CMER

Centre for Mobile Education and Research

Mobile Devices in Software Engineering

Week III

Overview

- Testing
- Test Stub & Driver
- White Box Testing
- Black Box Testing
- Testing Levels
- Emulators & Devices
- Testing Checklists
- Software Deployment
- Software Maintenance
- Test Cases

Testing

- As discussed in the previous weeks, the challenges that are faced with developing mobile applications means the testing phase for software development has a great importance which is already increased from development of applications on desktops

- There are certain guidelines that have been developed on how testing should be done for mobile applications, what to look for, how much time should be spent, etc.

Why is testing important?

- Software testing is an integral and important part of the software development process, it is at this point that hopefully all bugs within the system are discovered and corrected
- Bugs will usually exist in any system of larger size, this is usually due to design complexity rather then human error as it is difficult for humans to manage complexity of larger scales
- Bugs found in code at late stages in software development will be expensive and time consuming to fix

Why do we test?

- **Some of the reasons why we wish to test mobile applications:**
  - **Different J2ME implementation on devices**
  - **Different display sizes**
  - **Proprietary API extensions**
  - **Network Performance**
  - **Different processing power on devices**
  - **Discovering bugs or errors in code**
  - **Finding missing requirements**

Software Testing

- **Software testing goes hand in hand with Verification and Validation, which asks two questions:**

- **Verification**
  - **Have we built the system correctly?**
- **Validation**
  - **Have we built the correct system?**

Goal of Software Testing

- The goal of software testing is to improve the quality of the product, quality which is comprised of: [1]

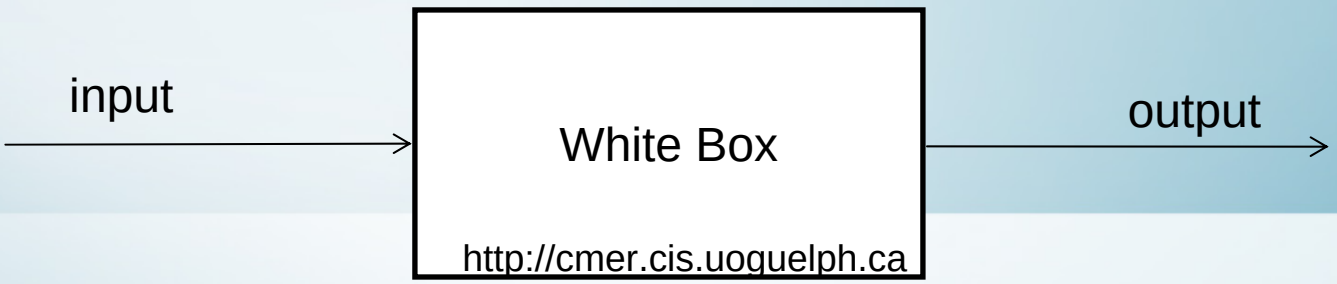
- Functionality (External Quality)
  - Correctness, Reliability, Usability, Integrity
- Engineering (Internal Quality)
  - Efficiency, Testability, Documentation, Structure
- Adaptability (Future Quality)
  - Flexibility, Reusability, Maintainability

Test Stub & Test Driver

- Test Stubs and Test Drivers are commonly used during the software testing period
- The use of stubs and drivers is necessary as testing may want to be done on components prior to being totally complete
- Test Stub
  - **A piece of code that simulates a missing component of the system**
- Test Driver
  - **A piece of code that passes test case data to a component which will be tested**

White Box Testing

- White Box testing is a testing technique where the tester has knowledge and access of the internal workings of the component they are testing
- Test cases are derived from the program structure as the testers are aware of the logic, style, and language of implementation
- The goal of White Box testing is to cover as much or all of the lines of code that create a component
- These types of tests, while not only discovering bugs, can find code that is unused and can be discarded



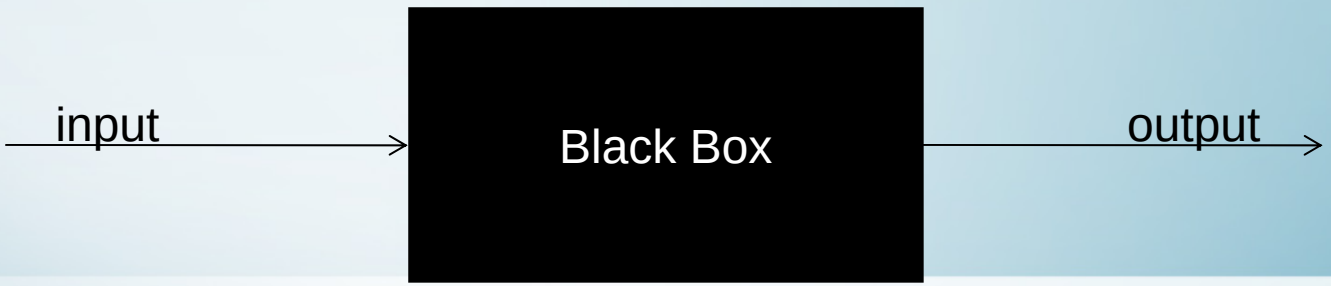
White Box Testing

- There are two ways in which White Box testing can be done

- Control Flow testing
  - This testing involves testing the way the individual lines of code are executed in order
- Data Flow testing
  - This testing involves looking at the lifecycle of a particular piece of data during the execution of a function in the system

Black Box Testing

- The Black Box Testing technique deals with validating the input and output of a particular component
- This technique does not have knowledge of the internal workings of the component being tested
- Tests take into account valid input data and all invalid input data and verify the correct responses by the system
- The majority of tests are designed to look at the functional requirements of the system when using this method



Gray Box Testing

- Gray Box testing is a technique that combines both White and Black Box testing techniques
- In this technique the user is aware of some of the internal workings of the component they are testing
- A tester will apply some tests to the internal mechanisms of the component and will then apply the rest of the tests using a Black Box testing technique

Levels of Testing

- There are various levels of testing that a system can undergo:
  - Unit Testing
  - Integration Testing
  - Functional Testing
  - System Integration Testing
  - User Acceptance Testing
  - Regression Testing
  - Usability Testing
  - Network Performance Testing
  - Server-Side Testing

Unit Testing

- Unit testing deals with the testing of each basic component that comprises the system
- Unit testing does not require the entire system to be complete, a single unit can be verified to be working properly
- Testing individual parts helps eliminate doubt of the individual units when the components are brought together to form a complete system
- Integration testing becomes easier when unit testing is done as there is some confidence in the units themselves

Integration Testing

- Integration testing is the testing of the individual units when they have been combined as a group
- Integration testing wishes to test the functional, performance, and reliability requirements of a group
- Three common strategies for Integration Testing:
  - **Top-Down**
  - **Bottom-Up**
  - **Umbrella**

Integration Testing Strategies

- **Top-Down**
  - **High level modules are tested first, minimizes the need for drivers but stubs complicate testing**
  - **Low level modules are tested late in development and poor support for early releases**
- **Bottom-Up**
  - **Low level modules are tested first, need for stubs is minimized but drivers complicate test management**
  - **High level functionality is tested late in development**
- **Umbrella**
  - **Minimizes the need for stubs and drivers, the inputs to functions are done in a bottom-up manner and the outputs of functions are done in a top-down manner**
  - **This approach could lead to much more regression testing**

Functional Testing

- Functional testing is done to ensure that a system meets the requirements set out by business cases, models, story boards, or any other design format
- A requirement is anything that a user utilizing the system would expect to see from the system based on the business rule
- All the components of the system are together and the entire application is tested
- Functional tests are usually are done with Black Box testing technique

System Integration Testing

- System Integration Testing is done to ensure that any new application that interacts with existing systems does so in a correct manner
- System Integration Testing occurs prior to User Acceptance Testing
- System Integration is a Black Box testing technique

User Acceptance Testing

- User Acceptance Testing is a Black Box testing technique
- Functional tests are created and executed by the users or clients of the system
- Tests give final confidence that the business requirements have been met by the developers of the system
- Tests should be written by different authors then those of the previous testing levels
- This is the final verification point prior to the deployment of the system into a real world environment

Regression Testing

- Regression testing is done when a piece of software was working as expected and is suddenly not working as expected
- A common practice is when bugs are found and fixed, the tests that originally located those bugs are used as regression tests whenever changes to the software is done
- It is a good practice to build a library of tests that can be used as a base set for regression testing any time there is a significant change to the system

Usability Testing

- Usability testing deals with the flow or processes a user will take when using the application
- We wish to look at the amount of screens a user must navigate to achieve their goal and ask questions such as, is the user navigating through too many screens to achieve their goal
- As the application is on a mobile device, text entry is difficult, we want to know if we have minimized the amount of text entry users have to do to fill our information
- If we notice we must navigate through too many screens we should consider remodelling the way that function is done

Network Performance Testing

- Network Performance Testing deals with verifying how the application reacts to certain conditions relating to the network
- Tests can be done to see how the application performs when the battery is low, the signal to the network has degraded, or when the connection has been lost
- These types of tests are carried out in emulators and on real devices, however testing on real devices and a network becomes difficult if we wish to reproduce scenarios as there can be factors outside our control

Server-Side Testing

- Your application may interact with a remote server, it will be important to test the server component as well
- There may be instances where you will not have control of the server you are interacting with, such as a publicly accessible stock quote server found at Yahoo or Google
- It will be important to simulate those servers as best as possible to test your application

When to stop testing?

- Testing can potentially never end, therefore it is important to know when to stop testing
- The amount of testing that is done on a system is the outcome of a number of factors
- Budget, time, and quality of the product will be considered when determining when to stop testing
- Commonly testing stop when budget, time, or test cases have been exhausted, however testing should ideally stop when the reliability of the product has met the requirements
- Testing is a balance between attempting to predict the potential bugs left in the system and the cost with continued testing

Emulators vs. Devices

- Tests can be carried out in the emulation environments where the implementation has been completed such as the Sun Wireless Toolkit or Blackberry JDE
- The use of emulators should be the first step in the testing process, ultimately you will need to use real devices to test your applications
- When developing mobile applications never deploy an application prior to physically testing it on a mobile device

Testing Checklist

- The following are testing checklists provided in "Engineering Wireless Mobile Applications“
- These checklists contain certain tests that are performed by Nokia and Motorola
- These tests will help you look at key areas in your mobile application and hopefully help you resolve any issues

Navigation Checklist

- **Navigational Path Tests:**
  - **Successful startup and exit**
    - **Application starts up properly and the entry to the application is consistent, the application should exit properly as well**
  - **Application Name**
    - **Application should display the name in a title bar**
  - **Keep the user informed**
    - **If the application does not start within a few seconds it should alert the user, larger applications should contain a progress bar**

Navigation Checklist (Cont.)

- **Readable Text**
  - **All the content in the application should be readable on both greyscale and colour devices, the text should be spelled correctly and should contain no grammatical errors**
- **Repainting Screens**
  - **Screens should be properly painted, the application should not repaint screens unnecessarily**
- **Soft Buttons**
  - **Soft buttons should be consistent throughout the application, layout of screens and buttons is consistent**

Navigation Checklist (Cont.)

- **Screen Navigation**
  - **The most commonly used screens should be the easiest to navigate towards and use the least amount of button presses as possible**
- **Portability**
  - **The application should have a friendly user interface on all devices that it has been tested on**

Network Checklist (Cont.)

- Testing the network you can look at:
  - Sending/Receiving data
    - The application should send and receive data properly if it interacts through a network
  - Name Resolution
    - IP addresses should be resolved correctly, data should be sent and received properly
  - Sensitive Data
    - Data should be masked or encrypted when sent over a wireless connection

Network Checklist (Cont.)

- **Error Handling**
  - **Error messages should be displayed properly, when an error box is dismissed the application should regain control**
- **Interruptions**
  - **When the device receives SMS messages or other alerts the application should display these messages properly and regain control when these messages are dismissed**

Software Deployment

- After testing has been completed the system is ready for deployment into a real world scenario
- Software Deployment consists of:
  - Release
    - When software is ready for users
  - Installation
    - Software is installed on the user's machine
  - Activation
    - Software is activated and used
  - Deactivation
    - When software is removed or no longer used on the system
  - Updating
    - When a newer version of the software is available and updated over the existing version

Software Deployment (Cont.)

- There are several different ways that an application can be placed on a mobile device
- Software can be downloaded from a website and then installed via a cable connection to the mobile device
- A newer method is over the air provisioning where a user downloads and installs the application wirelessly to their mobile device

Over the Air Provisioning

- Over the Air Provisioning is a relatively new method of obtaining applications onto your mobile device
- This method allows a user to enter a web portal, find an application, have that application download onto their device and install that application, this is all done wirelessly
- The mobile device is required to have a tool that can discover MIDlets available for download, either through the browser or some third party software

Software Maintenance

- Once the application has been deployed there may be input from users regarding the system you have developed
- You should provide a point of contact for these individuals to notify you of errors in your application that you may have overlooked during testing
- Providing fixes to these errors and updates will ensure customers continue to use your application

Test Cases

- When testing your application it is important to document the tests that you will carry out during this phase
- This will help you to keep track of tests you have done, tests which have passed or failed, and the conditions surrounding those tests
- It also allows others to view your test cases and provide input on test conditions you may have missed

Test Cases (Cont.)

- Test Cases are written based on use cases, business rules, sequence diagrams, story boards, etc.
- Test Cases usually contain the format of:
  - **Summary**
  - **System Configuration**
  - **Initial Condition**
  - **Steps of Execution**
  - **Results of Execution**
- Depending on the testing level there can also be other criteria as in the importance of the test case or the severity of the bug found

Test Case Example

Test Case Name	Invalid Characters
Summary	This test will verify an error message appears when a user enters invalid characters in the login text box and clicks submit
System Configuration	All Systems should be up and running normally
Initial Condition	User is at the login screen
Steps of Execution	1. Enter invalid characters such as (!,@,#,$,%,^,&,*,(,),-) into the user name field 2. Enter invalid characters such as (!,@,#,$,%,^,&,*,(,),-) into the password field 3. Click the submit button 4. An error message appears telling the user they have entered invalid characters into the fields
Results of Execution	Passed

References

- [1] http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/
- [2] Mahmoud, Qusay H., and Zakaria Maamar. "Engineering Wireless Mobile Applications." Int. J. of Information Technology and Web Engineering 1.1 (2006): 58-73.
- http://msdn.microsoft.com/en-us/library/aa292128(VS.71).aspx
- http://www.devbistro.com/articles/Testing/Requirements-Based-Functional-Testing
- http://www.buzzle.com/editorials/4-10-2005-68349.asp
- http://msdn.microsoft.com/en-us/library/aa292167(VS.71).aspx