Two Way Lookahead of Software Testing in Android Application

Saryu Arora [1]* and Arti Rana[2]
1* Amity Institute of Information Technology, Amity University, India,
2 Amity Institute of Information Technology, Amity University, India,

Abstract **- Software testing is concerted. The emergence of multicore architectures and the escalationof bug-prone multithreaded software makes testing even more painstaking.The objective of this subjective study was to sightsee and comprehend that Software Testing which is a software embraces of two different activities, depending upon whether one is the developer of the software or the tester.[1]The study used qualitative grounded notion as its research method.Commendably conducting developer testing as well as tester testing requires both effective tool support by tools and abilities by concerned people. In this paper, I describe my experiences and lessons learned in teaching and training software testing techniques and tool support in both the areas.As software engineering one may just end uptreating themselves in roles as either a developer or a tester, they must learn the skills for both. Thus, wide-spread acceptance of Software Testing depends upon mechanisms that provide both functional and performance transparency.**
The practical implementation of the above theory is expressed with the help of Android Application's testingwith the automated approach of android applications and their requirement. All are the tools available for the same in case of white box as well as black box. Elaborative view on UI and GUI testing for the same. Different goals that necessitate different types and levels of testing at different stages in the process.

KEYWORDS : ***Software testing, (BBT) black box testing, (WBT) white box testing, validation, verification, android application, UI testing, GUI testing.***

I. INTRODUCTION

What is software testing?
Software Testing is the progression of executing a program or system with the intent of discoveryof errors. It involves any activity intended at assessing an attribute or competency of a program or system and shaping that it meets its requisite results [2]. Software is not unlike other physical processes where inputs are received and outputs are shaped. Most systems fail in fixed ways. By contrast, software can fail in many peculiar ways. Identifying all of the different failure modes for software is commonly infeasible [2].
Testing is a two-headed monster with two different characters: one for developers, and a different one which is for testers. Also, testing possesses a different nature if one is a developer in comparison to if one is a software tester [1] [2].There arespecific areas where the differences are especially unambiguous: The first area is that developers and testers can see different things: developers can see the code they write, while testers generally do not see the code (i.e., black box testing). Visibility of the code (i.e., white box testing) has benefits for evaluating coverage of the code written. The second area of difference is the attitude toward testing. For developers, the major aim is to finish writing with the code, and then to perform testing, which is basically an activity to show whether the code works. On the other hand, testers know that the code is buggy, and their job is to discover where the problem is. Because of this thing, developers have a tendency to test casually, often picking "test cases" that are most probable to work. [1] The third area of difference is their duties. This means that there are quality assurance-related abilities for both developers and testers, yet they are unalike.
In Section 1 I present why software testing is essential? What are those core reasons that make testing asignificant part of any software? This is because in spite of many limitations, testing is a major part in software development. It is largelypositioned in every phase in the software development life cycle. Typically, more than 50% percent of the development time is consumed in testing.
In section 2 I describe the two look areas of software testing i.e. as that of a developer and the other as that of a tester. Which covers testing methods and testing from the perspectives of both tester and developer.Where the black box testing is performed by testers and white box testing is performed by developers.
In section 3 I define the clear edge difference between validation and verification in terms of methods of software testing. As Verification and validation is the standard name given to checking processes which make sure that the software imitates to its specification and meets the needs of the customer. The system should be verified and validated at each phase of the software development process using documents created in earlier phases.
In section 4 I present the android application structure and its design. A typical Android app encompasses of top level and detail/edit views.
In section 5 I provide with the automated approach of android applications and why is it required? What all are the tools available for the same in case of white box as well as black box. How UI and GUI testing is done? As different objectives that require different types and levels of testing at different stages in the process. These needs command whether it makes logic to test manually or to automate the testing.
In section 6 I conclude my research with the finest understanding and implementation of the study.
In section 7 I gather and put all the references that would be used during the gathering of information and analyzing it.

1) Why do we need to perform Software testing?
Testing is additional than debugging a code. Be whatever the need of testing is, it should be taken into concern very seriously.An error is a human act that produces an inappropriate result. A fault is anindicator of an error in software. A fault, if come upon, may lead to a failure, which is a deviation of the software from its likely delivery or service. Imperfectness of humans causes errors. Testing classifies faults, whose deletion increases the software quality by increasing the software's latent reliability. Testing is the dimension of software quality.Software is written by people; people are imperfect and make mistakes. Thus, testing is desirable as it brings balance and perspective.
Software testing is important not only in a software but in any business. Most of us have had apractice with software that did not work as predictable. Failed software can lead to major effects over an organization.
a) Loss of money – this can compriseof losing customers right through to financial penalties for non-compliance to legal requirements,
b) Loss of time - This can be triggered by transactions taking a long time to process but can include staff not being able to work due to a fault or failure,
c) Damage to business reputation – if an organization is incapable to deliver service to their customers due to software problems then the customers will lose confidence or faith in the organization.

II. TESTING CAN BE CATEGORIZED INTO TWO MAIN DIVISIONS

Developer's testing i.e. white box testing - White Box Testing (WBT) is also referred to as Code-Based Testing or Structural Testing. It is the method in which internal structure is well-known to tester who is going to test the software [9].White box testing includes the testing by viewing at the internal structure of the code & when you are utterly aware of the internal structure of the code then you can run the test cases and verify whether the system meet necessitiesstated in the specification document [9]. Dependingupon resulting test cases the user exercised the test cases by giving the input to the system andinspection for expected outputs with actual output. In this is testing method user has to drive beyond the user interface to find the precision of the system.Stereotypically such method are used at Unit Testing of the code as Unit testing is done by the developer.For developer to test the software application undergoing test is like a white/transparent box where the inside of the box is clearly seen to the developer working as a tester (as he is aware of the internal code), so this method is called as White Box Testing.
The White-box testing is one of the finest method to catch the slips in the software application in early step of software development life cycle [9]. In this processdeveloping the test cases is most vital part.
Tester's testing i.e. black box testing - Black box testing is the Software testing mode which is used to test the software without knowing the internal arrangement of code or program. This testing method is what maximum of tester actual perform in the practical life.Fundamentally software under test is called as "Black-Box", we are considering this as black box and without inspecting the internal structure of software we test the software. While black box testing, the tester is just aware of the inputs and what are the expected outcomes of the software and they are not aware of how the software or applicationis internally processing to those inputs to give the desired outputs. The Tester only pass the valid as well as invalid inputs anddefines the correct expected outputs. The major reason for performing black box testing is to check whether the software is working as per expected in requirement document & whether it is meeting the user expectations or not.

2.1) the Testing Spectrum
Software testing is intricate in each stage of software life cycle, but the way of conducting the testing at each stage of software development life cycle is different in environment and it has different intentions.
Unit testing is a code centered testing which is performed by developers, this testing is primarily done to test each and individual units of codedistinctly. This testing is done for small units of code or generally no bigger than a class [4] [16].
Integration testingauthenticates that two or more units or other combinations work together appropriately, and inclines to focus on the interfaces specified in low-level design [4].
System testingdivulges that the system works endways in a production-like site to deliver the business functions stated in the advanced design [4].
Acceptance testing is led by business possessors, the reason behind acceptance testing is to test what the system does in fact, meet their business requirements [4].
Regression Testing is the testing of software postalterations; this testing is done to check that the consistency of each software release, testing after changes is been made to safeguard that changes did not invoke any new inaccuracies into the system [4].
Alpha Testingis performed generally in the presence of the developer at the developer's site.
Beta Testingis conducted at the customer's site with no developer on site.
Functional Testing is executed for a completed application; this testing is to validate that it delivers all of the behaviors essential of it.

2.1.1) Black box / functional testing techniques are as follows:
. Equivalence partitioning (EP) is a specification-based technique. It can be practical at any point of testing and is often a decent method to use first.
The idea behind this technique is to partition a set of test conditions into sets that can be considered the same by the system, hence 'equivalence partitioning or equivalence classes'.
In equivalence-partitioning technique we are required to test only one condition from each partition. This is since we are supposing that all the conditions in one partition will be smoked in the similarmanner by the software. If one

condition in a partition will work correctly then others will also work correctly. Similar in case of failure.

- Valid Input Class = Keeps all valid inputs.
- Invalid Input Class = Keeps all Invalid inputs.

. Boundary value analysis is executed by forming tests that exercise the edges of the input and output Classes identified in the specification. Test cases can be derived from the 'boundaries' of equivalence classes. Normally programming mistakes occur at the boundaries of equivalence classes are known as "Boundary Value Analysis (BVA)". Sometime programmers do not succeed in checking, special processing is mandatoryat boundaries of equivalence classes. A broad example is programmers may inadequately use <as a replacement for<=. The choices of boundary values contain above, below and on the boundary of the class [4].

. Decision tables are human understandable rules used to express the test experts or design expert's knowledge in a condensed form. Decision Tables can be used when the result or the logic involved in the program is created on a set of decisions and rules which need to be tracked. Decision table is comprised of four things a) condition stub, b) the condition entry, c) the action stub and finally d) action entry [4] [12].

. State Transition Diagrams (or) State Graphs is an exceptional tool to seizure certain types of system requirements and document internal system design. When a system need to think of what took place before or when valid and invalid orders of operation survives, then state transition testing might be used. State graphs are used when system transfers from one state to another state [12] [3].

2.1.2) white box / functional testing techniques are as follows:

.Statement coveragein this each node or statements are crisscrossed at least once, statement testing also known as node coverage. It is a simple metric to estimate, and there are number of open source products which are used to calculate this coverage. In due course, the main benefit of statement coverage is to find out with part of code has been tested.

. Branch coverage/ decision coveragein this a branch is the outcome of a decision. This method is a better one because it provides with a deep view as compared to statement coverage [11].

. Cyclomatic complexity of a method is one plus the number of unique decisions in the method. It helps you define the number of linearly liberated paths, also called as basis set. It is a software metri that provides a quantitative measure of logical complexity of a program [12] [4] [3].Cyclomatic complexity is simply known as program complexity, or in other words McCabe's complexity.

It is calculated in one of the three ways:

V(G) = E – N + 2 , where E is the number of edges and N is the number is nodes in the graph.

V(G) = P + 1 , where P is the number of predicate nodes.

V(G) = R , where number of region in the graph.

. Identification of Basis Path. For identification of basis path, independent path is required. An independent path is any path through the program that introduces at least once a new set of processing statements or a new condition.

. Loop testing

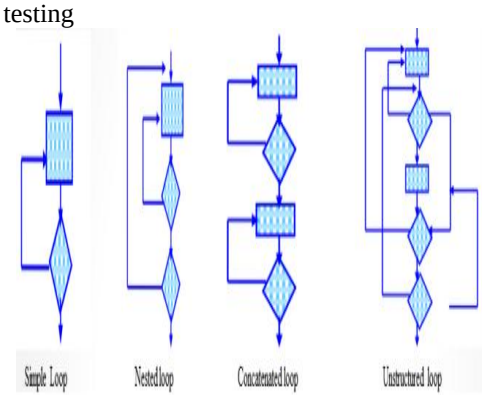


Figure-1[12]

- Simple loops – skip the entire loop. Make 1 pass through the loop. Make 2 passes through the loop. Make m number of passes through the loop, where, m<n, n is the maximum number of passes through the loop. Make n, n+1 passes through the loop [12].
- Nested loops – start at the innermost loop. Conduct simple loop test for the innermost loop. Work outwards, conducting test for the next loop keeping all the other loops at minimum. Continue until all the loops are tested [4].
- Concatenated loops – if the loops are independent, test them as simple loops or else test them as nested loops [12].
- Unstructured loops - to test them one needs to reconstruct their design [12].

III. VALIDATION AND VERIFICATION

Verification comprises checking that the programs toe the line to its specification.

Validation encompasses checking that the program as employed meets the expectations of the customer.

Validation: it says "Are we building the right product?"
Verification: it says "Are we building the product right?"

IV. ANDROID APP STRUCTURE

Android platform is comprised of 4 layers: Applications at the topmost, an Application Framework layer that delivers services to applications, e.g., providing data access, a Library/VM layer, and, at the bottommost, the Linux kernel.

Java and Microsoft's .NET rule. On the other hand, Java has the power, as it is No. 1 language in terms of developers. Java is the core of android mobile OS. Android delivers the tools and APIs obligatory to instigate developing applications for the Android OS using the Java

programming language. Android applications are stereotypically written in Java, possibly with some additional inherent code. The Java code is compiled to a .dex file, containing compressed bytecode. The bytecode runs in the DVM (Dalvik virtual machine), which in turn runs on top of a smartphonespecific version of the Linux kernel [13] [14]. Android applications are circulated as .apk files, which bundle the .dex code with a "manifest" (app specification) with .xml extension [5].

Android app workflow

A rich application framework easesAndroid app construction, as it offers a set of libraries, a high-level interface for interaction with low-level devices, etc. More essentially, for our purposes, the application framework arranges the workflow of an app, which makes it laidback to construct apps but rigid to reason about control flow [7] [5]. A typical Android applicationcomprises of distinct screens named Activities. An activity expresses a set of jobs that can be assembled together in terms of their actions and corresponds to a window in a conventional desktop GUI. An activity acts as a container for usual GUI elements such as toasts (pop-ups), text boxes, text view objects, list items, progress bars, check boxes. When interacting with an app, users traverse different activities using the aforesaid GUI elements [7]. Activities can serve different commitments. It provides a screen with which user can interact in order to do something. For e.g., in a classic news application, an activity home screen displays the list of present-day news; picking a news headline will trigger the switch to another activity that displays the full news item. Activities are regularly invoked from within the application, though some activities can be raised from outside the application if the host applicationpermits it.

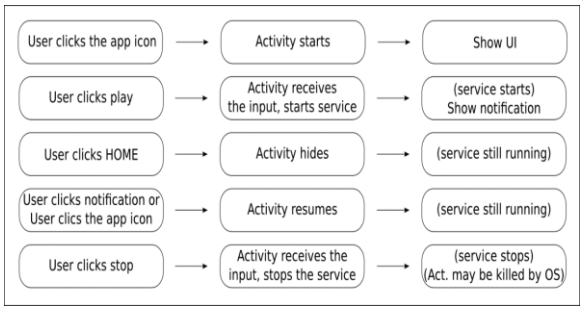


Figure-2[8]

In the above Figure-2 it is given an Android application for testing, the first step is to mechanically Detect Input/output Interfaces. An application's input interfaces signify the different ways in which it can be raised by either the execution environment or user.

V. AUTOMATED APPROACH FOR TESTING OF SOFTWARE

Why to automate?

The necessity for speed is basically the hymn of the information age. Because technology is now being used as a weapon on the priority list of customer interaction, delivery diaries are subject to market pressures. Late products can drop revenue, customers, and market share. But economic pressures also demand resource and cost cutbacks as well, leading many establishments toimplement automation to shrink time to market as well as cut testing budgets.

Some of the famous tools used for automating android applications are:

Robotium is undoubtedly the most standard framework for test automation of Android applications for now. It is a dominant tool for writing consistent automatic tests for nominal time. Tests are written in Java, and can also be written in other languages like Python. Tests cannot run on devices, they can only run on the emulator.

Monkeyrunner is counted in to Android SDK tools and provides API to control a device for functional testing of applications. It does not want the source code of an app and can be run both on the emulator as well as on the real device. Tests are either recorded or written with python.

Appthwack is an exciting service for testing on Android devices. The application can be loaded on the resource and can be installed on the device.Afterwards it is explored by running, taking measurements of memory and CPU usage, detecting errors and complications, having a minor monkey test. According to the survey Appthwack delivers a report with screenshots.

MonkeyTalk is a free tool with its own commanding script language and it also supports Javascript. Used for testing Android and iOS applications. With MonkeyTalk you can generate and store testing projects (test cases, test suites). It can be joined to Eclipse.

4.1) UI testing frameworkfor android (developer testing), the Android development environment offersan integrated testing framework centered on JUnit to test the applications [16]. At the moment, the framework has been typicallyplanned to carry out assertion based unit testing and fuzz (random) testing of activities [5]. Android testing is based on JUnit which is used for unit testing [5] [15].Unit tests state a developer that the code is doing things correct. Unit tests are written from a programmer's standpoint. They guarantee that a certain method of a class fruitfully performs a set of specific tasks. Each test confirms that a method outcomes with the expected output when provided with a known input. JUnit is very good at unit testing. In common, a JUnit test is a method whose statements test a part of the application in test. You establish test methods into classes called test cases (or test suites). Each test is a separate test of an individual module in the application under test. Each class contains related test methods [15].

It is relaxed to become astounded when you start scripting unit tests. The best way to start is to generate unit tests for new code. Start with new code, get used to the process, and then reexamine the present code to create a test suite for it. You ought to write unit tests formerly you write the code they will test. How can you pen down a test for something that doesn't exist? Learning this practice is 90%

conceptual and 10% practical. What is meant is that you merely pretend that the class you are writing the test for is present. Then write the test. The next step is to run your unit tests, fix the syntax errors i.e., implement the class with the interfaces just described by your test and run the tests for a second time. Repeat this process, whenever writing just adequate code to fix the failures. Run the tests till they pass. The code is "done" when all of the unit tests pass.

In all-purpose, there should be a unit test for every public method of your class. Put the unit tests in the same package as the related classes being tested. This type of union allows each unit test to call methods and reference variables that have access modifiers of package or protected in the class being tested. Evade using domain objects in unit tests. Domain objects are explicit to an application because very frequently the classes formed for a project apply to other projects. Reusing these classes may be direct. But if the tests for the reused classes use another project's domain objects, it can become a very time-consuming activity. In that case generally the test will either be dropped or amended.

Aggregating tests in suites
Using suite as a runner allows you to by hand build a suite comprising tests from many classes. To use it, annotate a class
with @RunWith(Suite.class) and @SuiteClasses(TestClass 1.class, ...). all the test cases in this class will be executed.
importorg.junit.runners.Suite;
@RunWith(Suite.class)

@Suite.SuiteClasses({
TestFeature...class,
TestFeature...class,})

Test execution order
By design, JUnit does not tell the implementation order of test method requests. Until now, the methods were basicallyraised in the order returned by the reflection API. JUnit will by default use a deterministic, but not expectable, order (MethodSorters.DEFAULT). if we want to change the test execution order just annotate your test class using @FixMethodOrder and specify one of the existingMethodSorters:
@FixMethodOrder(MethodSorters.NAME_ASCENDING)
: sorts and displays the names in lexicographic order.

Matchers and assertThat
A new assertion mechanism was built and the syntax was like this:
assertThat(a, is(5));
assertThat(b, is(not(10)));
Benefits of this assertion syntax is that it becomes more understandable and simple to type.

Ignoring tests
If for more or less reason, you don't want a test to fail, you just want it to be ignored, the syntax used is @Ignore("This Test is ignored")

How does JUnit work [4]?
Assertions
Use assert statements to assert that a thing is true (assertTrue(expected, actual), assertTrue(condition), etc), that a thing is false (assertFalse(condition), etc), that a thing is equal (assertEqual(expected, actual), etc). When an assert fails, the test failed for that specific case, henceforth the code needs to be fixed.The parameter order is expected value followed by actual value.
Version 4.x, JUnit:
@Test
Spot your test cases with the @Test annotation
For instance:
@Test
public void testAssertEquals() {
org.junit.Assert.assertEquals("…");}
@Test
public void testAssertFalse() {
org.junit.Assert.assertFalse("…");

Timeout for tests
Tests that are taking too long to run, can spontaneously fail. There are two options for prompting this behavior:
Timeout parameter on @Test Annotation. This only applies to test method. You can also specify timeout in milliseconds for a test method to fail if it takes lengthier than that number of milliseconds. If the time limit is exceeded, the failure is prompted by an Exception being thrown:
@Test(timeout=10000)
public void testWithTimeout() {
…….}
Timeout Rule. This applies to entire test class
The Timeout Rule relates the same timeout to all test methods in a class:
public class HasGlobalTimeout {
public static String log;
…

4.2) GUI testing for android- The widespreadpractice of GUIs for intermingling with software is moving towards the creation of more and more complex GUIs [6]. With the mountingdifficulty come encounters in testing the correctness of a GUI and its core software. Robotium is typically used to automate UI test cases and within uses run-time binding to Graphical User Interface (GUI) component.

V. ROBOTIUM FRAMEWORK

Robotium is an open source automation testing framework that is used to write a strong and dominant black box for Android applications [4]. The highlighting is generally on black box test cases. It completely supports testing for

natural and mixture applications. Natural apps are live on the device, i.e., designed for a specific platform and can be installed from the Google Play Store, howeverMix apps are partly built-in and partly web apps. It can also be installed from the App store, just the HTML is required in the browser, and the task is done [10] [4].
Robotium offers the following welfares:

- It test Android apps, both inherent as well as hybrid.
- It involvesnegligible knowledge of the application under test.
- The framework handgripsmany Android activities automatically.
- It needs minimal time to write solid test cases.
- With this, reading of test cases has improved to a great extent in comparison to standard instrumentation tests.
- Test cases are stronger due to the run-time binding to UI components.
- The execution of test cases is fast with this.

To develop constant and trustworthy tests, Robotium suggestsseveral methods that respond to various graphical elements in an Android app, which are as follows:

clickOnText("Login");

clickOnButton("Save");

searchText("Logout");

goBack();

getButton();

isRadioButtonChecked();

The following example shows a test containing the login process. The testLogin() method is fashioned to test the login process which contains the java code:

```
public void test_login(){
    solo.enterText(0,username);
    solo.enterText(1,password);
    solo.clickOnButton("Login");
     assertTrue(solo.searchText("Please wait while Logging in."));
    solo.waitForActivity("com.pointabout.mypersonal.Main TabActivity", 1000);
     solo.assertCurrentActivity("The activity has to be the Main Tab", "MainTabActivityUser");
    solo.sendKey(Solo.menu);
    solo.clickOnText("Logout");
    solo.waitForText("Are you sure you want to log out?");
    solo.clickOnButton("Logout");
     solo.waitForText("You have been logged out of mypersonal.");
```

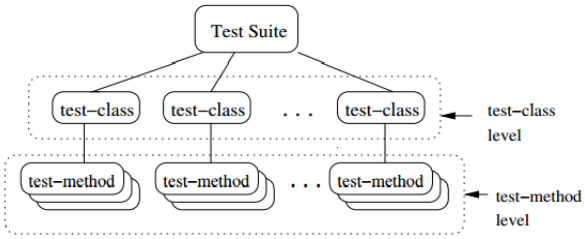
The solo object gives access to the enterText method which further needs two constraints that allow some text topass into a text input field in the app. The former value of the parameters signifies the IDs of the input text field on the login screen. The other parameter is the string that is to be put. The clickOnButton() method "clicks" the button for logging in the user.The waitForActivity()method waits for the user to login the application.Once the user is logged in,Robotium uses the assertCurrentActivity() method to check whether the main activity is shown or not. At the end of the test method, a JUnitassertTrueconfirmation is done to check whether or not the main activity. Also with the help of solo you can have access to the device buttons such as Menu, Home,sendKey().At last, the TearDown() method is called for closure of all the activities [10].

Executing tests- If we want to run Robotium tests on Eclipse, right-click the test class and select Run-As → Android JUnit Test. Robotium can also run over command line.

VI. CONCLUSION

In this research paper, we are trying to convey the importance of software testing, to what extend is it required in any application development. Along with that what all the types of software testing and who actually can perform this task. In which we elaborated on the Developer's testing as well as Tester's testing, which in technical words also known as Black box are testing and White box testing. Further we discussed about the soul of smartphones i.e., android application structure also how its testing is done from both perspectives, UI framework as well as GUI framework used for testing an android app. The JUnit framework motivates the coders to write down the test cases, so they can re-run those test cases whenever they modify their code in future. JUnit test cases are framed in java code, test class contains one or more than one test methods in it which further framed into test suite, as shown in

Figure 3 [17]



It presents the Robotium frameworkgrounded on JUnit,can be used to write the black box test cases to test the android app's functionality, system and acceptance level. By using Robotium,test cases outputs can be verified using JUnit assertions. Robotium is a GUI testing framework for Android applications that supports both black-box testing as well as white-box testing.

VII. REFERENCES

1. Teaching Software testing from two Viewpoints, Neil B. Harrison, Department of Computer Science, Utah Valley University, 800 West University Parkway Orem, Utah 84058801-863-7312
2. Software Testing 18-849b Dependable Embedded Systems Spring 1999, Authors: Jiantao Pan
3. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques, Mohd. Ehmer Khan, Department of Computer Science, Singhania University, Jhunjhunu, Rajasthan, India, Farmeena Khan, Department of Computer Science, EIILM University, Jorethang, Sikkim, India

4. Black box and white box testing techniques- a literature review, Srinivas Nidhra1 and Jagruthi Dondeti School of Computing, Blekinge Institute of Technology, Karlskrona, Sweden Jawaharlal Nehru Technological University, Hyderabad, Andhra Pradesh, India
5. A GUI Crawling-based technique for Android Mobile Application Testing DomenicoAmalfitano, Anna Rita Fasolino, PorfirioTramontana Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy
6. Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning Wontae Choi EECS Department University of California, Berkeley George Necula EECS Department University of California, Berkeley Koushik Sen EECS Department University of California, Berkeley
7. Research on Development of Android Applications Jianye Liu School of Information Yunnan University of Finance and Economics KunMing, China ljyxingyun@yahoo.com.cn Jiankun Yu School of Information Yunnan University of Finance and Economics KunMing, China yjk1102@163.com
8. Link mail. Google.com/images
9. Teaching and Training Developer-Testing Techniques and Tool Support Tao Xie1 Jonathan de Halleux2 Nikolai Tillmann2 Wolfram Schulte21North Carolina State University, 2Microsoft Research
10. AndroidRR Logan Donovan - lrd2127@columbia.edu, DmitriyGromovdg2720@columbia.edu, Riley Spahn-riley@cs.columbia.edu, DeepikaTunikoju - dt2417@columbia.edu
11. Automatic detection of infeasible paths in software testing D. Gong1 X. Yao1, 21School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu 221116, People's Republic of China 2College of Science, China University of Mining and Technology, Xuzhou, Jiangsu 221116, People's Republic of China E-mail: yxjcumt@126.com
12. Software Testing Methods and Techniques Jovanović, Irena
13. A Study of Android Application Security William Enck, Damien Octeau, Patrick McDaniel, and SwaratChaudhuri Systems and Internet Infrastructure Security Laboratory Department of Computer Science and Engineering The Pennsylvania State University {enck, octeau, mcdaniel, swarat}@cse.psu.edu
14. Implementing Security on Android Application 1, Kirandeep, 2, AnuGarg 1, School Of engineering and Science, Lovely Professional University 2, G.T. Road, Near Chehru Railway Bridge, Phagwara (Punjab)-144401, India
15. Enhancing Component Based Testing Using JUnit Tool in Net Beans Environment Ravinder Kumar Mr. Karambir Singh Student: CSE Department Asst. Proff. CSE Department U.I.E.T. kurukshetra university U.I.E.T. Kurukshetra University Kurukshetra, India Kurukshetra, India
[16] A Simple and Practical Approach to Unit Testing: The JML and JUnit Way YoonsikCheon and Gary T. Leaven
16. Empirical Studies of Test Case Prioritization in a JUnit Testing Environment Hyunsook Do University of Nebraska - Lincoln, dohy@cse.unl.edu Gregg Rothermel University of Nebraska-Lincoln, grother@cse.unl.edu Alex Kinneer University of Nebraska - Lincoln, akinneer@cse.unl.edu