler that is used to obtain test cases that reveal application faults like run-time crashes, or that can be used in regression testing.
- Test cases consist of event sequences that can be fired on the application user interface.

Future Works

- We plan to carry out an empirical validation of the technique by experiments involving several real world applications with larger size and complexity, with the aim of assessing its cost-effectiveness and scalability in a real testing context.
- In order to increase the effectiveness of the obtained test suites we intend to investigate further and more accurate techniques for the crawler to generate several kinds of input values, including both random and specific input values depending on the considered type of widget.
- Solutions for managing test case preconditions and postconditions related to persistent data sources (such as files, databases, Shared Preferences objects, remote data sources) will be looked for.

Thank you for your attention !!!