Testing Cross-Platform Mobile App Development Frameworks

Nader Boushehrinejadmoradi, Vinod Ganapathy, Santosh Nagarakatte, Liviu Iftode

Department of Computer Science, Rutgers University

ASE 2015
The Growing App Market

• A steady growth on the amount of time users spend on mobile devices
• By targeting multiple platforms developers can reach a larger audience

Number of Apps in the Market as of July 2015

<table>
<thead>
<tr>
<th>Store</th>
<th>Number of Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Play</td>
<td>1600000</td>
</tr>
<tr>
<td>Apple App Store</td>
<td>1500000</td>
</tr>
<tr>
<td>Windows Phone Store</td>
<td>340000</td>
</tr>
</tbody>
</table>
App Developer Distribution

• There are a large number of mobile developers
  ▪ Many of these developers are small teams/individuals
• Most developers focus on one mobile platform
  ▪ Why is that?

Image source: http://blog.appfigures.com/most-app-developers-stick-with-one-store/
Conventional Cross-Platform Mobile Development

• Porting and maintaining code is costly
  ▪ Different languages
  ▪ Different APIs
  ▪ Different tools
Mobile App Development with Cross-Platform Frameworks

Host Language

Cross-Platform Framework

- C#
- JS
An Overview of Cross-Platform Frameworks

- Write code once, run on multiple platforms
  - Reduce development costs
  - Save development time
  - Reach a larger market
- Can be divided into 2 types
  - Hybrid
  - Native
An Overview of Cross-Platform Frameworks

• Write code once, run on multiple platforms
  ▪ Reduce development costs
  ▪ Save development time
  ▪ Reach a larger market

• Can be divided into 2 types
  ▪ Hybrid
  ▪ Native
Xamarin at a Glance

**Compilation**
- App source code
  - Written in C# with calls to Windows API
- Visual Studio
  - Native Windows App
- Xamarin
  - Native iOS App
    - Compatibility library
  - Native Android App
    - Compatibility library

**Runtime**
- Windows App
  - Windows SDK
- iOS App
  - iOS SDK
- Android App
  - Android SDK
Observation

• A growing demand exists for cross-platform frameworks
  ▪ Ease developing new apps
  ▪ Simplify porting of existing apps
• The framework has to correctly translate code to platforms that are different in many aspects
  ▪ Can inconsistencies occur in translation?
Example: Apps are not Always Consistent

- A call to `DateTime.Now.ToString()` is made to display current time
- Inconsistency in translating this method call to the target platform

https://bugzilla.xamarin.com/show_bug.cgi?id=23405
Insight

• By design, Cross-platform frameworks strive to provide **consistent behavior** across all targeted platforms

• Leverage this design assumption to automate bug finding
  ▪ Help developers to detect and fix bugs earlier
  ▪ Customers gets access to a more consistent framework
X-Checker: Cross-platform Test Tool
X-Checker

A tool that automatically tests cross-platform mobile frameworks:

- Generate randomized test cases to run on different mobile platforms
- Detect inconsistencies in framework by comparing test results in different platforms
Test Case Generator

• A Test case is a sequence of valid method calls
  ▪ Run generated sequences on one platform
  ▪ Iteratively extend valid code sequences
  ▪ Based on Randoop [Pacheco et al]
  ▪ Use all valid test cases
X-Checker Uses Differential Testing

• A Cross platform framework is a good candidate to apply differential testing [McKeeman]
  ▪ Differential testing requires that two or more comparable systems be available to the tester
  ▪ Present the systems with a series of generated test cases
  ▪ Observe the results. if they differ, we have found a bug-exposing test case

• Differential testing will not generate false positives
Testing Considerations

• Need to ensure applications run on target platforms
• Keep UI as simple as possible
• Ensure a unified initial state across platform at the start of the test
Detecting Inconsistencies

• Apps are run on their respective platform
  ▪ Test case either returns successfully or throws an exception
  ▪ Serialize test case state to detect inconsistencies across platforms

• The test case result can either be similar on all platforms or lead to one of three types of inconsistencies
Inconsistency Type I

```
public static int TestMain( FileIO serialStream, FileIO logStream){
    try{
        //Simple test case which runs to completion on both platforms
        Complex b = new Complex(0,0);
        Complex exp = new Complex(1,0);
        Complex res = Complex.Pow(b,exp);
        //Serialize objects
        serialStream.appendToLog(b.GetType.FullName, serializer.serialize(b));
        ...
        return 0;
    }
    catch (System.Exception e){
        //Not reached in this test case
        ...
        return -1;
    }
}
```

The state of the Android app and the Windows phone app are different:

- **Android**: res = 0
- **Windows Phone 8.1**: res = NAN
Inconsistency Type 2

public static int TestMain( FileIO serialStream, FileIO logStream) {
    try {
        //Simple test case which leads to different exception
        string s = "test";
        int index = -1;
        double val = CharUnicodeInfo.GetNumericValue(s, index);
        //Not reached in this test case
        ...
        return 0;
    } catch (System.Exception e) {
        //Each return different exceptions
        logStream(e.GetType().FullName);
        return -1;
    }
}

Windows Phone 8.1: ArgumentOutOfRangeException
Android: IndexOutOfRangeException
Inconsistency Type 3

public static int TestMain( FileIO serialStream, FileIO logStream){
    try{
        //Test case which leads to an exception on one platform
        NameTable nt1 = new NameTable();
        NameTable nt2 = new NameTable();
        XmlNamespaceManager nsMgr = new XmlNamespaceManager(nt2);
        ...
        XmlParserContext xpctxt = new XmlParserContext(nt1, nsMgr, ...);
        //Not reached on Windows Phone
        ...
        return 0;
    }
    catch (System.Exception e){
        //Each return different exceptions
        logStream(e.GetType().FullName);
        return -1;
    }
}

Windows Phone 8.1: XmlException
Android: No exception is thrown
Implementation

• Test case generation: *Randoop*

• Cross-platform framework: *Xamarin*

• Object serializer: *Json.net*
<table>
<thead>
<tr>
<th>Library</th>
<th>#Classes</th>
<th>#Methods</th>
<th>#Tests</th>
<th>#Inconsistencies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Type 1 Type 2 Type 3</td>
</tr>
<tr>
<td>Microsoft.CSharp.dll</td>
<td>6</td>
<td>56</td>
<td>1848</td>
<td>0 0 0</td>
</tr>
<tr>
<td>Microsoft.VisualBasic.dll</td>
<td>17</td>
<td>127</td>
<td>613</td>
<td>0 0 0</td>
</tr>
<tr>
<td>System.Collections.Concurrent.dll</td>
<td>10</td>
<td>77</td>
<td>349</td>
<td>0 0 0</td>
</tr>
<tr>
<td>System.Collections.dll</td>
<td>29</td>
<td>172</td>
<td>532</td>
<td>0 0 0</td>
</tr>
<tr>
<td>System.ComponentModel.dll</td>
<td>5</td>
<td>4</td>
<td>1578</td>
<td>0 0 0</td>
</tr>
<tr>
<td>System.Dynamic.Runtime.dll</td>
<td>29</td>
<td>201</td>
<td>790</td>
<td>1 0 0</td>
</tr>
<tr>
<td>System.Globalization.dll</td>
<td>14</td>
<td>288</td>
<td>567</td>
<td>3 3 0</td>
</tr>
<tr>
<td>System.Linq.dll</td>
<td>5</td>
<td>172</td>
<td>591</td>
<td>0 0 0</td>
</tr>
<tr>
<td>System.Linq.Expressions.dll</td>
<td>44</td>
<td>633</td>
<td>590</td>
<td>1 0 1</td>
</tr>
<tr>
<td>System.Net.Http.dll</td>
<td>44</td>
<td>524</td>
<td>746</td>
<td>3 0 3</td>
</tr>
<tr>
<td>System.Net.NetworkInformation.dll</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>System.Net.Primitives.dll</td>
<td>13</td>
<td>105</td>
<td>956</td>
<td>0 1 1</td>
</tr>
<tr>
<td>System.Net.Requests.dll</td>
<td>10</td>
<td>132</td>
<td>1169</td>
<td>0 0 0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>354</strong></td>
<td><strong>4758</strong></td>
<td><strong>22465</strong></td>
<td><strong>20 13 14</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Library</th>
<th>#Classes</th>
<th>#Methods</th>
<th>#Tests</th>
<th>#Inconsistencies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Type 1 Type 2 Type 3</td>
</tr>
<tr>
<td>System.Net.HttpListener.dll</td>
<td>24</td>
<td>346</td>
<td>820</td>
<td>2 3 3</td>
</tr>
<tr>
<td>System.Text.Encoding.dll</td>
<td>5</td>
<td>66</td>
<td>940</td>
<td>1 0 0</td>
</tr>
<tr>
<td>System.Text.RegularExpressions.dll</td>
<td>10</td>
<td>103</td>
<td>848</td>
<td>0 0 0</td>
</tr>
<tr>
<td>System.Xml.ReaderWriter.dll</td>
<td>24</td>
<td>346</td>
<td>820</td>
<td>2 3 3</td>
</tr>
<tr>
<td>System.Xml.XDocument.dll</td>
<td>23</td>
<td>637</td>
<td>612</td>
<td>0 1 1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>354</strong></td>
<td><strong>4758</strong></td>
<td><strong>22465</strong></td>
<td><strong>20 13 14</strong></td>
</tr>
</tbody>
</table>
Results

• Discovered 47 Unique inconsistencies
• 14 inconsistencies were fixed after being reported
  □ Most fixes were made within 1-2 weeks of the report
Discussion

• Duplicate inconsistencies and false positives
  - Test Cases with type 1 inconsistency can have duplicates
  - Flaky test cases introduce false postives
  - Filtering is used to remove duplicates and false positives

• Not all inconsistencies are bugs
  - Documented deviations of behavior
    • Object.GetHashCode()
    • System.UriBuilder Constructor(String)
  - Undocumented deviations known to developers
    • XmlDocumentReader.ReadContentAsString()
Summary

Conventional Cross-Platform Mobile Development

- Porting and maintaining code is costly
  - Different languages
  - Different APIs
  - Different tools

Inconsistency Type I

```java
public static int TestMain(String stream, String logstream){
  try{
    //Simple test case which runs to completion on both platforms
    Complex b = new Complex(0,1);
    Complex a = new Complex(1,0);
    //Serialize objects
    logstream.appendNum(0.0, (b.GetType().FullName, b));
    return 0;
  }
  catch (System.Exception e){
    //Not reached in this test case
    return -1;
  }
}
```

Windows Phone 8.1: res = 0
Android: res = NaN

X-Checker: Cross-platform Test Tool

naderb@cs.rutgers.edu
Backup Slides
Filtering

• Use filtering to eliminate problematic API calls
Structure of a Cross-platform Application

- Applications are broken down into 2 parts
  - Platform specific code
  - Shared Code