IVV TESTING MODEL FOR MOBILE APPLICATIONS

Eugen Dumitraşcu, Nicolae-Iulian Enescu, Gheorghe Marian

University of Craiova, Romania
eugen.dumitrascu@cs.ucv.ro, nicu.enescu@cs.ucv.ro, gheorghe.marian@cs.ucv.ro

Abstract: In this article, we present the specific features of mobile applications as distributed applications. We enumerate the series of evaluation's indicators of quality for mobile applications. We also present the testing model IVV (Integration Verification and Validation) and the IADT method (Inspection Analysis Demonstration and Test). However, we refer especially to the functional type testing of GUIs (Graphical User Interfaces) of some software applications and in particular of a mobile application.

Keywords: mobile application, verification strategy, IVV, IADT

1. INTRODUCTION

Software testing is an essential component in order to achieve and improve a software quality and in all software development. Software testing is characterized by the existence of many methods, techniques and tools that must fit the test situation, including technical properties, goals and restrictions. The software testing is a part of the software life cycle and must be structured according to the type of product, environment and language used. Software testing has focused on two separate issues, verification (static testing) and validation (dynamic testing) (Enescu *et al.*, 2005).

Verification is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. It is the process of evaluating, reviewing, inspecting, and doing desk checks of work products such as requirement specifications, design specifications, and code (IEEE, 1983). Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. Verification and validation are complementary (IEEE, 1983). Functional testing verifies how the application works. In functional testing, the tested data is selected according to the reference functional specification.

2. MOBILE APPLICATIONS

An application is mobile if runs on a portable computing device that is always or occasional connected to a network. This definition includes applications that run on portable computers like notebooks, PDA (Personal Digital Assistant) or mobile phones. It also implies forms of client-server applications where the application that runs on that type of device is a client application.
It is not necessary to have a permanent connection to the network, some applications are written to be run with an occasional connection to the network. Here we distinguish three different types of connections through which we can describe how an application is connected to a server (Dumitraşcu, 2003):
- permanent connection – the application does not function without connection to the network. So, the server existence is critical for the client application. For example, the application that runs on mobile phone based on mini browsers HDML/WML or other type of application 'thin client'.
- permanent available – the application requires an available connection to the network, but it runs even if the connection is not available. This model is referred to as disconnected occasional model. The server existence is important but not critical.
- occasional connection – the application runs without the network connection. The existence of a server is rather optional.

The data synchronization has an important role in mobile application (mApplications) that refers to the exchange of data between two applications contained in different stored zones. The synchronization is important for the permanent available and occasional models, whwreas without a connection to the network the client and the server do not each indicate the changes that occured.

The security is another important chapter of mobile applications, especially in wireless communications over the public network. The security is more than cryptography of data; it refers to verifying the user's identity.

In this domain, it appeared a series of standards, protocols and even virtual machins for mobile applications, such as: WAP (Wireless Application Protocol), SOAP (Simple Object Access Protocol), SyncML, XHTML Basic, J2ME (Java 2 Micro Edition).

A mobile application is considered a client-server application on three levels: on the first level it is the client application that works on portable devices and that is a graphic interface, most of the times a web interface, based on a WAP protocol; the second level or middle level contains applications that run on different servers, for instance web servers, or a mobile application server; the third level contains the storage data level or diverse databases (Dumitraşcu, E., 2003).

The quality of these applications is estimated by the indicators or metrics for distributed applications on multi-levels.

We enumerate some indicators for estimating a quality of mobile applications (Dumitraşcu, 2004):
- the graphic quality of applications that refers to the displayed mode of a user interface of a mobile application. In this domain a series of standards, protocols and even virtual machines for mobile applications have appeared;
- the degree of communicating with a remote server is given by the number of accesses in the time unit of mobile client application to server. The mobile applications didn't have to be connected permanently to a network and that is why we measure only the number of the accesses to the network;
- the access time at the wireless network;
- the synchronization degree with different and varied mobile applications.

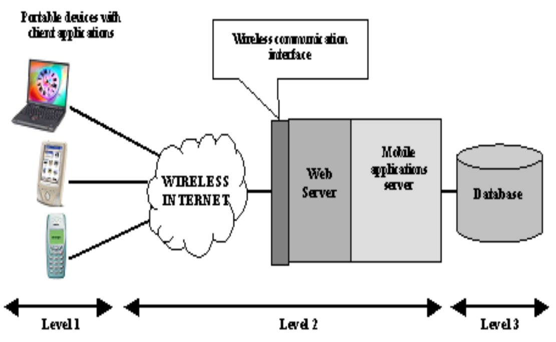

Portable devices with client applications
Wireless communication interface
WIRELESS INTERNET
Web Server
Mobile applications server
Database
Level 1
Level 2
Level 3

Fig. 1.The structure of a mobile application.

3. VERIFICATION STRATEGY

The functional testing verification represents a part of the testing process of software applications. The functional verification has the following objectives: to verify the behavior of application, to detect the defects of software, to test robustness and the performance of application in operational nominal context and degraded context.

The verification strategy required for an item (symbol or object of interface) defines four categories of tests:
- Nominal – functional verification in nominal mode
- Degraded – functional verification in degraded mode
- Robustness – verification the functional robustness
- Performance – verification of performance

The model used to test the functionality of an application is IVV (Integration Verification and Validation) and the main method is IADT (Integration Analysis Demonstration and Test). The objective of Integration is to assemble the software product from the components, ensuring that it functions properly. The objective of Verification is to ensure that the final and intermediate software product satisfy their technical requirement. Verification covers nominal operation, but also degraded operation and robustness, whether or not that specifically related requirement is included in the specification. The objective of Validation is to determine that the software product satisfies the requirements in its intended operational environment.

The IADT method represents:
- *Inspection* is a visual verification of an item.
- *Analysis* is a verification based upon analytical evidences obtained by calculation, without intervention on the verified item. The techniques used are modeling, simulation and forecasting.
- *Demonstration* is a verification of operational characteristic observable by the overseeing of the functioning component without physical measurement.
- *Test* is a verification of functional characteristics that are measurable and directly or indirectly reachable.

The nominal functional tests verify the nominal behavior for an item in the context of valid data. In this test, we verify the graphical interface for all display states of item for nominal symbolism like position, colors, shape, text; logic of display; and the behavior for all operational values of inputs data.

The degraded functional tests verify the degraded behavior in the context of invalid data. In this test we verify the graphical interface for all display states for invalid symbolism; logic and behavior of display for invalid inputs data.

The robustness tests verify how the operational behavior of an item is not degraded when the input data are out of operational context.

The performance tests verify how the operational behavior of an item is not degraded when the input data are forced to values limit.

For each item (symbol or object from user interface) we create a top test procedure with tests according to the described model presented above. Each top procedure contains intermediate procedures and each of these contains scenarios of test with one or many test cases.

In case that many items form a part of the interface, these are tested together. In same cases, one or many procedures or scenarios could be absent.

4. EXPERIMENTAL RESULTS

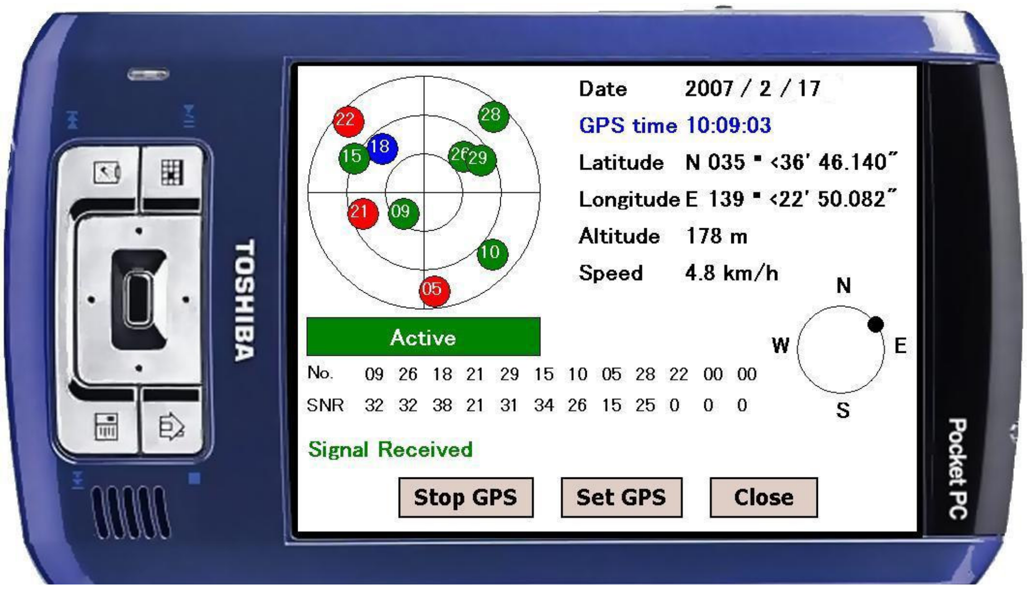
To emphasize the verification strategy model presented above we consider a mobile application that receives the current GPS position and sends it to a server to store the information about position into a database to analyze the performed route

The most essential function of a GPS receiver is to pick up the transmissions of at least four satellites and combine the information in those transmissions with information in an electronic almanac, all in order to figure out the receiver's position on Earth. Once the receiver makes this calculation, it can tell you the latitude, longitude and altitude (or some similar measurement) of its current position.

The graphical user interface that we refer to is depicted in the Fig. 2, where we present the satellites' positions and the color of strength of their signals, data and GPS time, the coordinates (longitude, latitude and altitude), the pointing of walk and the list of satellites and their strength signal. The color of the strength signal is set from the configuration signal interface presented in the Fig. 3.

For the graphical interface of the application from Fig. 2 we apply the verification strategy model for certain symbols or graphical objects as latitude and longitude.

LATITUDE (top procedure)
This symbol is a label, which indicates the latitude received from satellite.
Graphical description for valid state:
- "Latitude" in black color
- "N" or "S" black flag
- 3 black digits, right justified, leading zero, for degrees, followed by ° unit.



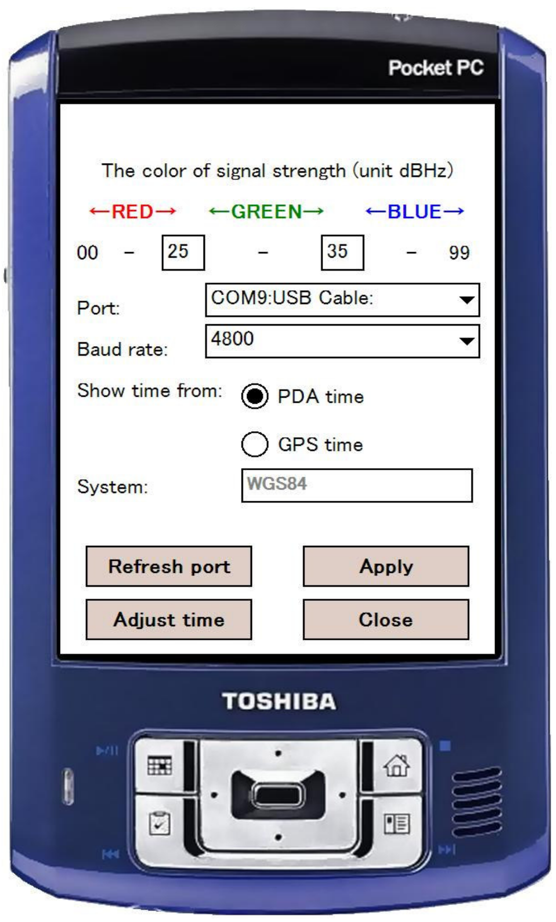
Date 2007 / 2 / 17
GPS time 10:09:03
Latitude N 035 ° <36' 46.140"
Longitude E 139 ° <22' 50.082"
Altitude 178 m
Speed 4.8 km/h

Active

No.	09	26	18	21	29	15	10	05	28	22	00	00
SNR	32	32	38	21	31	34	26	15	25	0	0	0

Signal Received

Fig. 2. Information of satellites



The color of signal strength (unit dBHz)

←RED→ ←GREEN→ ←BLUE→

00 - 25 - 35 - 99

Port: COM9:USB Cable:

Baud rate: 4800

Show time from: [✓ PDA time] GPS time

System: WGS84

Refresh port | Apply | Adjust time | Close

Fig. 3. Settings of the strength

- 2 black digits, right justified, leading zero, for minutes, followed by ' unit.
- 2 black digits, right justified, leading zero, for seconds, followed by black decimal point, 3 black digits represent the thousandth of seconds and by " unit.
- located in right of the diagram with satellites positions, below GPS time.

Graphical description for invalid state:
- "Latitude" in black color
- 3 black dashes followed by ° unit.
- 2 black dashes followed by ' unit.
- 2 black dashes followed by point, 3 black dashes and " unit.

The input parameter for latitude is positionLat that has the operational range [-90°00'00.000", +90°00'00.000"] and the codable range [-80°00'00.000", +180°00'00.000"].

a) Valid state display and input increase in operational range (Nominal and Regression procedure)

i) Graphical display scenario
Test case no. 1: positionLat = +35°36'46.140"
Display in black "Latitude N 035°36'46.140" "
Test case no. 2: positionLat = -35°36'46.140"
Display in black "Latitude S 035°36'46.140" "

ii) Increase input in operational range
Test case no. 1: positionLat from -90°00'00.000" to -0°59'59.999" with step 0°00'00.001"
Display in black "Latitude S" and values according with evolution from -90°00'00.000" to -0°59'59.999".
Test case no. 2: positionLat from 00°00'00.000" to +90°00'00.000" with step 0°00'00.001"

Display in black "Latitude N" and values according with evolution from 00°00'00.000" to +90°00'00.000".

b) Input decrease in operational range (Nominal procedure)

i) Decrease input in operational range
Test case no. 1: positionLat from +90°00'00.000" to 0°00'00.000" with step -0°00'00.001"
Display in black "Latitude N" and values according with evolution from +90°00'00.000" to 0°00' 00.000".

Test case no. 2: positionLat from -0°59'59.999" to -90°00'00.000" with step -0°00'00.001"
Display in black "Latitude N" and values according with evolution from -0°59'59.999" to -90°00'00.000".

c) Invalid state display for invalid input (Degraded and Regression procedure)

i) Invalid input
Test case no. 1: positionLat is invalid (not receive from satellite) .
Display in black "Latitude ", the flag "N" or "S" is not displayed and 3 black dashes followed by ° unit, then 2 black dashes followed by ' unit, then 2 black dashes followed by point, 3 black dashes and " unit.
Like that: "Latitude ---°--'--.---" "

d) Input increase outside operational range (Robustness and Regression procedure)

i) Increase input outside operational range
Test case no. 1: positionLat from -180°00'00.000" to -90°00'00.001" with step 0°00'00.001"
Display in black "Latitude S 090°00'00.000" "
Test case no. 2: positionLat from +90°00'00.001" to +180°00'00.000" with step +0°00'00.001"
Display in black "Latitude N 090°00'00.000" "

e) Input decrease outside operational range (Robustness procedure)

i) Increase input outside operational range
Test case no. 1: positionLat from +180°00'00.000" to +90°00'00.001" with step -0°00'00.001"
Display in black "Latitude N 090°00'00.000" "
Test case no. 2: positionLat from -90°00'00.001" to -180°00'00.000" with step -0°00'00.001"

Display in black "Latitude S 090°00'00.000" "

LONGITUDE (top procedure)
This symbol is a label, which indicates the longitude received from satellite.

Graphical description for valid state:
- "Longitude" in black color
- "E" or "W" black flag
- 3 black digits, right justified, leading zero, for degrees, followed by ° unit.
- 2 black digits, right justified, leading zero, for minutes, followed by ' unit.
- 2 black digits, right justified, leading zero, for seconds, followed by black decimal point, 3 black digits represent the thousandth of seconds and by " unit.
- located in right of the diagram with satellites positions, below Latitude.

Graphical description for invalid state:
- "Longitude" in black color
- 3 black dashes followed by ° unit.
- 2 black dashes followed by ' unit.
- 2 black dashes followed by point, 3 black dashes and " unit.

The input parameter for latitude is positionLong that has the operational range and also the codable range [-180°00'00.000", +180°00'00.000"].

a) Valid state display and input increase in operational range (Nominal and Regression procedure)

i) Graphical display scenario
Test case no. 1: positionLong = +139°22'50.082"
Display in black "Longitude W 139°22'50.082" "
Test case no. 2: positionLong = -139°22'50.082"
Display in black "Longitude E 139°22'50.082" "

ii) Increase input in operational range
Test case no. 1: positionLong from -180°00'00.000" to -0°59'59.999" with step 0°00'00.001"
Display in black "Longitude E" and values according with evolution from -180°00'00.000" to -0°59'59.999".

Test case no. 2: positionLong from 00°00'00.000" to +180°00'00.000" with step 0°00'00.001"
Display in black "Latitude W" and values according with evolution from 00°00'00.000" to +180°00'00.000".

b) Input decrease in operational range (Nominal procedure)

i) Decrease input in operational range
Test case no. 1: positionLong from +180°00'00.000" to 0°00'00.000" with step -0°00'00.001"
Display in black "Longitude W" and values according with evolution from +180°00'00.000" to 0°00' 00.000".

Test case no. 2: positionLong from -0°59'59.999" to -180°00'00.000" with step -0°00'00.001"
Display in black "Longitude E" and values according with evolution from -0°59'59.999" to -180°00'00.000".

c) Invalid state display for invalid input (Degraded and Regression procedure)

i) Invalid input
Test case no. 1: positionLong is invalid (not receive from satellite)

Display in black "Longitude ", the flag "W" or "E" is not displayed and 3 black dashes followed by ° unit, then 2 black dashes followed by ' unit, then 2 black dashes followed by point, 3 black dashes and " unit.
Like that: "Longitude ---°--'--.---" "

Since the operational range is identical to codable range, we don't have Robustness test procedures.

5. CONCLUSIONS

The testing of software programs is done for two reasons: detecting errors and estimating reliability.

It must be remembered the following three testing principles:
- the testing is the process of running a program with the intention for finding errors
- a good test is the one that has a higher probability to detect a previously undiscovered error
- a successful test is the one that detects a previously undiscovered error.

When the test produces a situation when the results of the actual module do not match the expected results, there are two possible explications: the module contains an error or the expected results are incorrect.
The integration and testing represent almost 40% of programming engineering costs and the other 60% is represented by the development costs. Therefore, testing is an important phase in realizing the software products.

The model IVV is a verification model, which applies for diverse applications, namely mobile applications, embedded applications, or other applications that have a graphical user interface.

Some producers of embedded software for avionics or mobiles apply this method in the testing process of their software.

REFERENCES

Dumitraşcu, E. (2004). The indicators of evaluating the quality for distributed applications. *Paper presented in the PhD program*, A.S.E., Bucharest.

Dumitraşcu, E. (2003). Structures of distributed applications. *Paper presented in the PhD program*, A.S.E., Bucharest.

IEEE (1983), *IEEE standard for software test documentation: IEEE/ANSI standard 829–1983*

Enescu, N., E. Dumitraşcu, I. Ivan, P. Sinioros (2005). The Classification of Software Testing Techniques. *The Proceedings Of The Seventh International Conference On Informatics In Economy*, p.1328-1333

Ivan, I., P. Pocatilu, C. Toma, A. Leau (2001). e3-commerce: e-commerce, mobile application. Aplicaţia e3com, in the review: *Economical Computing*. Economical Computing Department and the INFOREC Association, **vol. V, No. 3 (19)/2001**, p. 16 – 23

Myers, G. (1978). *The art of software testing*. John Wiley & Sons Ed.

Ould, M., and Unwin, C. (1986). *Testing in software development*, Cambridge Univ. Press.

Roper, M. (1994). *Software testing*, McGraw–Hill, 1994

Sommerville, I. (1996). *Software engineering*, Addison–Wesley Ed.