How to Test Mobile Apps

Course Notes

Robert V. Binder
System Verification Associates, LLC



Contents

Course Overview 2
Audience 2
Summary 2
Lectures and Labs 3
1. Introduction 3
2. Why We Test 3
3. Model-Driven Testing 4
4. Use Cases 4
5. Actions and Variables 5
6. Use Case Variants 5
7. Test Cases 6
8. Expected Results 6
9. Negative Testing 7
10. The Basic Tour 7
11. Testing Device Events 8
12. More Tours 8
13. Running Your Tests 9
14. Scaling Up 9
About the Author and Instructor 10
Glossary 11
Further Reading 14
Credits 15
Music 15
Beta Testers 15
Notices 15
Copyright 15
Acknowledgments 15
Trademark Acknowledgments 15
Case Study Sources 15

Course Overview

Audience

This course is designed for persons having basic computer skills and familiarity with using a smartphone or cell phone. No prior knowledge of software programming or testing is assumed. The total presentation time is just under four hours, not including participant lab time.

Summary

How to Test Mobile Apps presents practical test design techniques for mobile application developers. You'll learn how to release high-quality apps that will garner more stars. You'll learn many mobile-specific testing strategies as you create a complete test plan for a mobile app of your choice. No background in programming or software testing is needed.

The test approach is black-box and applicable to all popular mobile platforms including Android, Blackberry, iPhone, iPad, Windows Mobile, Palm OS, and mobile web apps including HTML5 sites. The resulting test plans can be run by hand or with an automated testing system.

You'll learn how to discover bugs that commonly plague all kinds of mobile apps, including failures related to incoming calls, power cycles, low signal strength, varying form factors/platforms, and input errors that can freeze a smartphone.

Course notes, a test plan template, and a case study test plan are included. The template has all elements of a complete test plan, following IEEE standard for software test documentation. You'll learn how to transform the template into your own test plan in step-by-step labs for each lecture.

When you finish the course, you'll have a complete ready-to-use test plan for your mobile app. Because the approach is platform agnostic, your test plan can be re-run to support cross-platform development, and multiple deployment configurations.

Lectures and Labs

1. Introduction

<table>
<tr>
<td>

- About this Course
- What you will Learn
- Advanced Topics not in this Course
- What's in it for You?
- Course Overview
- About the Labs
- Case Study: Groupon Mobile App
- Resources and Questions
- Recap

</td>
<td>

Lab 1: Start your Test Plan

1. Choose a mobile app to test.
2. Download and open the test plan file.
3. Edit the Cover Sheet tab.
4. Save As new file with the app name.

Case Study Example

Groupon-Example-Test-Plan-01.xlsx

</td>
</tr>
<tr>
<td colspan="2">Groupon Mobile App
http://www.groupon.com/mobile</td>
</tr>
</table>

2. Why We Test

<table>
<tr>
<td rowspan="2">

- Why We Test
- What is a Bug?
- What is Software Testing?
- Why do We Test?
- Some Jargon
- Recap

</td>
<td>

Lab 2: Entomology

1. What do AUT, DUT, and SUT stand for?
2. What is the difference between programming, testing, and debugging?
3. How many bugs have been found in your app?
4. How would users have reacted if you had not fixed them?
5. How many bugs do you think are left in your app? Why?

</td>
</tr>
<tr>
<td>Case Study Example

Not applicable</td>
</tr>
<tr>
<td colspan="2">The Firefox mobile bug reports provide many examples
http://www.mozilla.org/en-US/mobile/4.0/releasenotes/</td>
</tr>
</table>

3. Model-Driven Testing

<table>
<tr>
<td rowspan="2">

- The Mobile App Test Model
- Tour
- Use Case
- Variant
- Variables
- Test Step
- Action
- Event
- Model/Lecture Roadmap
- Recap

</td>
<td>Lab 3: What's in your Model?

1. What are the parts of the MAT?
2. What other kinds of models have you used?
3. Besides what you know about your app, is there any other written description that you could use to develop your model? For example: requirements or design documentation, user documentation, help file, feature description.</td>
</tr>
<tr>
<td>Case Study Example

Not applicable</td>
</tr>
<tr>
<td colspan="2">No Links</td>
</tr>
</table>

4. Use Cases

<table>
<tr>
<td rowspan="2">

- How do Users Use the App?
- Use Cases of Groupon Mobile?
- How to Identify Use Cases
- Groupon Mobile Use Cases
- Use Case Checklist
- Add Use Case and Basic Variant
- Use Case Details
- Recap

</td>
<td>Lab 4: Map your Use Cases

1. Give an example of well-named and poorly named use case.
2. Make a list of use cases for your app.
• Review the sources you collected in lab 3. Distill a list following the approach you've just seen.
• In your test plan file, add a tab for each use case (copy the use case template tab.)
• Save the file.
• After you've worked out an initial list of use cases, apply the checklist from slide 8, and revise as needed.</td>
</tr>
<tr>
<td>Case Study Example

Groupon-Example-Test-Plan-04.xlsx</td>
</tr>
<tr>
<td colspan="2">Groupon Mobile web site
http://www.groupon.com/mobile</td>
</tr>
</table>

5. Actions and Variables

<table>
<tr>
<td rowspan="2">

- Test Steps
- Variables
- Sign Up Use Case
- Sign Up Actions and Variables
- How to Map Actions and Variables
- Add Variables and Test Steps to Variants
- Recap

</td>
<td>Lab 5: Actions and Variables

For each use case:
1. Interact with the AUT.
• Note each step and what is output (panel, popup, audible output, haptic output, etc.) These are your actions.
• Note the names of the controls and data inputs involved in each step. These are your action variables.
2. Open your test plan file and add an action tab for each use case:
• Add a column for each action variable.
• Add a row for each action in your use case. We'll add data for the variables later.</td>
</tr>
<tr>
<td>Case Study Example

Groupon-Example-Test-Plan-05.xlsx</td>
</tr>
<tr>
<td colspan="2">No Links</td>
</tr>
</table>

6. Use Case Variants

<table>
<tr>
<td rowspan="2">

- Go with the Flow
- Where do flows go in our test plan?
- What is a Use Case Variant?
- Groupon Browse Deal Use Case
- Browse Deal Options
- Comparison of Browse Deal Options
- Variants or Variables?
- Browse Deal Variants
- How to add a variant to your Test Plan
- Recap: Use Case Variants

</td>
<td>Lab 6: Variations

1. When should you add a use case variant, or variables?
2. Check your use cases. Are there any that would be simpler if split into variants?
3. If yes, revise your test plan's use cases accordingly.</td>
</tr>
<tr>
<td>Case Study Example

Groupon-Example-Test-Plan-06.xlsx</td>
</tr>
<tr>
<td colspan="2">No Links</td>
</tr>
</table>

7. Test Cases

• How to Select a Test Case • Typical Test Data • Groupon Sign Up Test Cases • Add Test Cases to Actions • Where to Add Test Steps? • Recap: Test Cases	Lab 7: Add Typical Test Cases 1. Open your test plan file. 2. For each use case action tab, choose and enter typical data values for each action variable. 3. Save the file.
	Case Study Example Groupon-Example-Test-Plan-07.xlsx
No Links	

8. Expected Results

• What are Expected Results? • How to get Expected Results • Why Bother? • Huh? • Expected for Sign Up • What about “Exploratory Testing”? • Recap: Expected Results	Lab 8: Setting Expectations 1. Open your test plan file. 2. For each variant: • If you’ve already entered expected results, check them to be sure they’re useful (refer to slides 5 and 6.) • Add expected results as needed. 3. Save the file.
	Case Study Example Groupon-Example-Test-Plan-08.xlsx
No Links	

9. Negative Testing

<table>
<tr>
<td rowspan="2">

- Going Negative
- Why do we need Negative Tests?
- The Bad and the Ugly
- Use Case Variants for Error Tests
- Testing the Limits
- Excluded Combinations
- Structures and Mutants
- Walk on the Fuzzy Side
- Putting it All Together
- What's Expected of the Bad or Ugly?
- Negative Tests for *Sign Up*
- Recap: Negative Testing

</td>
<td>Lab 9: The Bad and The Ugly

1. Open your test plan file.
2. Make a variant for each use case, e.g., myUseCase-boundary-tests (copy and rename the tab.)
3. For each variable, decide what atypical values to use (off low, off high, fuzz, etc.).
4. Add an action for each atypical value with an appropriate expected result.
5. Save the file.</td>
</tr>
<tr>
<td>Case Study Example

Groupon-Example-Test-Plan-09.xlsx</td>
</tr>
<tr>
<td colspan="2">No Links</td>
</tr>
</table>

10. The Basic Tour

<table>
<tr>
<td rowspan="2">

- What is a Tour?
- The Basic Tour
- Groupon Basic Tour
- The Test Plan Tour Tab
- Adding the Groupon Basic Tour
- Recap: The Basic Tour

</td>
<td>Lab 10: Just your Basic Tour

1. Design a basic tour for your app.
2. Open your test plan file.
3. On the Tours tab, edit the Basic Tour description and add each variant in the tour.
4. Try to run this Basic Tour on your app. Revise as the variants or tour plan as needed.
5. If your Basic Tour takes more than five minutes, could you shorten it?
6. Save the file.</td>
</tr>
<tr>
<td>Case Study Example

Groupon-Example-Test-Plan-10.xlsx</td>
</tr>
<tr>
<td colspan="2">No Links</td>
</tr>
</table>

11. Testing Device Events

- Device Events
- Events and Actions
- Event Bugs
- How to Test Event Effects
- Adding Event Test Steps to the Sign Up Use Case
- Events and How to Trigger Them
  - Background/Foreground
  - Background Load
  - Power Sleep Cycle
  - Battery Drain
  - Incoming Call or Text
  - Memory Cards
  - Camera Usage
  - Bar code scanner
  - Accelerometer
  - Reboot
  - GPS Signal fade/interference
  - WiFi Signal fade/interference
  - Cellular Signal fade/interference
- Expected for Events?
- How Many Event Tests Do I Need?
- Recap: Testing Device Events

Lab 11: The Main Events

1. Open your test plan file.
2. Add your use cases to the Event Plan sheet.
3. Assign events to use cases so that every use case has at least one event, and every event is assigned to at least one use case.
4. Create a new event test variant for each use case: copy and rename the basic variant.
5. Insert a new column(s) for the event trigger(s).
6. Add row(s) for the event trigger(s).
7. Save the file.

Case Study Example

Groupon-Example-Test-Plan-11.xlsx

My Touch Bug
http://www.youtube.com/watch?v=LfI7AT_eD14

12. More Tours

- Minimum Recommended Tours
- More Tours
- Tours, Use Cases, and Variants
- Install-update-uninstall Tour
- Whitaker's Tours
- User-oriented Tours
- Certification Tours
- How to Map your own Tours
- Recap: More Tours

Lab 12: Touring Tests

1. Open your test plan file.
2. Is every variant used in at least one tour? If no, add tours so that they are.
3. Define a new tour. Choose one of Whitaker's tours, Install-update-uninstall, a cert tour, or devise your own.
4. Update the Tours tab, following the steps shown in lecture 10.
5. Save the file.

Case Study Example

Groupon-Example-Test-Plan-12.xlsx

No Links

13. Running Your Tests

<table>
<tr>
<td rowspan="2">

- Can I Test on an Emulator?
- Make a File for each Test Run
- Adding a Test Run for Groupon Mobile
- Verdicts
- Test, Observe, Decide, Record
- Logging your Test Run
- Entering Test Results
- When the App Fails, you Succeed
- Recap: Running your Tests

</td>
<td>Lab 13: Run your Test Plan

1. Make a copy of your entire test plan file and name it with the date and time of your test run.
2. For each tour and each of its use case variants: Perform the actions and events, entering the planned data, controls, and triggers. Compare the actual result to the expected result. Note the verdict.
3. Save the file.</td>
</tr>
<tr>
<td>Case Study Example

Groupon-Example-Test-Plan-13.xlsx</td>
</tr>
<tr>
<td colspan="2">Mozilla Bug Report Guidelines
https://developer.mozilla.org/en/bug_writing_guidelines</td>
</tr>
</table>

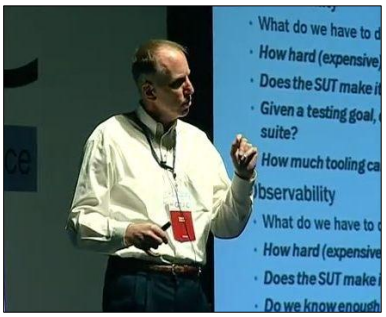
14. Scaling Up

<table>
<tr>
<td>

- Scaling Up
- Deployment Scope - Groupon Mobile
- Deployment Scope
- Which Deployment Configurations?
- Beyond Manual Testing
- Automated Mobile App Testing
- App Life Cycle and Testing
- Course Recap

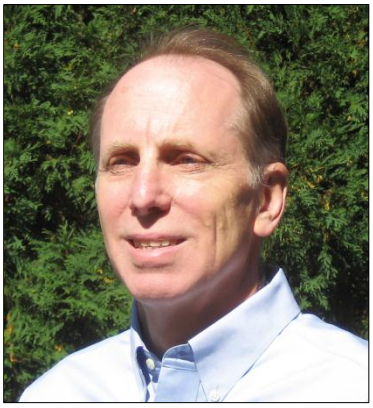
</td>
<td>No lab</td>
</tr>
<tr>
<td colspan="2">No Links</td>
</tr>
</table>

About the Author and Instructor



Robert V. Binder is a business leader, serial entrepreneur, and software technologist with extensive systems engineering experience.

As President of System Verification Associates, he has provided solutions for clients facing existential regulatory challenges. As CEO and founder of mVerify Corporation, he took a unique solution for mobile app testing to market. He led RBSC Corporation's consulting practice in software process and advanced software testing, delivering expertise and solutions globally.



Binder has developed hundreds of application systems and advanced automated testing solutions, including two projects released as open source. He was awarded a U.S. Patent for model-based testing of mobile systems.

He is internationally recognized as the author of the definitive *Testing Object-Oriented Systems: Models, Patterns, and Tools* and two other books. His 58 articles in leading publications and peer-reviewed journals have been cited over 2,100 times. He is a member of the Editorial Board of the *Journal of Software Testing, Verification, and Review* and served as member of Agitar Software's advisory board.

Binder is a sought-after keynote speaker for events like the Google Test Automation Conference and IEEE International Symposium on Software Reliability Engineering.

He received the MS in Electrical Engineering and Computer Science from the University of Illinois at Chicago and the BA and MBA from the University of Chicago. He is an IEEE Senior Member and holds the CSDP.

Glossary

Term	Definition
accelerometer	A device component that detects changes in the physical orientation of the device.
action	A sequence of user interactions that accomplishes a small step. For example, start an application (scroll to the application icon, touch the icon.)
action variable	See **variable**.
actor	A class or kind of persons who use an application, e.g. customer, merchant, help desk.
actual result	Given the input of a test case, the output or other actions observed from the AUT.
agnostic	Jargon. In software, indicates that a technology or technique may be used in more than one environment or platform.
alternate flow	The additional or specialized steps of a use case that may or may not be taken.
app	Application. A software program, service, or interactive web site that runs on a device.
AUT	Application Under Test
automated testing	A test run in which a software program enters test inputs, observes actual results, and logs verdicts for the AUT.
basic flow	The steps of use case that included in all of its flows.
basic tour	A brief tour that includes only a few main features.
blocked	A test verdict given when a test step cannot be performed owing to a bug.
boundary value	For a variable, the values that are valid (within the boundary) or invalid (outside of the boundary).
bug	(1) An observed failure of the AUT: incorrect output, hang, freeze, crash, response too slow, security flaw. (2) A missing or poorly designed feature.
bug id	An identifier assigned to a bug by a bug tracking system.
control	A graphic element of device display (cancel button, scroll slider, etc.) used to command the AUT.
deployment	The release of an app to all of its intended users and environments.
device	A hardware/software platform on which an app runs.
DUT	Device Under Test. The hardware unit on which an AUT runs.
emulator	A software program that provides a virtual platform on which to run a mobile app.
escape	When a bug remains in software after release to its users, that bug is said to have escaped.
event	A stimulus recognized by a device that may have an effect on the application under test. For example, loss of wireless signal or incoming call.
event plan	A check list in the test plan template used to assign event tests to use cases.
exception flow	The additional or specialized steps of a use case taken only when atypical inputs or conditions occur.
expected result	Given the input of a test case, the output or other actions expected from the AUT.

Term	Definition
fail	A test verdict given when actual results do not match expected results.
failure	A manifestation of a bug.
field	A part of a panel that accepts alphanumeric data.
fuzzing	An attempt to overflow a field with excessive and/or corrupted input.
GPS	Global Positioning System
hard key	A physical button on a device or accessory that may be pressed.
haptic	A tactile input or output, for example a vibration to indicate an incoming call.
IEEE	Institute of Electrical and Electronics Engineers. An international professional organization that publishes a wide range of technical standards for information technology and electronics. Pronounced as “eye triple e.”
IEEE 829	An internationally recognized standard for software test documentation.
impairment	A physical condition that reduces the strength or quality of a wireless signal.
key press	The action of pressing a virtual or real alphanumeric key on a device. Causes a single character to be input to an application.
manual testing	A test run in which a person enters test inputs, observes actual results, and logs verdicts.
MAT model	Mobile App Test model. The model for mobile testing consisting of tours, use cases, variants, test steps, variables, actions, and events.
mobile app	A software program that runs on a mobile device.
mobile device	A small portable computer with built-in wireless communication capabilities, for example, a cell phone, smartphone, or tablet computer. Usually hand-held, but may be embedded in other systems.
mutant	A test case that breaks a rule of the structured variable, for example an email address with no @ delimiter.
negative test	A test that checks how the AUT responds to atypical input values or action sequences.
panel	A structured image displayed on device, e.g., a sign-in panel.
pass	A test verdict given when actual results match expected results
SID	Step ID. A sequence number for the test steps in a variant.
skipped	A test verdict given when a test step is not performed at the tester’s discretion.
soft key	An image whose display area is sensitized to touches, emulating the effect of pressing hard keys.
SUT	System Under Test
test case	A set of values for the variables of an action and an expected result.
test model	A structured and simplified description of the AUT that makes it easy to produce test cases.
test plan	A documented collection of tours, use cases, and variants.
test plan template	A generic test plan that can be used to produce a specific test plan.
test run	One execution of a test plan, producing actual results and verdicts for each executed test case.
testing	The use of software with the intent of finding bugs.
tour	A sequence of use case variants selected to reflect a usage pattern that may be likely to reveal certain kinds of bugs or that is representative of a usage scenario.

Term	Definition
trigger	An action or condition that results in an event recognized by the DUT.
typical test	A test that checks how the AUT responds to typical inputs.
UCID	Use case ID. A sequence number for the use cases in a test plan.
use case	A sequence of actions that accomplishes a user goal. For example, select payment type.
use case variant	See **variant**.
variable	A data item, control, or trigger that is part of a variant.
variant	A sequence of test steps for a use case flow or a test purpose such as negative tests or event tests. A variant is created from the variant template sheet.
variant template	A sheet in the test plan template to be used to create specific variant sheets.
verdict	For a test case, the decision reached after comparing expected and actual results: pass, fail, skip, or blocked.
VID	Variant ID. A sequence number for the variants of a use case.

Further Reading

Testing Object-Oriented Systems: Models, Patterns, and Tools. Robert V. Binder. Addison-Wesley, 1999.

In-depth presentation of 38 test design patterns, including efficient selection of typical and invalid values. Nearly all patterns can be used as black-box test strategies for mobile apps.

Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design. James A. Whittaker. Pearson Education, 2010.

Explains the concept of testing tours and provides test planning guidance.

Black-Box Testing. Boris Beizer, Wiley & Sons, 1995.

Comprehensive discussion of analysis for testing any software system using only external behavior.

Use Case Modeling. Kurt Bittner and Ian Spence. Pearson Education, 2003.

In-depth and practical guide to developing use case models.

Integrated Test Design and Automation. Hans Buwalda, Paul A. Watters, Iris Pinkster, and Dennis Janssen. Addison Wesley, 2001.

Explains how to identify small steps ("actions" in this course) and use them to create automated test suites.

A Practical Guide to Testing Wireless Smartphone Applications. Julian Harty. Morgan & Claypool Publishers, 2009.

Provides guidance automation of test suites on mobile devices. Shows how to use automation to better exercise permutations of real-world use.

Fuzzing: Brute Force Vulnerability Discovery. Michael Sutton, Adam Greene, Pedram Amini. Person Education, 2007.

Explains fuzzing approach to finding security bugs, mostly API testing. Some strategies applicable to user interface.

IEEE Standard 829-1998, IEEE Standard for Software Test Documentation. The Institute of Electrical and Electronics Engineers, Inc., 1998.

Provides a topical outline for documentation of test planning, test specification, and test reporting, along with basic guidance for writing and using a test plan.

Credits

Music

Ástor Piazzolla was an Argentine tango composer and bandoneón player who originated a unique musical form blending elements of Tango, Jazz, and classical music. The opening clip is from Tanguedia III and the closing clip is from Milonga Loca on *Tango: Zero Hour*, released on American Clave Records in 1993. Available from Amazon, iTunes, and other retailers.

Beta Testers

Molly Hooks and Devi Sadhu took the first version of this course and provided feedback and suggestions. Their cooperation is greatly appreciated.

Notices

Copyright

All course materials, including video lectures, test plan templates, and these notes are Copyright © 2011 Robert V. Binder, Chicago, Illinois. All rights reserved. No part of these course materials, including streaming video lectures and these notes may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without written permission of Robert V. Binder.

The Test Plan Template is provided under the GNU Public License.

Acknowledgments

Materials from *Testing Object-Oriented Systems: Models, Patterns, and Tools* (Addison-Wesley, 2000) used by permission of the author.

Trademark Acknowledgments

All terms mentioned in these course materials that are known to be trademarks or service marks designated as such. Use of a term in these materials should not be regarded as affecting the validity of any trademark or service mark.

Case Study Sources

Persons, organizations, places, and events portrayed in workshop case studies are fictitious or based on public information. Cases are prepared as the basis for class discussion rather than to illustrate either effective or ineffective handing of a technical problem or administrative situation.