Mobile User Interface Development

Quincy Wofford
March 6, 2016

University of Kansas
How is mobile development done today?

Redundancy...not always a good thing

Bulky OR Expensive?
How is mobile development done today?

Redundancy...not always a good thing

Obj-C

@interface Foo : NSObject
@property (readonly) int bar;
- (instancetype)initWithBar:(int)bar;
+ (instancetype)fooWithBar:(int)bar;
@end

@implementation Foo
- (instancetype)initWithBar:(int)bar {
    self = [super init];
    if (self) {
        _bar = bar;
    }
    return self;
}

+ (instancetype)fooWithBar:(int)bar {
    return [[[self alloc] initWithBar:bar];
}
@end

scala

class Foo(bar : Int)

class Foo {
    private var bar;

    public Foo(int bar) {
        this.bar = bar;
    }

    public int getBar() {
        return bar;
    }
}

Bulky OR Expensive?
Bulky AND Expensive
What’s good about Java/Obj-C??

- Android’s Java/XML framework supports **24,000** unique devices [OpenSignal, 2015]
- Despite operating with more limited hardware options, IOS devices make up over 10% of the industry market share

<table>
<thead>
<tr>
<th>Period</th>
<th>Android</th>
<th>iOS</th>
<th>Windows Phone</th>
<th>BlackBerry OS</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015Q2</td>
<td>82.8%</td>
<td>13.9%</td>
<td>2.6%</td>
<td>0.3%</td>
<td>0.4%</td>
</tr>
<tr>
<td>2014Q2</td>
<td>84.8%</td>
<td>11.6%</td>
<td>2.5%</td>
<td>0.5%</td>
<td>0.7%</td>
</tr>
<tr>
<td>2013Q2</td>
<td>79.8%</td>
<td>12.9%</td>
<td>3.4%</td>
<td>2.8%</td>
<td>1.2%</td>
</tr>
<tr>
<td>2012Q2</td>
<td>69.3%</td>
<td>16.6%</td>
<td>3.1%</td>
<td>4.9%</td>
<td>6.1%</td>
</tr>
</tbody>
</table>

Source: IDC, Aug 2015
What’s Swift??

• A programming language released by Apple in 2014, entirely for mobile development
• Swift code and Obj-C code can be used in the same app
• Using Apples X-Code IDE, Swift has a real time compiler and visualization engine called a "Playground"
Why Swift???

- Free as in **freedom!** Apache 2.0 license
- Type-free, clojure’s and map functionality supported, function chaining
- Type extensions!
- Swift can generate LLVM-IR (intermediate representation). Something like bytecode for a virtual machine.
- The UI framework piece of Android (ie- XML hell), will be replaced by JSON!

---

### JSON

```
[
  {
    "xml": "sucks"
  }
]
```

### XML

```
<xml version="1.0">
<dict>
  <key>Label</key>
  <data>xml</data>
  <data>sucks</data>
</dict>
</xml>
```
The LLVM compiler enables many possibilities, developed by Chris Lattner

We are excited about this structure for the same reason JRM enjoys success, *high code reusability*. 
How is it possible? pt 2

Let’s make an adder!

**Step 0:** Write some Swift, compile it to LLVM bitcode...

```swift
// add.swift
func addTwoNumbers(first: UInt8, second: UInt8) -> UInt8 {
    return first + second
}

swiftc -emit-object addtwoNumbers.swift

nm add.o

    ...  
    _TF3add13addTwoNumbersFTVSSs5UInt86secondS0__S
    ...
```
Let’s make an adder!

**Step 1:** Android API’s need Java, use Java Native Interface (JNI) bridge

```c
// jni-bridge.c

// Let's work around Swift symbol mangling
#define SWIFT_ADD_TF3add13addTwoNumbersFTVSs5UInt86secondS0__S0_

uint8_t SWIFT_ADD(uint8_t a, uint8_t b);

jstring Java_com_example_helloswift_HelloSwift_stringFromJNI(JNIEnv * env, jobject thiz)
{
    uint8_t a = 123;
    uint8_t b = 45;
    uint8_t c = SWIFT_ADD(a, b);

    char result[255];
    sprintf(result, "We are computing the addition of %d and %d.\n\nThe result is %d. This computation was written in Swift and is running on this Android device!", a, b, c);

    return (*env)->NewStringUTF(env, result);
}
```
Let’s make an adder!

**Step 2**: Pack it into a shared library, and use it

```bash
cmake -DBUILD_SHARED_LIBS=ON
make

arm-linux-androideabi-ld \  
  add.o  
  jni_bridge.o  
  -shared # Build a shared library  
  -lc # We'll need the libc  
  -L$NDK/platforms/android-13/arch-arm/usr/lib

static {
  System.loadLibrary("add");
}
```
Q: Swift was developed with the Xcode IDE in mind. What about Android?
A: RemObjects Silver provides a cross-platform IDE, “Silver”
Swift with Android? That seems too good to be true...

Unfortunately, it is a little too good to be true.

- Android API’s are written in Java, and Cocoa (IOS) API’s are written in Obj-C.
- Cocoa Touch provides many features that make basic functions, like interacting with the user interface, very easy. Silver does not port Cocoa functionality to Android.
Swift with Android? That seems too good to be true...

Unfortunately, it is a little too good to be true.

• AndroidManifest.xml is not going away, (probably) ever.
• Multiple image resource directories aren’t going away.
• For these reasons, write once run anywhere capabilities are not guaranteed, yet.

```xml
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <LinearLayout
      android:layout_height="wrap_content"
      android:id="@id/linearlayout1"
      android:layout_width="wrap_content"
      android:layout_x="20dp"
      android:layout_y="16dp">
    <EditText
        android:text="15"
        android:id="@id/etSecond"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="phone" />
    <Button
        android:id="@id/btnStart"
        android:text="START"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
  </LinearLayout>
</AbsoluteLayout>
```
Best practice for Android and Swift.

It may be too early to write all your applications for Swift

• The Swift language was only very recently released (2014), and the open source community has only had access to the project source for 1 year.

• Playgrounds make working with IOS in XCode with Swift simple, but no such tool exists for Android and Swift.

• Creating library resources in Swift is a great idea, because as these two platforms converge, the Swift libraries you create will work with Android and IOS.
References for further reading and program writing

- Romain Pellerin’s GitHub has a side by side comparison of a particular IOS and Android app.

- RemObjects has some nice documentation on their website, and free downloads to get you started building cross platform tools

- Chris Lattner’s LLVM resource provides all the information you might be interested in for compiling LLVM source to arm processors for Android.