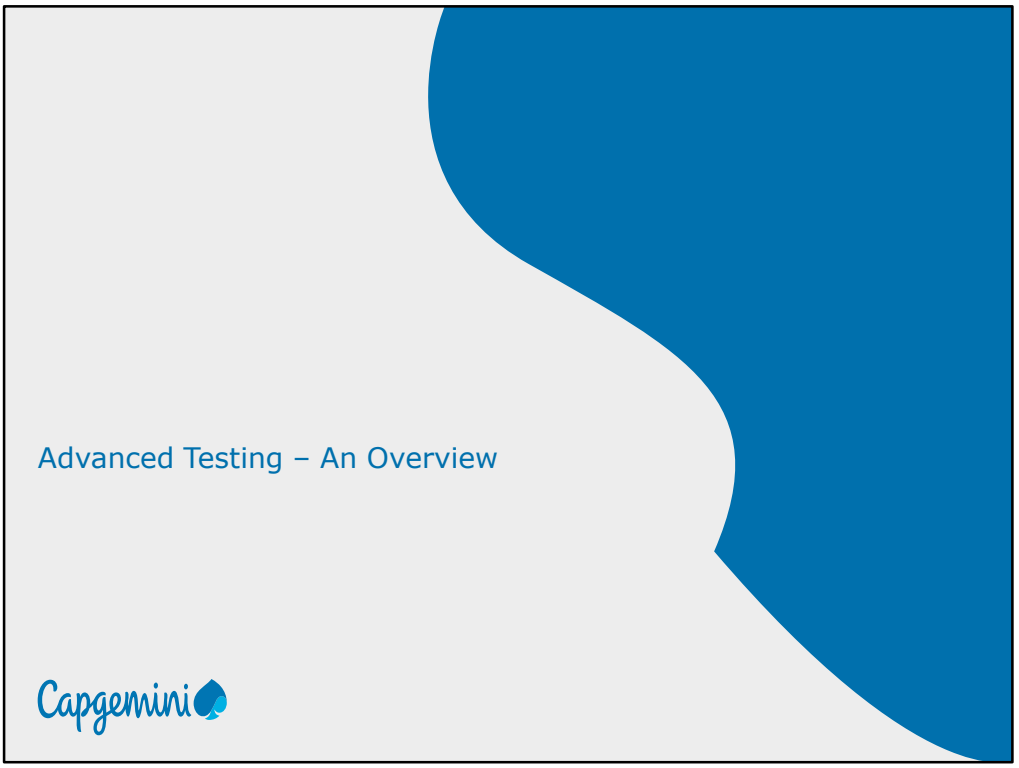
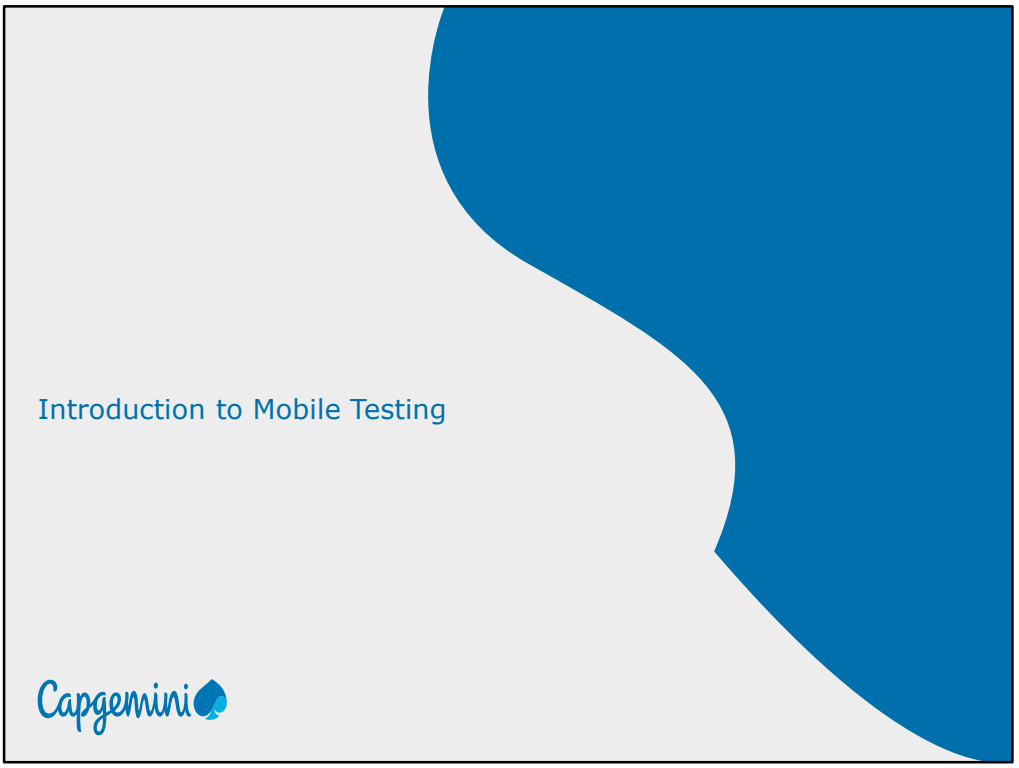

Advanced Testing – An Overview
Capgemini

Lesson Objectives

- Mobile Testing – An Overview
- Introduction to Mobile Platforms
- Types of Mobile Applications
- Overview of Mobile Application Testing
- Overview of Mobile Device Testing
- Overview of Mobile Automation Testing Framework
- Android Testing Frameworks
- iOS Testing Frameworks


Introduction to Mobile Testing
Capgemini

Mobile Testing – An Overview

- Mobile Technology is transforming the way people use their cell phones
- In the fast-growing world, the mobile phone is not only a device to make and receive telephone calls but also a multipurpose personal gadget
- Right from publishers, retailers, automobile dealers, financial service providers, pharmaceuticals & healthcare providers, the mobile application usage trend has now extended to schools and universities providing faster interaction between staff and students
- It is important to build an mobile app with all features and functionality required by the customer and which is beneficial to the app user, but it is even more critical to have a rigorous mobile testing plan before the mobile app is deployed

Introduction to Mobile Platforms

Operating System	Developed By	Latest available version
Android	Google Inc.	Lollipop, Android 5.0-5.1
iOS	Apple Inc.	iOS 8.X
Blackberry	Blackberry Ltd.	Blackberry 10.2.1
Windows	Microsoft Inc.	Windows 10 Mobile
Symbian	Symbian Foundation	Discontinued

Types of Mobile Applications

- Mobile Web : Web apps are not real applications but they are actually websites that open in your smartphone with the help of a web browser. Mobile websites have the broadest audience of all the primary types of applications.
  - E.g. http://www.google.co.in
- Native App : A native app is developed specifically for one platform. It can be installed through an application store such as Google Play Store or Apple's App Store
  - E.g. WhatsApp, Hangout
- Hybrid App : Hybrid Apps are a way to expose content from existing websites in App format. They can be well described as a mixture of Web App and Native App.
  - E.g. Instagram, Wikipedia

Overview of Mobile Application Testing

"Mobile application testing is a process by which an application software developed for handheld mobile devices is tested for its functionality, usability, and consistency. Mobile application testing can be automated or manual type of testing."

- Types of Mobile Application Testing
  - Functional Testing
  - Compatibility Testing
  - Localization Testing
  - Performance Testing
  - Memory Leakage Testing
  - Interrupt Testing
  - Installation Testing
  - Uninstallation Testing
  - Updates Testing
  - Usability Testing

Types of Mobile Application Testing :

1. Functional Testing : Functional testing is the most basic test for any application to ensure that it is working as per the defined requirements. Functional testing is aimed to ensure that it is working as per the defined requirements. Following are the examples of some test scenarios.

- Verify that movie ticket availability is displayed for a chosen theatre and on a selected date
- Verify that past dates are not included in the search results.

2. Compatibility Testing : Compatibility testing has got the highest stack when it comes to mobile application testing. The purpose of a mobile app compatibility test, in general, is to ensure an app's key functions behave as expected on a specific device. The compatibility itself should only take a few minutes, and can be planned well in advance. Following are the examples of some test scenarios.

- Verify that flight search is performed successfully with Android device.
- Verify that flight search is performed successfully for Apple iPad.

3. Localization Testing : Localization testing allows you to test mobile application adaptation for a specific target audience in accordance with its cultural specifics. Following are the examples of some test scenarios.

- Determine languages supported by the application.
- Ensure the correctness of the translation.
- Check the date formats.
- Check the delimiters in numbers.

Overview of Mobile Device Testing

- Network Connection Testing : This testing is performed to ensure expected functioning of mobile applications in different network configurations like GSM, TDMA and standards (2G, 3G, 4G) etc. This testing validates behavior of an application when out of network reach and recovery when device enters into network zone again.
- SD Card Interactions : This testing is performed to validate primary functions of the SD Card interaction with a mobile phone.
- Bluetooth Testing : Devices like keyboards, mouse, wireless headphone can communicate only within the radius of the 10 meters. This testing would be performed to ensure that user can search and connect to all available devices within the range and as per the security settings. This testing also validates various other primary functions of Bluetooth connectivity like successful data transmission, device pairing, authentication, security etc.

Overview of Mobile Device Testing (Cont.)

- Wi-Fi Testing : Testing your mobile phone WiFi connection is a great way to make sure your Internet is operating at the speed promised by your service provider, but you aren't limited to running those tests on your desktop computer. Mobile phone WiFi testing is a perfect way to test the power of your Wi-Fi signal at various places in your home or office.
- Concurrency Testing : We generally take the help of concurrent testing to make sure that multiple users can concurrently access a program at the same time. While applying concurrency testing for a mobile device, as such there will be only single user. So it eliminates the need of concurrency testing for a mobile device.

Overview of Mobile Automation Testing Framework

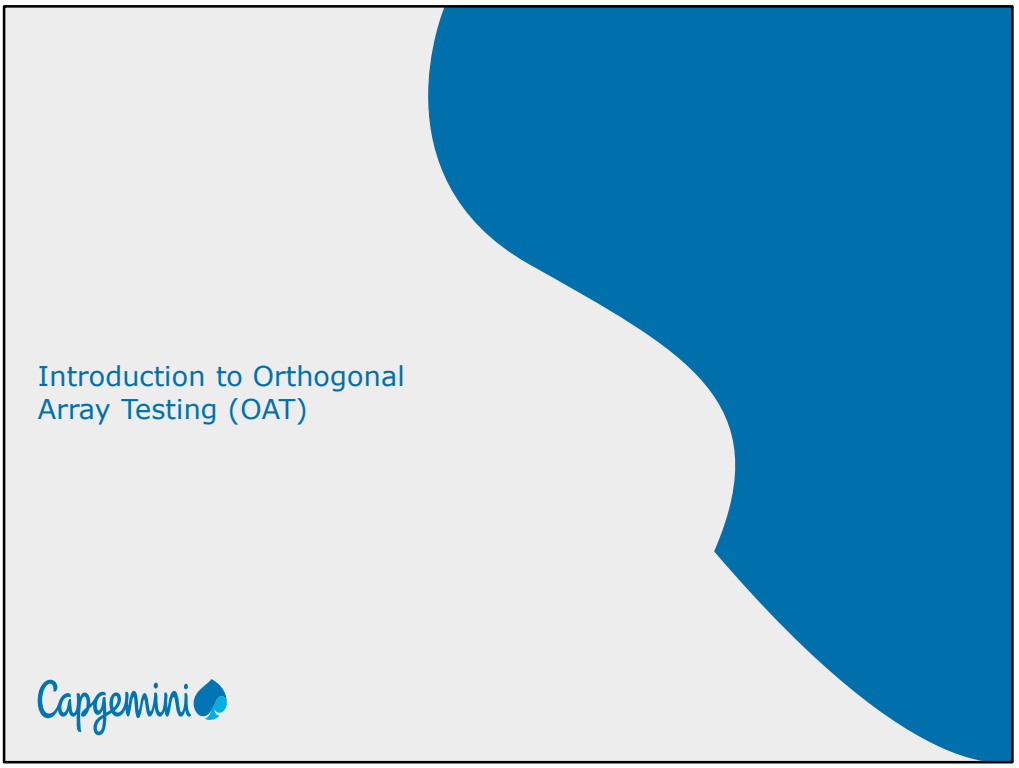
- For mobile testing automation, we need a good mobile automation testing framework
- On the top of that framework, we can build our test cases
- Mobile automation testing frameworks can be segregated by the operating system of the mobile device
- In the following chapters, we will discuss two types of mobile testing frameworks: Android testing frameworks and iOS testing frameworks

Android Testing Frameworks

- There are many Android testing frameworks available in the market
  - Robotium: Robotium is an open-source test framework for developing functional, system and acceptance test scenarios. It is very similar to Selenium.
  - UIAutomator: UIAutomator is a test framework by Google that provides advance UI testing of native Android apps and games. It has a Java library containing API to create functional UI tests and also an execution engine to run the tests.
  - Appium: Appium is an open-source test automation framework to test native and hybrid apps and mobile web apps. Appium library functions inside the framework make calls to the Appium server running in the background which operates the connected device.
  - Calabash: Calabash is a functional testing framework that can be used for both iOS and Android functional testing. On paper, it must be one of the easiest frameworks to use and even non-developers should be able to create functional tests using it.
  - Selendroid: Selendroid is a relatively new kid on the block and can be used to functionally test your Android applications. Apparently, if you are used to Selenium, Selendroid should be an easy way to use your knowledge to create your functional tests for Android.

iOS Testing Frameworks

- Like Android testing frameworks, there are many iOS testing frameworks available in the market
  - Appium
  - Calabash
  - Zucchini: Zucchini is an open-source visual functional testing framework for iOS applications based on Apple UIAutomation
  - UI Automation: For your more typical functional tests (or black-box tests), in which you're going to write code that simulates an end-user navigating your app, there is UI Automation. UI Automation is provided by Apple and is the Apple-sanctioned way of performing iOS functional testing.
  - FRANK – BDD for iOS: If you want to do end-to-end testing in iOS and wish you could use BDD and Cucumber, there's a tool called Frank that will allow you to create acceptance tests and requirements using Cucumber.


Introduction to Orthogonal
Array Testing (OAT)
Capgemini

Overview of Orthogonal Array Testing (OAT)

- These days the kind of software systems that are developed by project teams are very much complex in nature due to its code complexity
- In the conventional method, test suites include test cases that have been derived from all combination of input values and pre-conditions
- As a result n number of test cases has to be covered
- But, in reality, test engineers do not get that leisure time to execute all the test cases so as to uncover all the possible defects
- Hence, the test managers wanted to optimize the number and quality of the test cases to ensure maximum test coverage with minimum effort
- This effort is called Test Case Optimization
- One of the commonly used methods using which test optimization can be achieved is through implementing Orthogonal Array Testing (OAT)

What is Orthogonal Array Testing (OAT)?

- Orthogonal Array Testing (OAT) is one of the test case optimization techniques
- Orthogonal array testing is a systematic and statistical way of a black box testing technique used when number of inputs to the application under test is small but too complex for an exhaustive testing

- It is used as a statistical technique to generate the permutation of inputs, resulting in test cases with optimal test coverage to derive effort reduction in Test planning and Test design phase
  - Example: When a train ticket has to be verified, the factors such as - the number of passengers, ticket number, seat numbers and the train numbers has to be tested, which becomes difficult when a tester verifies input one by one. Hence, it will be more efficient when he combines more inputs together and does testing. Here, we can use the Orthogonal Array testing method.
  - This type of pairing or combining of inputs and testing the system to save time is called Pairwise testing. OATS technique is used for pairwise testing.

How OAT's is represented?

- OA's are commonly represented as:
  - Runs (N) – Number of rows in the array, which translates into a number of test cases that will be generated
  - Factors (K) – Number of columns in the array, which translates into a maximum number of variables that can be handled
  - Levels (V) – Maximum number of values that can be taken on any single factor
- A single factor has 2 to 3 inputs to be tested. That maximum number of inputs decide the Levels

How OAT's is represented? (Cont.)

- How to use this technique:
  1. Identify the independent variable for the scenario.
  2. Find the smallest array with the number of runs.
  3. Map the factors to the array.
  4. Choose the values for any "left over" levels.
  5. Transcribe the Runs into test cases, adding any particularly suspicious combinations that aren't generated.

Advantages of OAT

- Guarantees testing the pairwise combinations of all the selected factors
- The test set could be easily augmented if there were particularly suspicious three-and four- way combination that should be tested.
- Creates an efficient and concise test set
- Less error prone than test sets created manually
- Very effective in integration testing and regression testing
- Test design time and test execution time can be reduced

Limitations of OAT

- OATS does not guarantee 100% test coverage, it can only ensure optimal test coverage
- Scripts generated through OATS may have to be manually validated and additional test cases may have to be added to ensure maximum test coverage

OAT Tools

- Following are the set of some tools used to perform OAT
  - Hexawise
  - OATS
  - CATS
  - AETG
  - TConfig