Mobile Software Testing – Automated Test Case Design Strategies

Selvam R

Research Scholar, Department of Computer Applications, Bharath University,
Selaiyur, Chennai - 600 073 Tamil Nadu, India
Associate Professor, Department of Computer Applications
Arignar Anna Institute of Management Studies & Computer Applications, University of Madras
Pennalur, Sriperumbudur-602 105, Tamilnadu, India

Dr. V. Karthikeyani

Professor
Government Arts College for Women
Salem-Tamilnadu, India

Abstract - Mobile devices are poised to challenge PCs as the application platform of choice, with 500 million mobile internet devices expected to ship in 2012 compared to 150 million PCs. The convergence of all digital devices into mobile platform model augments the software companies, software developer, and venture capitalist firms to turn their focus into mobile application platform (for example mobile social networking application like face book and mobile VOIP like Skype) a futuristic platform for increased revenue, new challenges and growth potential. But the commercial success of these applications depends on their working smoothly and securely on a wide variety of handheld devices and wireless networks. More and more virtual mobile application stores are built on the web. The web itself is in the transforming form to adapt to the mobile devices to thrive on. The sudden growth in the mobile application and the complexity in the divergence of the devices that uses these applications present increased challenges and opportunities for the software testing companies and software testers to conquer this small device. Performing such testing quickly and cost-effectively greatly expands the market for such applications. This paper deals the nuances of Automated Test Case Design Strategies for Mobile Software Testing.

Keywords: Mobile, Software Testing, Automated Test Case, Wireless networks

I. INTRODUCTION

Mobile application or shortly called as Mobile apps is a term used to describe the applications that have been developed for small low-powered handheld devices such as PDA, mobile phones etc.,. These applications are either pre-installed on phones during manufacture or downloaded from the web mobile app store or through other mobile software distribution platform.

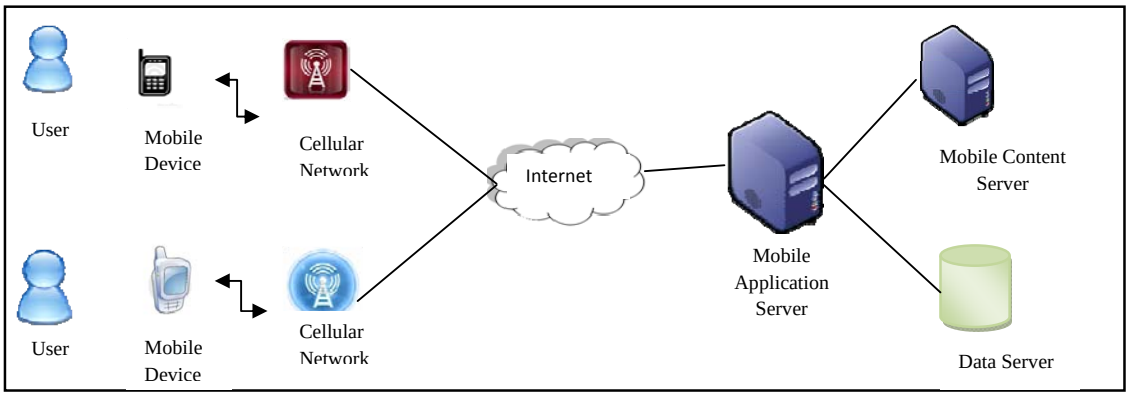

Fig 1. Mobile Web Application Interaction Scenario

Fig 1. depicts the Mobile Web Application Server Interaction with various mobile devices through the wireless cellular network.

A typical mobile system varies in terms of devices, user functionalities and the accessibility and operability in various locations and the way it can connect with servers by various methods. The mobile system typically includes mobile devices, wired and wireless connectivity, backend server and underlying Network Infrastructure.

The Mobile applications that run on this device may be a **stand-alone** application which depends solely on the device hardware capabilities E.g. Software games, calendar and address book etc. or the **application may run remotely at server** and the device acts as a platform for viewing and interacting E.g. the Mobile TV, E-mail, Mobile Browser based applications etc. The mobile applications can **run on either device or server**, depending on architecture and device capability and the nature of the mobile apps. Since the core business applications needs a more sophisticated environment to run increasingly these apps are providing mobile application interfaces in mobile devices to access the features of these typical application by the mobile users.

The mobile devices nowadays available to the end user has been built with more powerful processor, storage, software capabilities, wide array of connectivity features, multi-media capabilities with powerful audio/video capabilities, sensors, accelerometers and GPS facilities.

This increased amalgamation makes **1-G (1st Generation) analog mobile phones** which have been only used for voice applications turns into a **most powerful multi-function device 3-G (3rd Generation) & 4-G (4th Generation) smart phones** of the present and future. This smartness of this mobile device have been tamed and complimented with suitable mobile applications to make the most out of it otherwise the smart hardware facilities present in the mobile phones will be rendered useless. In a typical scenario, the camera that is inbuilt with mobile phone is not only merely help the user to capture the image or record the video that he desires, it also help him to process the image or video simultaneously like face identification etc.

The mobile application and the mobile devices available in the market have taken upbeat and fulfill the idea of ubiquitous computing. The wide variety of mobile applications that catches the human imagination is developed by the software developers worldwide. These applications are distributed through several platforms like Apple Mobile app store, Nokia Ovi, Android Market and other third party servers and by service providers in different countries.

The application that has been developed for one platform may or may not run on all devices that have been sold by hardware manufacturers. Even on the devices that use the same platform also differs in device capabilities which may make the mobile application rendered unsuitable.

The mobile applications are developed in quick succession with little or no software tests and distributed without thorough testing because the mobile application developer concentrates wholly on his software functionalities rather than the device complexities and most of the applications are being tested on emulators rather than on real mobile devices in most cases. The bugs present in the mobile applications may endanger not only the application that it present and may affect the device in which it runs. The need for mobile application testing assures prime importance for the successfulness of any developer, enterprise to make footprints into the mobile software arena.

II. THE CHALLENGES IN TESTING MOBILE APPLICATIONS

The Key points that are to be considered while testing mobile applications are to be addressed with the following questions:

- Which mobile devices and operating system versions will this application support?
- How do we test applications to make sure they run on those platforms?
- What modifications must be made to accommodate the differences among platforms?
- How will industry innovations be supported going forward, since new mobile devices, technologies, and applications are constantly being introduced?
- How will development and testing processes accommodate the inherent differences among wireless network protocols and mobile service providers?
- How do we know how much testing is enough?

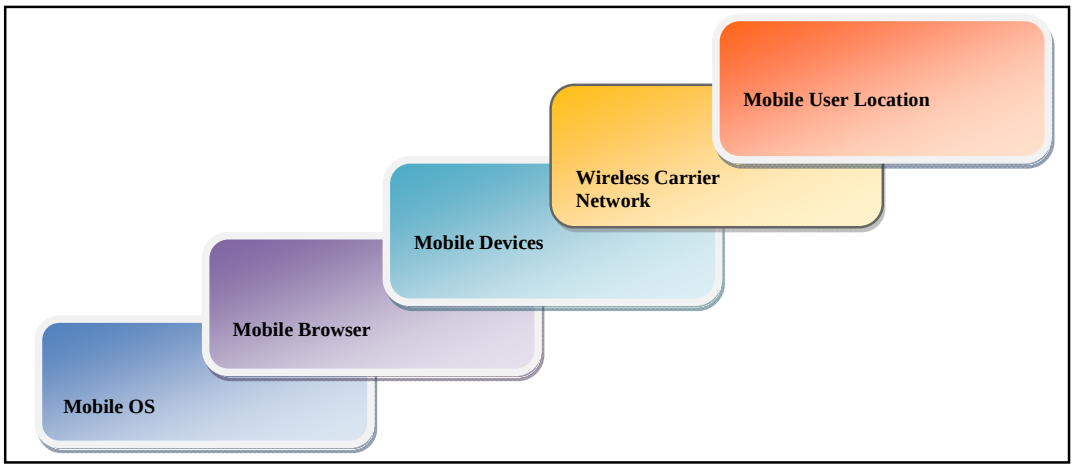

Fig. 2. Mobile Application Testing Complexity Matrix

As illustrated in fig. 2 the mobile application testing matrix is becoming more and more complex scenario with the addition of each factor whose multitude changes over the period of time. The complexity arise from each of these factor has been discussed in the following paragraphs.

A. Hardware Diversity and Complexity Challenges

In the PC test environment world, testers have essentially only one central processing unit platform (x86-compatible microprocessors) on which they must test applications. Most of the other hardware components that go into a PC or Mac, such as the disk drives, graphics processor and network adapters are usually thoroughly tested for compatibility with those operating systems and pose a relatively minor risk of problems. Their display formats also fall within a relatively narrow range of choices, and the input devices (mostly keyboards and mice) are well-known familiar.

But mobile voice and data service carriers differentiate themselves by offering a wide range of handsets, each with unique configurations and form factors that can have unpredictable effects on the performance, security and usability of applications. Various handsets are built around a wide variety of processors, running at various speeds with widely varying amounts of memory, as well as screens of different sizes operating at different resolutions and in different orientations (landscape, portrait or both.)

Today's mobile handsets also contain a greater, and a more rapidly-changing, variety of hardware than the typical PC. These components may include Wi-Fi and Bluetooth network capabilities (along with, of course, cellular connectivity), an FM radio, a camera and in more and more cases a **GPS receiver** and even an **accelerometer**, which senses the movement of the device to reorient the display from portrait to landscape. Many handheld devices rely on **multiple digital signal processors** (one to handle voice communications, the other to process the audio, video and images associated with applications), as well as multiple input devices, such as a touch-screen and a keypad. Each combination of components interacts in different ways with each other, and with the operating system, to create potential compatibility and performance issues that must be addressed in testing.

B. Software Platform Diversity and Software Complexity Challenges

In addition to these hardware-based concerns, the tester must cope with the complexity of the software environment. To assure an application will work on most customers' PCs, a tester need only test it on the most popular current versions of the Windows, Apple Macintosh and Linux operating systems. To assure performance on the same range of mobile devices, a tester must address all current versions of the iPhone, Windows Mobile 7, Symbian, Android, iPhone and RIM Blackberry OSes, as well as the MeeGo platform developed by Nokia.

The handheld OS market is not only more splintered than that for PCs, but is also changing much more quickly. For example, as little as five years ago Palm was one of the dominant operating systems for mobile devices, but its influence is rapidly fading in favor of, for example, the iPhone OS. Such changes require that testers maintain knowledge of the test tools required for an ever changing cast of operating systems.

Such complexity continues throughout the entire mobile software stack. Testers must also build and execute scripts checking the interaction among the various applications on the handset, as well as between the application and components such as the camera, microphone, charger, Bluetooth, Wi-Fi and cellular radios.

Applications must be tested for their compatibility with any of the networks on which any given device might run. The networks operated by different carriers provide various levels of bandwidth, sometimes even within the same session as the user moves among cells or coverage areas. Different carriers use different methods to tunnel their own traffic into the TCP IP protocol used by the Web, changing how applications receive, transmit and receive data. They also use different Web proxies to determine which Web sites their users can access, and how those sites will be displayed on their devices. All of these differences can affect the stability performance or security of a mobile application, and must be tested to assure the end-user experience.

Just as mobile operating systems are constantly changing, so are the networks, protocols and other key elements of the infrastructures used by the network providers. Carriers worldwide are upgrading their networks from 2-G to 3-G, and even to 4-G with LTE (Long Term Evolution) networks.

Finally, in order to be certified by carriers or handset manufacturers (and thus gain access to wider markets) applications may have to be tested for compliance with industry or carrier-specific standards. Each of these added test requirements increases the complexity, cost and time required assuring proper mobile application performance.

C. Testing Mobile Application on Emulators

In an ideal world, all mobile application testing have to be done on the target mobile device so that every possible interaction among its hardware and software elements, as well as with the wireless carrier's network, could be tested in the most accurate and reliable environment. However, acquiring every possible target device and performing manual testing on it is too complex, costly and time-consuming to be feasible during every stage of testing. **Device emulators** – software that simulates the Performance and behavior of the physical device – are far easier to obtain and less expensive than samples of the physical devices. While they can be less accurate test platforms than the actual hardware, they can be a cost-effective complement to testing on the physical device when used appropriately.

Emulators for a wide range of popular handsets are available on the Web, ranging from low-cost freeware to more expensive commercial applications that offer higher levels of formal support. Emulators can be used to test Web applications using the software development kit for a browser or by packaging the application as a .jar, .apk or .sis (platform specific) file, installing the application on the emulated device and testing the application.

Web servers and Web pages meant to be accessed from browser plug-ins, by loading information about the handset such as user agent details, headers and handset/vendor ID into an .xml file. Using XHTML and WML add-ons to the browser, testers can verify if the Web pages display correctly on the device browsers.

Because of the emphasis on time-to-market, many mobile applications are developed using RAD (rapid application development) in which multiple versions of the software are quickly developed, assessed by end users, and tweaked accordingly. This rapid-fire cycle of coding and re-coding makes it almost impossible to assess how each change affects the application's performance, stability or security.

III. MOBILE APPLICATIONS TESTING – A BIG PICTURE

The key to the successful implementation of Mobile Application delves around the successful test results obtained from mobile software testing. The Mobile Application testing is more complex than the desktop or web applications which is inherent in nature due to the mobility nature of the device when compare to PCs. The ever increasing potential of the mobile devices touted to be the next generation mobile computers attracts a greater emphasize on the testing in the software development life cycle and calls for innovation in the testing methodologies and techniques. The Mobile Application Test Matrix has been discussed and assumed to be greatly influenced by the Testing Environment, Levels of Testing, Testing Techniques and Scope of the Testing.

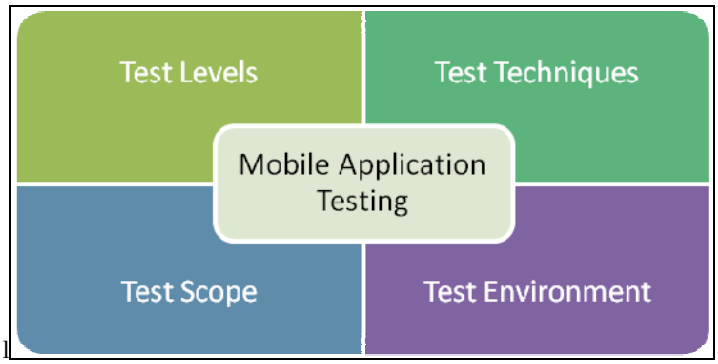

Fig.3. Mobile Application Test Matrix

As depicted in Fig 3. Automated Test Case Design for Mobile Application revolves around the four factors shown in the Test Matrix. The details of each of the factors are discussed below:

A. Test Environment

The Test Environment for Mobile Application is always been a debatable issue whether testing on Real Device or testing on emulator is sufficient to derive the test outcome and ascertain the test results.

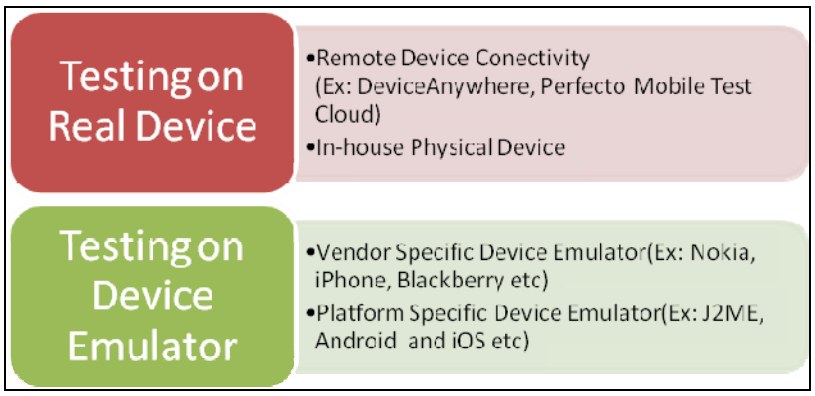

Fig.4. Mobile Application Test Environment

Fig.4 illustrates the Mobile Application Test Environment.Testing on emulators is considered to be the cost effective solution and can be very useful to cover a wide range of devices (with different screen resolutions for example) to which we may not have physical access. But a tester must bear in mind that all activities cannot be realistically emulated like taking a picture or video. Performance factors cannot be evaluated in real scenario. Because of these and other more subtle differences a tester must not assume that just because an application that works perfectly on an emulator will also run perfectly on a real device. For testing mobile applications there are two categories of emulators are available – a vendor specific and platform specific. The vendor specific emulators are provided for the purpose of testing the mobile application compatibility in their hardware profiles whereas the Platform specific emulators provided for general purpose mobile application testing which is device agnostic and runs a similar test in the wide range of device having similar Application platform. Nokia, Blackberry, iPhone and other vendors have released their own Device Emulators. Google Android, Apple iOS and J2ME by SUN Microsystems which is now owned by Oracle are some of the Platform specific Device Emulators.

Testing on Real Devices is considered to be the optimized solution in the real world scenario but considered to be costly solution. The wise decision has to be arrived based on the nature of the application. If the application runs only based on the device capabilities and interfaces and the testing features are supported by the emulator then the application need not be tested on real devices. On the contrary if the application is based on geo-location capabilities of the mobile devices and user presence in various networks then it has to be tested on real devices. In a real test environment it is not possible to test the application on multitude of the devices offered by various vendors and the complexity still increases if it has to be tested on various wireless networks since the restrictions exercised by one network to the other network operator varies. To the mobile software testers, be it

individual tester or by testing companies, assistance are rendered by the **Perfecto mobile cloud Automation and DeviceAnywhere** for Testing on Real Devices. They have hosted a numerous combination of devices and carrier location to obtain the accurate test results based on subscription services. The tester can build and run test cases from his PC itself and test the application which he desires to be implemented on various mobile platform and multitude of devices.

The best solution is initial phase testing on emulator and end testing on the real devices.

B. Test Levels

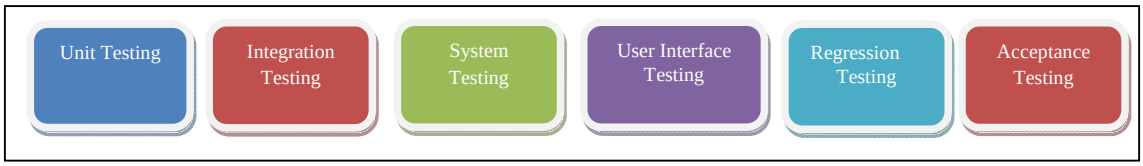

Fig.5. Mobile Application Test Levels

Fig.5 depicts the various tests carried out at different levels and discussed in the following paragraphs.

Unit Testing

A unit is smallest testable piece of software that can be compiled, linked, loaded for example functions/procedures, classes, and interfaces which is normally done by programmer. The Test cases are written after coding. The purpose of this test is to find (and remove) as many errors in the mobile software. Unit testing is also referred as Component Testing.

Integration Testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). As in the case of Android platform, the integration is between one Activity and other Activity.

System Testing

System testing tests a completely integrated system to verify that it meets its requirements. In the System Testing environment the Mobile Application will be tested on a whole that it meets all the specification specified by the application.

User Interface Testing

The User Interface Testing represents the ways we can interact and control the device. The User Interface is governed by

- Built-in QWERTY keyboard/keypad / Soft Keyboard / Hard Keys
- Touch screens
- Screen Orientation and Resolution
- Peripherals that you can plug in to the device
- Trackballs, Track wheels and Touch Pads
- Shortcuts to the application on the Home Screen
- Graphical User Interface position and size like Buttons, Radio Button and Menus etc.

Regression Testing

Regression testing is the (re-)execution of a fixed subset of the acceptance Test cases after something has changed. The goal is to manage the risk that a change fixed a known problem or didn't cause a previously working area of the application to break. In this context, "change" can take several forms, from such things as developer-supplied bug fixes to Mobile operating system updates. The art in regression testing is to design test cases that target those changes or focus on the localized dependencies of the change.

Acceptance Testing

Acceptance testing refers to the execution of a (large) set of test cases on the target device itself in a lab setting. It is intended to verify that a particular product or application meets the requirement specifications of the mobile application under test. The operational integrity and performance of an application that has passed acceptance

testing on a reference platform is often used as a benchmark against which all other versions of the application on all other platforms can be measured. This can be ensured by the user who run or test the product.

C. Test Scope

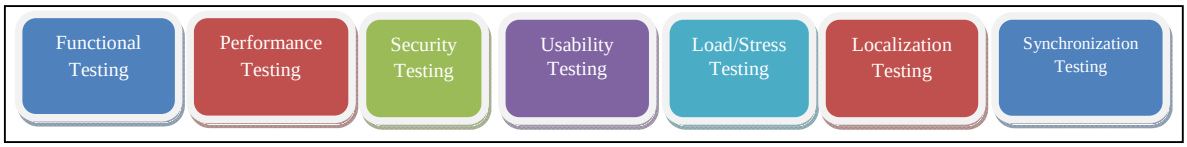

Fig.6. Mobile Application Test Scope

Fig 6. depicts the scope of Mobile Application Testing which has been based on the test levels.

Functional Testing
The most essential testing procedure is to verify that the basic functionality of the mobile application. The functions performed by the application may be device specific. Such functional testing should occur when the application is in the early development phase.

The function testing differs with Handset Maker/Model, Operating System, Browser, Wireless Carrier, Location and Language.

Performance Testing
Performance Testing is another critical step in the successful launch of a mobile application. The key factor to be aware of here is mobile carriers, which can affect the speed and ease-of-use of the application. If the application performance is poor for the end user, they will be likely to divert to other apps and services. The application must therefore be tested with different devices and carriers depending on the county and region.

Since your application is going to be used on devices in various locations with various network connection speeds, it is important to plan testing coverage for the following scenarios:

- Only Wi-Fi connection
- Only 3G connection / Only 2G Connection
- With no SIM card in the device
- In Airplane mode (or all connections disabled)
- Using the network through a USB connection to a PC

Security Testing

The threat of passwords or usernames being left unencrypted on the device is a real one that can be taken advantage of by other users. During installation of your application on a mobile device you are likely to encounter options for application execution permissions. For example, on Windows Mobile devices there are three levels: Privileged, Normal, and Blocked, which are specified by the application's certificate, and define how much access the application will have to the device itself. On Blackberry, you may be asked to allow, prompt for, or deny your application access to various connection types, interactions, or user data on the device. Outside communication of any sensitive data should be done over SSL/TLS because it can't be assumed that the user will configure their Wi-Fi to be secure. Even after installation, it is possible to change the security settings of our device or application permissions. Tester should include some test cases to cover these scenarios as well.

Usability Testing

Usability testing plays an important role for launching a quality mobile application. Usability testing should occur as soon as the basics table for the application is ready. A user may have a good experience on the Nokia, while another might have bad experience on an on –touch screen device. In this instance, changes to the UI should be considered to improve the experience on the non-touch screen device. Needless to say, usability testing is an extremely critical step towards launching a high-quality mobile application.

Load/Stress Testing

Although mobile devices' memory and power have improved over the past few years, mobile applications still have much more memory and CPU constraints than desktop applications. With users expecting applications to hold up even under high traffic conditions with many applications running, stress testing is a must to find

exceptions, hangs, and deadlocks that may go unnoticed during functional and user interface testing. Performing the same operations over and over again, particularly those that load large amount of data repeatedly. Performing the repeated operations at varying speeds – very quickly or very slowly to stress test the Mobile Application.

Localization Testing

Localization testing is the part of the software testing process focused on the language and internationalization of the application. Localizing an application requires a basic understanding of the character sets typically used in modern software development, as well as an understanding of the issues associated with them. Localization includes translating of the application user interface and adapting graphics for a specific culture/locale. The process can also include translating any help content associated with the application.

Synchronization Testing

Synchronization is a feature that enables exchanges, transforms and synchronizes data between two different applications or data stores. Synchronization could be either cradle-based or wireless. The SyncML Consortium is on a mission to get mobile application developers and handheld device makers to use a common, XML-based data synchronization technology. This category lists the different failure that could be encountered while synchronizing data between two applications or between two devices like PC and Mobile.

D. Test Techniques

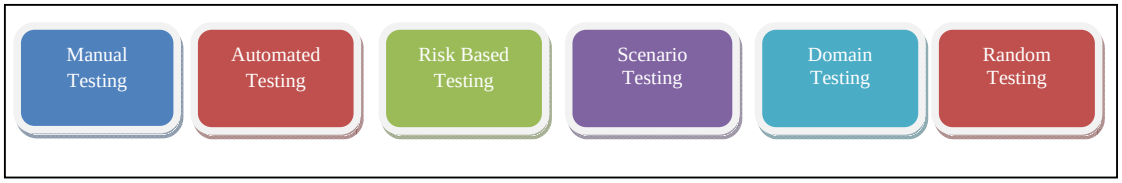

Fig.7. Mobile Application Test Techniques

Fig.7 illustrates the various testing techniques that can be adopted for carrying out the planned tests.

Manual Testing

To date, most mobile software functional testing has been done manually. Testers, typically following a script of step by step instructions, interact with the device as a user would, visually verify results. In situations where use cases have been thoroughly documented and where the tests are executed by a well-managed and motivated staff, this manual functional test method can produce sufficient results. However, the process is invariably tedious for the testers and, because of the human element, error prone.

Automated Testing

Test automation is the use of software to control the execution of tests, compare actual outcomes to predicted outcomes, set up test preconditions, and to automate test control and test reporting functions. Automated testing is used in those situations where extensive manual test execution is impossible or expensive. Reliability and stability tests often fall into this category. What to automate, when to automate, or even whether one really needs automation are crucial decisions which the testing team must make. Automated testing offers fast, repeatable and comprehensive test execution, including overnight or over-weekend test runs; however, it requires significant up-front investment in tools, test-script development and data collection, and is not amenable to fast-changing requirements.

Risk Based Testing

Imagine how programs may fail and test for them. Risk-based testing is employed to test high-probability failure points identified in the application. Mobile Switching Center failures and software upgrade errors may occur in a mobile application execution environment.

Scenario Testing

User Scenario or Use Case Testing is a combination of tests that will be used in real life. Have multiple applications running on our device so we can switch between our application and other device applications often and at different states within our application:

- On Android or Blackberry we can press and hold the hard Home or Menu key, respectively, to quickly switch between running applications.
- Using all means to launch our application, for example on Windows Mobile using the Touch Flo, the file explorer, and if the application is already launched, maximizing and minimizing the application.

Domain Testing
Domain testing is focused on variables such as inputs, outputs and other variables. With domain testing the tester tries ranges and options and subdivides the test world into classes. Performed predominantly when testing input parameters, the idea is to find the highest probability errors with a relatively small set of tests.

Random Testing
Random automated testing (or monkey testing) continuously performs high-volume testing with new and potentially random cases. The idea is to have an automated test system which creates, executes and evaluates a huge numbers of tests. Randomly sending screen taps and keystrokes to our application.

IV. AUTOMATED TEST CASE DEVELOPMENT

Automated test case has been developed to carryout Unit Testing in various platforms and the sample test code has been given for different platforms are listed under the following topics. Even though the test cases has been developed under various platform the Testing template remains common for most of the platform. Each test case covers a defined Test Class which extends the parent TestCase class of the testing tool, The test case class constructor is used by the testing framework when we run the test. It calls the super constructor with parameters that tell the framework which mobile application should be tested. We use setUp() to initialize variables and prepare the test environment and a test method which defines the testing activity. A set of assert methods to obtain and display the test result. Messages are only displayed when an assert fails.

A. Automated Unit Test Case Development for Android platform

Android offers a powerful but easy-to-use testing framework that is well integrated with the SDK tools. Because writing tests is an important part of any development effort. Android test suites are based on Junit. The following test case has been developed and tested in Eclipse IDE. A useful general test case class with Android testing, is Android TestCase. It extends both TestCase and Assert. It provides the JUnit-standard setUp() and tearDown() methods, as well as well as all of JUnit's Assert methods. In addition, it provides methods for testing permissions, and a method that guards against memory leaks by clearing out certain class references.

```
package com.utest.helloandroid.test;

import com.utest.helloandroid.HelloAndroid;
import android.test.ActivityInstrumentationTestCase2;
import android.widget.TextView;

public class HelloAndroidTest extends ActivityInstrumentationTestCase2<HelloAndroid> {
   private HelloAndroid mActivity;  // the activity under test
   private TextView mView;          // the activity's TextView (the only view)
   private String resourceString;

   public HelloAndroidTest() {
    super("com.utest.helloandroid", HelloAndroid.class);
   }
   @Override
   protected void setUp() throws Exception {
      super.setUp();
      mActivity = this.getActivity();
      mView = (TextView) mActivity.findViewById(com.example.helloandroid.R.id.textview);
      resourceString = mActivity.getString(com.example.helloandroid.R.string.hello);
   }
   public void testPreconditions() {
    assertNotNull(mView);
   }
```

```
    public void testText() {
      assertEquals(resourceString,(String)mView.getText());
    }
}
```

B. Automated Unit Test Case Development for J2ME platform

One of the unit-testing tools most widely used by Java developers is the JUnit framework. A similar framework for J2ME unit testing is J2MEUnit. Here is a sample test:

```
import j2meunit.framework.*;
public class TestLoginWindow extends TestCase {
  public TestLoginWindow() {
    super("null");
  }
  public TestLoginWindow(String s) {
    super(s);
  }
  protected void runTest() throws java.lang.Throwable {
    if(getTestMethodName().equals("testLoginWindow"))
      testLogindWindow();
    }
    public Test suite() {
      return new TestSuite(new TestLoginWindow().getClass(),
        new String[] {"testLoginWindow"});
    }
    public void testLoginWindow(){
      // test it
      // use assert(String, boolean)
    }
  }
}
```

When using J2MEUnit in our testing, we need to:

- Create a subclass of TestCase, like TestLoginWindow in the example.
- Override the method runTest() as in the example. Because J2MEUnit doesn't use reflection, when weoverride runTest() you must call getTestMethodName() to check the name of the test method (testLoginWindow() in the example above).
- Override the method suite() so that it returns a TestSuite object containing the name of the class for the test case and the names of the test methods.
- To check a value, we have to all the assert() method and pass a boolean that is true if the test succeeds.

C. Automated Unit Test Case Development for iPhone

```
*Test.h
import <Foundation/Foundation.h>
#import <SenTestingKit/SenTestingKit.h>
@interface SUnitTest : SenTestCase {
NSMutableArray* _array;
}
@end

*Test.m
#import "SUnitTest.h"
@implementation MyUnitTest
- (void) setUp {
_array = [[NSMutableArray arrayWithObjects:@"1", @"2", @"3", nil] retain];
}
- (void) testSize {
STAssertEquals(3, (int)[_array count], @"Size must be three.");
```

```
}
- (void) testIndex {
STAssertEqualObjects(@"2", [_array objectAtIndex:1], @"Must retrieve object at index of 1.");
}
- (void) tearDown {
[_array release];
}
@end
```

D. Automated Unit Test Case Development for Windows Phone 7 platform

To develop test cases for Windows phone 7, we need the Silverlight Unit Test Framework tweaked for Windows Phone 7 Let us create a SampleTestProject. We have to add references to the two Silverlight Unit Test Framework DLLs.

Microsoft.Silverlight.Testing.dll
Microsoft.VisualStudio.QualityTools.UnitTesting.Silverlight.dll

Because of recent changes in WP7, we can no longer set the root visual in the App.xaml.cs like normal. Instead, we'll setup the test page in the Loaded() event of your MainPage.xaml.cs in our test project. The following code hides the SystemTray on the phone (the bar at the top of the screen) which otherwise would cover up the top of test results screen, and hooks up the back button to allow you to return back from drilling into test results.

```
using Microsoft.Phone.Shell;
using Microsoft.Silverlight.Testing;
void MainPage_Loaded(object sender, RoutedEventArgs e)
{
   SystemTray.IsVisible = false;

   var testPage = UnitTestSystem.CreateTestPage() as IMobileTestPage;
   BackKeyPress += (x, xe) => xe.Cancel = testPage.NavigateBack();
   (Application.Current.RootVisual as PhoneApplicationFrame).Content = testPage;
}
```

Now that we've setup the test page, we can start creating test classes and test methods. Here is a sample set of simple tests:

```
[TestClass]
public class BasicTests : SilverlightTest
{
   [TestMethod]
   public void AlwaysPass()
   {
      Assert.IsTrue(true, "method intended to always pass");
   }

   [TestMethod]
   [Description("This test always fails intentionally")]
   public void AlwaysFail()
   {
      Assert.IsFalse(true, "method intended to always fail");
   }
}
```

V. RESULTS AND DISCUSSION

The strategy for development of automated test cases for testing Mobile Application need to use a combination of testing tools and techniques to meet our mobile application quality requirement. Emulator is a good choice for initial testing phases and depending on the inherent nature of the application category whether it is device -centric or data-centric, the test case has to be developed. Real Remote Device Testing environment is needed in the case of mobile web application to test the interoperability between various networks and devices. In the extreme case we have to opt for performing some manual testing with real devices on real networks. Unit testing and Functional Testing itself covers 60 to 70% of the successfulness of installation of mobile application. The designing of test case has to be taken into consideration of various testing strategies illustrated in this paper. The test case hat be focused and categorized based on the Device specific features, Network Issues, Mobile Web Browser (in case of Mobile Web Application), Mobile Operating System specific features (UI) and Mobile User Location (for Location based applications and localization features).

CONCLUSION

Application development for mobile devices is a fast growing phenomenon. According to eWeek.com there were more than 550million mobile Internet users in 2010, and the number is expected to surpass 1 billion by the end of 2013. All these users of mobile devices, like iPhone, iPad, Blackberry, Android, the upcoming Windows Phone 7, or even eReaders like the Amazon Kindle, create a never-ending demand for more and more unique and useful mobile applications. The arena for these new applications is vast from social networking application, where 55% of mobile internet users mostly use their devices social networking and emailing. Users want mobile applications to be simple and fast. Just one nagging bug or usability issue can spoil the entire experience. Vendors simply can't afford to go to market with an application that might have a critical bug or usability issue. Yet surprisingly, there is no previously existing comprehensive guide on how to test the particular complexities of mobile device applications. The strategies presented here are intended to highlight some of the areas where the testing of mobile device applications differs from testing desktop or regular web applications. It is important to plan a test strategy that is mobile-specific, or we may overlook crucial areas of testing like how network connectivity (or lack thereof) affects our application, how screen resolution and orientation changes can spoil a user's whole experience, and whether our application fulfils what users of a particular device have come to expect.

REFERENCES

[1] A Guide to Emulators – http://mobiforge.com
[2] Android Developer Site - http:/developer.android com
[3] IEEE. "IEEE Standard for Software Test Documentation," IEE Std 829. 2000.
[4] Lee, V., Schneider, H., & Schell, R. Mobile applications: Architecture, design and development. Indianapolis, Indiana, USA: Prentice Hall Professional Technical Reference, 2004.
[5] Malloy, A. D., Varshney, U., & Snow, A. P. Supporting mobile commerce applications using dependable wireless networks. Mobile Networks and Applications, 7(3), 225 - 234. 2002.
[6] Mobile complete: http://www.deviceanywhere.com/
[7] Testlabs Blog. Top 10 Tips For Testing iPhone Applications http://blog.testlabs.com/search/label/iPhone accessed September, 2010.
[8] Tommi Mikkonen – Programming Mobile Devices - an Introduction for Practitioners, John Wiley & Sons, 2007
[9] Unified Testing Criteria for Java Technology-based Applications for Mobile Devices, http://javaverified.com (2009)
[10] Wikipedia : http://en.wikipedia.org/wiki/Mobile_application_development

AUTHORS PROFILE

Selvam R is working as an **Associate Professor** in Department of Computer Applications from August 2002 to till date at Arignar Anna Institute of Management Studies & Computer Applications Pennalur, Sriperumbudur- Tamilnadu, India. He has wide experience in teaching SoftwareEngineering, Software Project Management, Object oriented Analysis and Design, Java Programming, His research interest focused on Mobile computing arena. He associates himself with Computer Society of India (CSI), IEEE and various other International Computer Societies and organization for academic advancements.

Dr Karthikeyani V is working as a **Professor** in Department of Computer Science at Govt Arts College for Women, Salem -Tamilnadu, India. She was awarded Doctoral degree from Periyar Universtiy, Salem-Tamilnadu, India. She has published 18 International Journals, 8 National Journals and Presented several papers in International and National Conferences. She is a life member in Computer Society of India (CSI), ISTE (Indian Society for Technical Education), ACM-CSTA (Computer Science Teacher Association) and various other International Computer Societies and organization for knowledge exchange and enhancement.