Cross-Compiling Android Applications to the iPhone

Wolfgang Korn
blueCarat AG, Cologne
wolfgang@xmlvm.org

blueCarat
Software & Consulting

The Team

- Core team:
  - Arno Puder, SFSU
  - Sascha Haeberling, Google Inc
  - Wolfgang Korn, blueCarat AG
- Contributors:
  - Kevin Glass
  - Panayotis Katsaloulis
  - Gergely Kis
  - Jessica Lee
  - Joshua Melcon
  - Alexander Seewald
- Sponsors:
  - blueCarat AG, Cologne, http://www.bluecarat.de

Background

- Smart phones have entered the mainstream.
- Most phones have similar capabilities.
- Common subset useful for certain class of applications (e.g. games).
- Developers are interested to push their applications to different platforms.
- Problem: Porting an application to different phones requires significant effort.

Proposed Solution

- Android is used as development platform:
  - Developer only needs to know Android SDK.
    - Make use of well-documented SDK.
    - Powerful tools (debugging).
    - Support for arbitrary screen resolutions using layout managers (Nexus One, iPhone, iPad, Netbooks).
  - Android application is cross-compiled to Objective-C or JavaScript.
  - Does not require a JVM on the iPhone!

- Approach for Android to iPhone cross-compilation:
  - Create Java API for Cocoa Touch.
  - Cross-compile Java to Objective-C.
  - Write Android compatibility library on top of Cocoa Touch Java API.

iPhone “Hello World” in Objective-C

```
@interface HelloWorld : UIApplication
-(void) applicationDidFinishLaunching: (NSNotification*) n;
@end

@implementation HelloWorld
-(void) applicationDidFinishLaunching: (NSNotification*) n
{
    UIScreen* screen = [UIScreen mainScreen];
    CGRect rect = [screen applicationFrame];
    UIWindow* window = [[UIWindow alloc] initWithFrame: rect];
    rect.origin.x = rect.origin.y = 0;
    UIView* mainView = [[UIView alloc] initWithFrame: rect];
    [window addSubview: mainView];
    UILabel *title = [[UILabel alloc] initWithFrame: rect];
    [title setText: @"Hello World!"];
    [title setTextAlignment: UITextAlignmentCenter];
    [mainView addSubview: title];
    [window makeKeyAndVisible];
}
@end
```

iPhone “Hello World” in Java

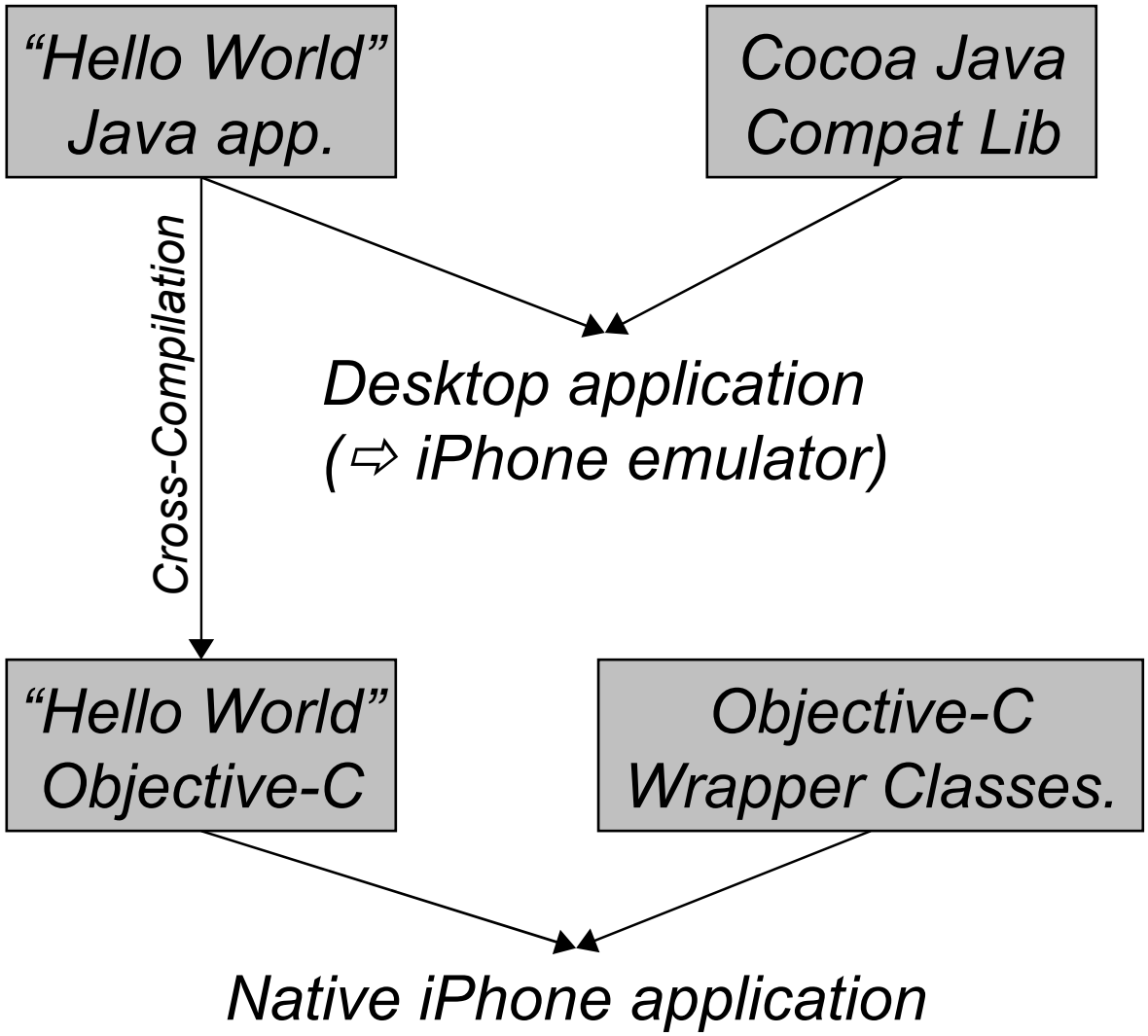
```
public class HelloWorld extends UIApplication
{
  public void applicationDidFinishLaunching(NSNotification n)
  {
    UIScreen screen = UIScreen.mainScreen();
    CGRect rect = screen.applicationFrame();
    UIWindow window = new UIWindow(rect);

    rect.origin.x = rect.origin.y = 0;
    UIView mainView = new UIView(rect);
    window.addSubview(mainView);

    UILabel title = new UILabel(rect);
    title.setText("Hello World!");
    title.setTextAlignment(
        UITextAlignment.UITextAlignmentCenter);
    mainView.addSubview(title);

    window.makeKeyAndVisible();
  }
}
```

Tool Chain Overview

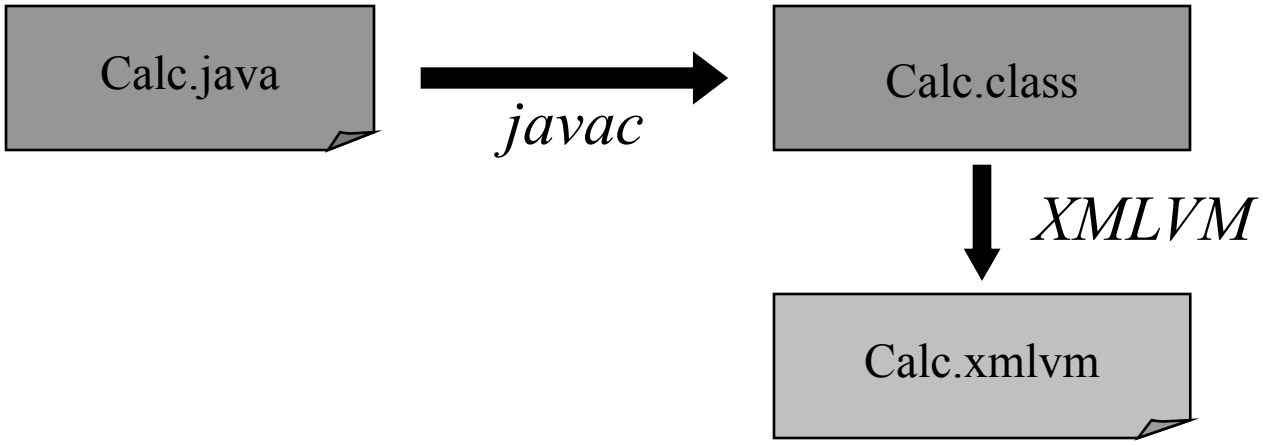


Challenges

- Design goals:
  - If possible, Java API mimics 1:1 the Objective-C API.
  - Make use of strongly typed interfaces.
- Objective-C challenges:
  - Namespaces, method overloading: use name-mangling.
  - Garbage collection: use Objective-C reference counting.
  - No static member variables: use global variables.
- Cocoa Touch challenges:
  - Makes use of C-functions (e.g., `CGColorCreate`)
  - Uses value types (e.g., `CGRect`)
  - Uses pointers for output parameters (e.g., `NSURLConnection`)
  - Makes heavy use of delegation (e.g., data source for `UITableView`)

Creating XMLVM

```
// Java
class Calc {
    static int add(int x, int y)
    {
        return x + y;
    }
}
```



XMLVM of Class Calc

```
<?xml version="1.0" encoding="UTF-8"?>
<xmlvm xmlns:dex="http://xmlvm.org/dex">
  <class name="Calc">
    <method name="add" isStatic="true">
      <signature>
        <parameter type="int" />
        <parameter type="int" />
        <return type="int" />
      </signature>
      <dex:code register-size="3">
        <dex:add-int vx="0" vy="1" vz="2" />
        <dex:return vx="0" vx-type="int" />
      </dex:code>
    </method>
  </class>
</xmlvm>
```

Cross-Compiling XMLVM to Objective-C

- Simply map DEX registers to variables of the target language!
- These translations are done using XSLT.
- Mappings exist for different languages.
- The XSLT excerpt below demonstrates the translation of `<dex:add-int/>` (Integer add) to Objective-C.

```
// Objective-C
typedef union {
    id      o;
    int     i;
    float   f;
    double d;
} XMLVMElem;

<!- XSL template -->
<xsl:template match="dex:add-int">
  <xsl:text>_r</xsl:text>
  <xsl:value-of select="@vx"/>
  <xsl:text>.i = _r</xsl:text>
  <xsl:value-of select="@vy"/>
  <xsl:text>.i + _r</xsl:text>
  <xsl:value-of select="@vz"/>
  <xsl:text>.i;</xsl:text>
</xsl:template>
```

```
_r0.i = _r1.i + _r2.i;
```

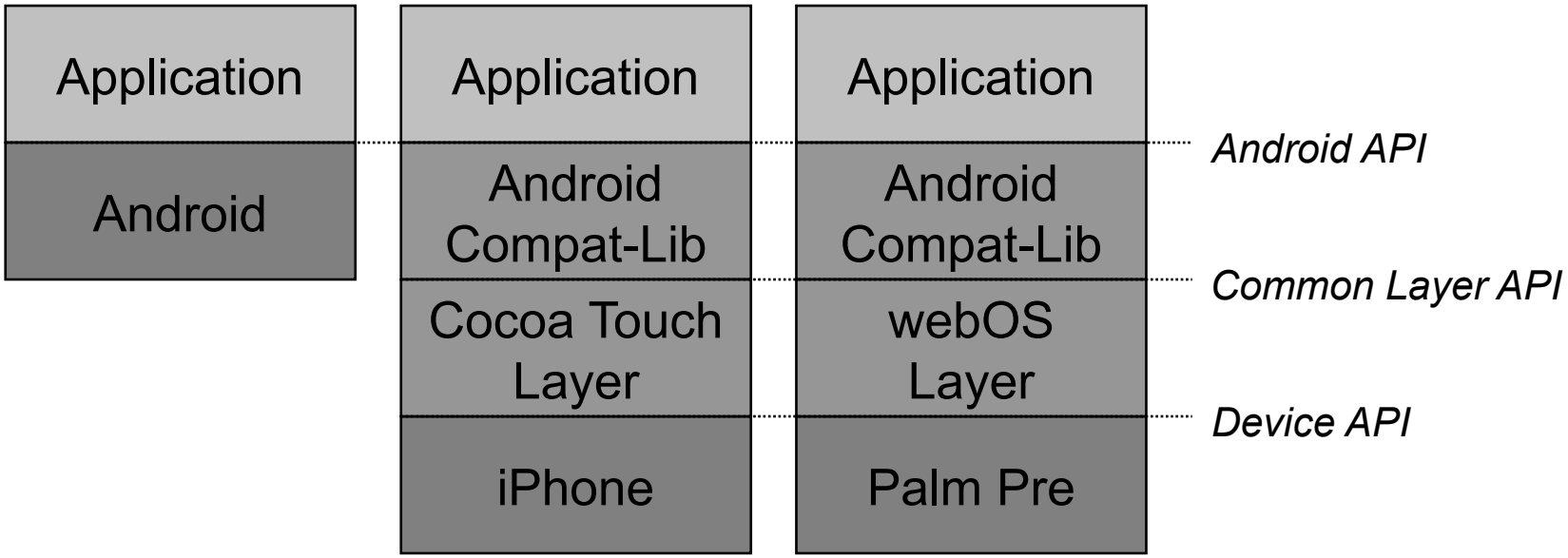
Class Calc in Objective-C

```
// Objective-C generated via stylesheet
@implementation Calc

+ (int) add___int_int :(int)n1 :(int)n2
{
    XMLVMElem _r0;
    XMLVMElem _r1;
    XMLVMElem _r2;
    _r1.i = n1;
    _r2.i = n2;
    _r0.i = _r1.i + _r2.i;
    return _r0.i;
}

@end
```

Android Compatibility Library



- Android Compatibility Library:
  - Offers Android API.
  - Device-independent portions of Android.
  - Builds upon a Common Layer API.
  - Written in Java.
  - Is cross-compiled to Objective-C.

Example 1: Button to UIButton Mapping

```
package android.widget;

public class Button extends View {
   protected UIButton button;

   // ...

   public void setOnClickListener(OnClickListener listener) {
        final OnClickListener theListener = listener;
        button.addTarget(new UIControlDelegate() {

            @Override
            public void raiseEvent() {
                theListener.onClick(Button.this);
            }

        }, UIControl.UIControlEventTouchUpInside);
   }
}
```

Example 2: R-class

```
// Generated by Android
public final class R {
public static final class drawable {
        public static final int ball=0x7f020001;
// ...
}
```

- Resources in Android are referenced through R-class:

```
ImageView image = …;
image.setImageResource(R.drawable.ball);
```

- Problem: map integer (0x7f020001) to file name (ball.png)
- Solution: use Java reflection at runtime for the reverse mapping.

```
String rClassName = "R$drawable";
Class<?> rClass = Class.forName(rClassName);
Field[]  fields = rClass.getDeclaredFields();

for (int i = 0; i < fields.length; i++) {
    String fieldName = fields[i].getName();
    int    id        = fields[i].getInt(rClass);
    // ...
}
```

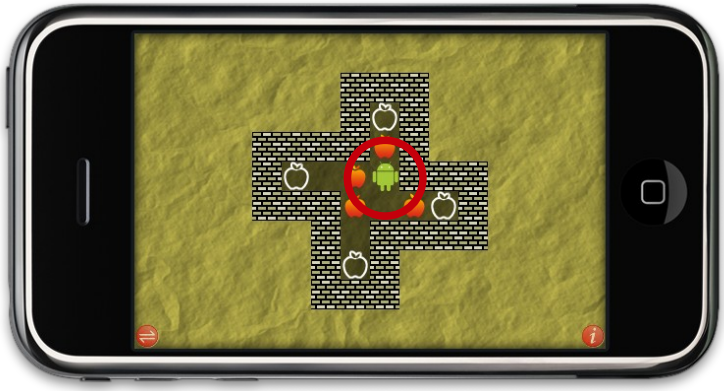
Example 3: Layout Manager

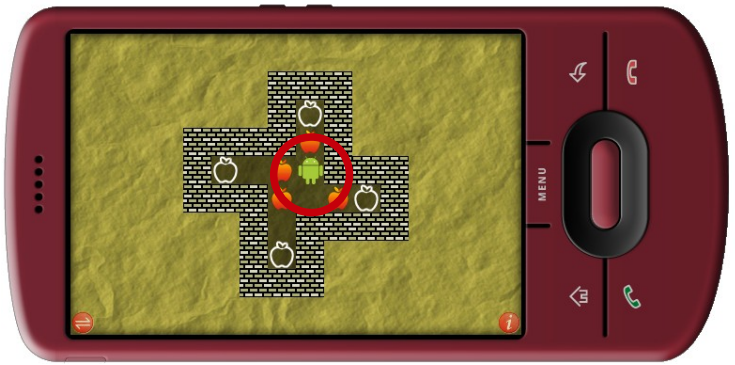
- Android uses declarative UI descriptions.
- UI descriptions are saved in XML files.
- Those files are copied to the iPhone.
- At runtime, XML files are parsed and UI is built on the iPhone.
- XMLVM cross-compiles Android's implementation of the layout managers.


```
<LinearLayout orientation="vertical”
              layout_width="fill_parent"
              layout_height="fill_parent">
    <EditText ... />
    <TextView ... />
    <Button layout_width="wrap_content"
            layout_height="wrap_content”
            id="@+id/sayHello"
            text="Say Hello" ... />
    <LinearLayout background="@drawable/xmlvm_logo" ... />
</LinearLayout>
```

Xokoban

- Sokoban version developed for Android and cross-compiled to the iPhone and webOS.
- Available in the Android Market and iPhone AppStore.
- Features:
  - 2D animation.
  - Dialog boxes.
  - Accelerometer and swipe interface.
  - Landscape full-screen mode.
  - Saving of settings.





Legions

FunkyMobileGames

- Implementation of the Stratego board game.
- Developed for Android.
- AI engine implemented by Vincent De Boer (3rd time Stratego world champion).
- Android version of Legions uses:
  - Activity lifecycle.
  - Sound.
  - Animation.
  - Custom dialogs/toasts.
  - Declarative layouts using different layout managers.
  - File I/O.

Licensing

- 70% of all Open Source projects are under GPL license.
- GPL is incompatible with commercial usage.
- Frequent requests for more liberal licenses (BSD, …)
- Problem: seldom contributions in return for more liberal licenses.
- Solution: dual licensing where the “commercial” license is a non-transferrable linking exception.

Conclusions

- Summary:
  - Java for the iPhone.
  - Android Compatibility Classes.
- Benefits:
  - Leverage existing Android/Java skills.
  - Only knowledge of a single platform is required.
  - Reduces development costs.
  - Shortens time-to-market.
- Commercial support at **http://xmlvm.org/support/**