Testing Java ME Applications

Paul POCATILU
Economy Informatics Department, Academy of Economic Studies, Bucharest, România

Today, mobile applications have a wide use and their development is growing fast. Testing mobile applications is an important aspect of their development, keeping in mind the importance of these applications and their specific characteristics. In this paper are shown the main aspects of testing the mobile applications, focusing on unit testing of Java ME applications.

Keywords: *mobile applications, mobile devices, software testing, WAP, Java ME, JUnit*

1 Introduction

Mobile applications are used on various domains like banking, stock exchange, Internet browsing, multimedia transfers, m-learning. Mobile applications development and their use it is growing very fast, having different hardware and software platforms.

Mobile devices, like mobile phones, smartphones, PDAs (Personal Digital Assistants), communicators, pagers etc., compared with desktop computers, are characterized by:

- less computation capacity
- less memory (RAM and ROM)
- small displays
- limited user interface
- reduced dimensions
- reduced bandwidth.

Every mobile device has a specific application development platform, based on the operating system:

- Windows Mobile/CE
- Symbian OS based
- Blackberry OS
- Android
- iPhone OS
- Palm OS/ACCESS
- Linux based
- proprietary operating systems.

These constraints and variety of software platforms influence the way of designing the applications and the testing process.

Mobile applications are standalone or desktop applications and distributed applications. Standalone mobile applications are designed to perform specific tasks without the need of a network connection. Mostly mobile applications made for PDAs are such examples of stand-alone applications.

Distributed applications use a network connection that can be permanently or temporarily. WAP applications for mobile phones that access an Internet server are examples of distributed applications as is depicted in figure 1. The most used distributed applications are Web-based.

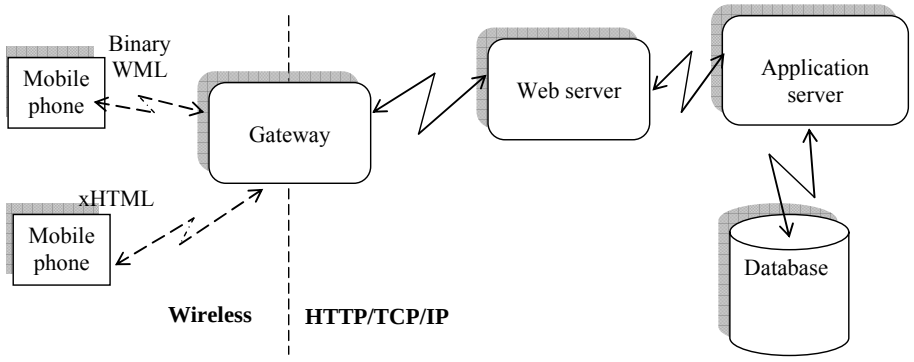

Fig.1. Mobile Web applications architecture

The request from the WAP enabled phone is sent to the WAP gateway that makes the conversion from the WAP stack (for WAP 1.0) or from the optimized wireless opti-

mized HTTP/TCP/IP (WAP 2.0) to the HTTP/TCP/IP stack and encodes the network packets that will be sent to the Web server as a HTTP request. The request is processed on the Web server and the response is send back to the mobile phone browser through the WAP gateway that decodes the packets.

2. Java Micro Edition (JME)

JME platform runs on many mobile devices, which have installed a Java Virtual Machine. The biggest benefit of using the Java platform for mobile device development is that is possible to produce portable code that can run on multiple platforms. It is almost impossible to port the complete functionalities of an application to all mobile devices because wireless devices have a vast range of capabilities in terms of memory, processing power, battery life, display size, and network bandwidth.

Java ME is divided into several different configurations and profiles. Configurations contain Java language core libraries for a category of devices. In this moment there are two configurations:

- *Connected Limited Device Configuration* (CLDC) – designed for small, resource-constrained devices (cell phones, low-end PDAs);
- *Connected Device Configuration* (CDC) – designed for relatively big and powerful devices (high-end PDAs, set-top boxes, network appliances); CDC has more capabilities than CLDC in terms of security, mathematical, and I/O functions.

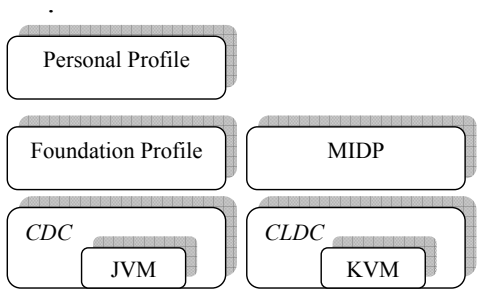

Fig.2. Java 2 Micro Edition platform

On top of each configuration are profiles. A profile defines device-specific API libraries, like GUI, networking, and persistent-storage APIs. Each profile has its own runtime environment and is suited for a range of similar devices. The main profile for the CLDC is Mobile Information Device Profile (MIDP). For the CDC there are two important profiles: the Foundation Profile and the Personal Profile.

There are emulators for different mobile devices, so until the application is running on the device, it is implemented and tested on those emulators. The emulators can be configured to provide the desired Java packages, existing on the real device.

3. Mobile application testing

The testing of mobile applications is given by the application type: standalone, distributed and Web-based.

If the mobile application is stand-alone, the testing process is similar to the desktop applications one, having in mind the limited resources of the mobile devices. The testing of these applications includes:

- functional testing
- usability testing
- structural testing
- performance testing.

For distributed mobile applications, the testing process is different. The client application and server application need to be tested not only independently but also together in interaction.

In a Web-based application the client is an Internet browser, and the main functionality is built on the server-side. Testing Web-based applications require the following type of testing [POCA03]:

- functional testing
- compatibility testing
- content testing
- performance testing
- load testing
- security testing
- server-side testing (Web server, application server)
- database testing.

Testing Web based mobile applications require the testing of the server applications and the testing of the content that is send to the mobile device. On the client side, the

script functions need to be tested (WMLScript, JavaScript etc.)
There are many aspects that influence the testing process and make it difficult. When a failure occurs, it can have many causes:

- poor user interface design
- bugs in application
- network problems
- insufficient memory
- Web server configuration
- database management system not working properly
- database scripts contains errors.

Before using the application on the mobile device, it will be tested using an emulator and the environment where is developed.
Due their constraints, testing mobile applications is a challenging process. Testers need to focus on many additional aspects that are not specific to desktop applications.

4. Automating unit testing

The automation of software testing consists of a series of processes, activities and tools brought together in order to execute the software under test and to record the result of the tests. The testing process has the following activities:

- test planning
- test design
- test implementation
- test execution
- test evaluation

Each activity has specific deliverables that are used from a phase to another. At the end, bug reports and other documentation will result. These documents are used by the development team to identify the cause of faults and to correct them. After the test plan has been elaborated, based on specific inputs (budget, resources, timeline), the next step is to analyze the requirements and to define the objectives of testing for the test team. The design phase is focused mainly on the definition and design of test procedures. At this time a decision will be made about what should be tested manually and what will be tested automatically. Test cases and test procedures are the result of test implementation phase. Test scripts are written in specific programming languages like Visual Basic, Java or C++. In this phase, some test scripts can be reused from the previous tests. Test execution has as an input the test plan and the test procedures. After the execution of the tests, the results are evaluated using an oracle. An oracle is a specialist that could decide if the result is correct or not.

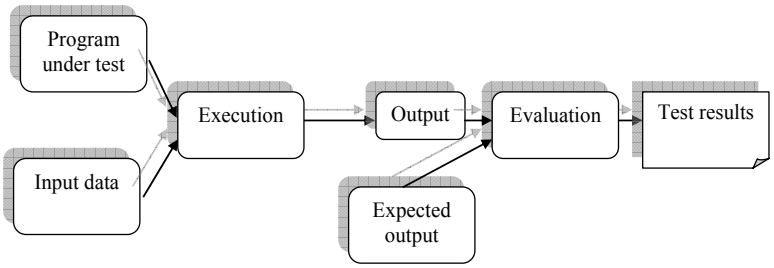

Fig.3. Execution and evaluation phases for unit testing

Tools for automated software testing of mobile applications are various, and they can be used in different areas of testing. There are many tools to assist software testing: capture/playback tools, tools for automated execution of tests, coverage analyzers, test case generators, logical and complexity analyzers, code instrumentation tools, defect tracking tools and test management tools.
For mobile applications developed using Java the JUnit framework is very good for the automated testing of the functional issues. One of the implementation is JMUnit (http://sourceforge.net/projects/jmunit/).
JMUnit is a framework for regressive unit testing of Java code. The main objects used are test cases and test suites, figure 4 [SILV08].
Each class under test (CUT) has an associated test case. Test case classes inherit *TestCase* class. For each CUT method that will be tested there is a method in the test case equivalent class. For example, *translate* method from the CUT *Dictionary* will have

the corresponding method *testTranslate* in test case class *TestDictionary* class. *TestDictionary* class includes an instance of *Dictionary* class.

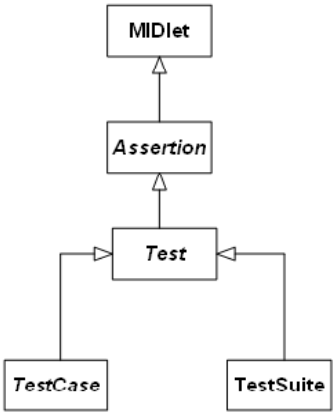

Fig.4. JMUnit main classes

The test suite includes several test cases. The base class is *TestSuite*. A test case is added to the test suite using *add* method.

Table 1 shows *testTranslate* method used to test *translate* method from *Dictionary* class. The *testTranslate* method checks if the correct answer is returned. The *assert** methods (*assertEquals*, *assertNotEquals*, *assertNull*, *assertNotNull*, *assertSame*, *assertNotSame*, *assertFalse*, *assertTrue*) pass or fail if the given expression complies or not with the assertion.

When the test suite runs, it counts the numbers of failures and successes and details are given, that allowing to developers to find and correct the errors.

Example of a test method within a test case:

```
public void testTranslate()
{ String word = "xyz";
  String expectedResult = "The word does
      not exists in the dictionary";
  //create client
  DServiceSoap_Stub client = new
      DServiceSoap_Stub();
  //call translate method
  String result =
      client.translate(word);
  assertEquals(
   result.equals(expectedResult), true);
  //...
}
```

5 Conclusions

Testing mobile applications is a very complex process, depending on the application type. For Web based applications, not only the specific response sent to the mobile device has to be tested, but the whole application from the server side. That includes the testing of JSP, ASP.NET and PHP scripts.

The use of mobile devices emulators makes the applications compatibility to be solved before the application is deployed to the mobile devices. There are emulators for almost all mobile phones platforms. Having many IDE that allows developing, running and debugging mobile applications, the testing process becomes easier. Every language used in developing mobile applications has specific characteristics that influence the testing process.

Using testing frameworks based on JUnit for Java ME applications is a good approach, having in mind experience of this framework. Of course, unit testing does not have to be limited to this framework, other methods and techniques shall be used.

References

[IVAN01] Ion IVAN, Paul POCATILU, Cristian TOMA, Alexandru LEAU - *e3-commerce: e-commerce, mobile application. Aplicaţia e3com*, Informatica Economică, vol. V, nr. 3(19), 2001, pag. 16-23

[MIKK07] Tommi Mikkonen – *Programming Mobile Devices - an Introduction for Practitioners*, John Wiley & Sons, 2007

[PATT05] Ron Patton – *Software Testing*, Sams Publishing, 2005

[POCA02] Paul POCATILU – *Automated Software Testing Process*, in Economy Informatics, vol. II, nr. 1, 2002, pp. 97-99

[POCA03] Paul POCATILU, Marius POPA – *Internet Applications Testing*, Proceedings of 6th International Conference on Economic Informatics, IE'2003, Digital Economy, Bucureşti, 8-11 May 2003, pp 1028-1032

[POCA05] Paul POCATILU – *Testing Java Programs with JUnit*, in Informatica Economică, vol. IX, nr. 2(34), 2005, pp. 51-55

[SCHL01] Christian SCHLÄPFER, Michal KUBIK, Guido ZAVAGLI, - *Mobile Applications with J2ME™*, Ericsson Radio Systems AB, 2001

[SILV08] Brunno Silva, Carl Meijer - *JMUnit 1.2. Java Micro Unit*, http://sourceforge.net/projects/jmunit/, 2008

[******] java.sun.com/javame/

[******] www.orangepartner.com