Component Technologies on Google Android

Master Seminar

Michael Eckel

University of Applied Sciences Gießen-Friedberg
Department of Mathematics, Natural Sciences and Computing

January 7, 2011

ANDROID

Outline

Outline

What is Android?

- Mobile Operating System
- Developed by Google
- First version released in 2008
- Largely open-source
- Based on Linux 2.6
- Programming is done in Java

System Architecture

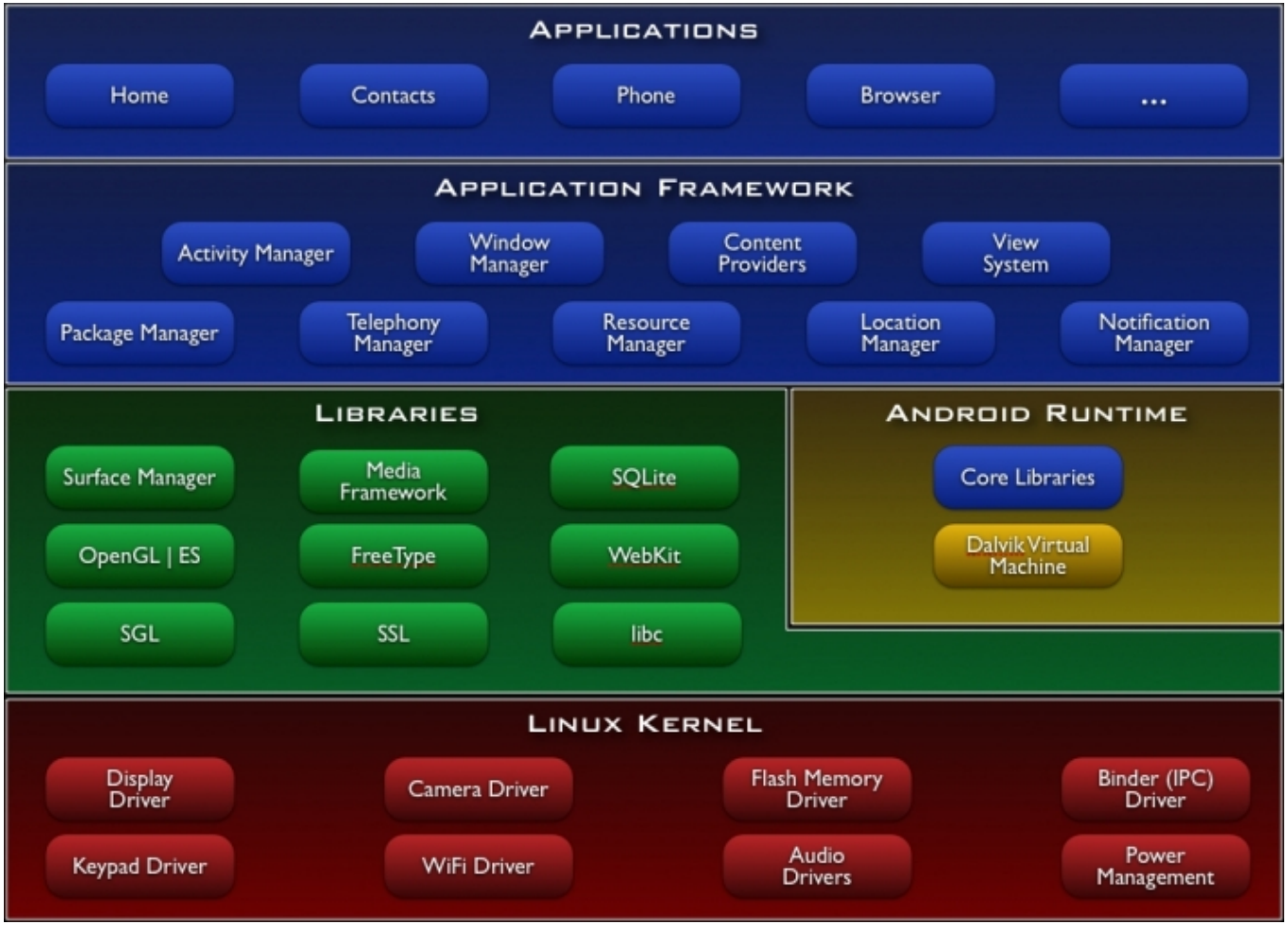


Figure: System architecture of Android

Source: `http://developer.android.com/images/system-architecture.jpg`

What is the Dalvik Virtual Machine (DVM)?

- Developed by Google
- Based on Apache Harmony
- Optimized for mobile devices
- Makes use of processor registers (unlike the JVM)
  - Register machine vs. Pushdown automaton
- Executes DEX bytecode (*Not Java bytecode!*)

DEX bytecode

- Much more compact than Java bytecode
- But does *not* support
  - all Java language features (e. g. *Reflection*)
  - the whole Java framework (e. g. Swing, AWT, . . .)

DEX bytecode

- Much more compact than Java bytecode
- But does *not* support
  - all Java language features (e. g. *Reflection*)
  - the whole Java framework (e. g. Swing, AWT, . . .)

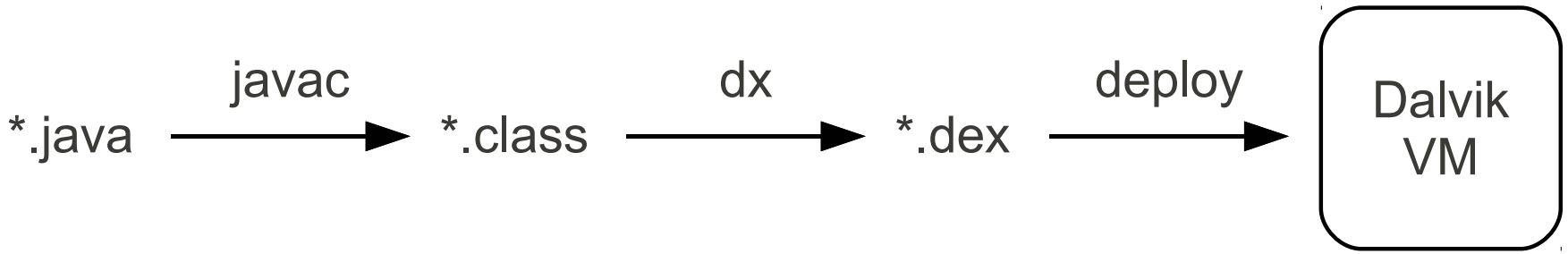


Figure: How to create DEX bytecode from Java sourcecode

Android Applications

- Android applications consist of
  - DEX bytecode
  - Ressources (e. g. images, sounds, . . .)
  - Data (e. g. SQLite databases, XML files, . . .)
- They are subject to a lifecycle
- . . . and are executed in a *sandbox*
- One Linux process per application
- One DVM per application
- Own UID and GID

Outline

Definition of a Software Component

Definition by the European Conference on Object-Oriented Programming (ECOOP) in 1996:

> *"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."*

Definition of a Software Component

Definition by the European Conference on Object-Oriented Programming (ECOOP) in 1996:

> *"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."*

In short:

1. Contractually specified interface
2. Explicit context dependencies
3. Independent deployment

What is a Component Model?

Components must conform to a component model!

A component model specifies the following:

- Form and properties of components
- How components interact with each other
- How components can be combined

Outline

Components in Android

- Activities
  - Interaction with the user
  - Presentation of the user interface
- Services
  - Used to run background tasks
  - They do not have an user interface
- Content Providers
  - Providing data to other applications
  - "Break out" from the sandbox
- Broadcast Receivers
  - Receive broadcasts (from Android system or other applications)

Outline

Communication between Components

- Is done via *Intents* and *Intent Filters*
- Intent announces wish for communication
- Intent Filter receives this wish
- Intent must match conditions of the Intent Filter

Intent Filters

- Are *explicitly* defined in the Android manifest (`AndroidManifest.xml`)
- That complies to point 1 of the definition of a component

```
...
<receiver android:name="org.example.MyReceiver">
  <intent-filter>
    <action android:name="org.example.TEST" />
  </intent-filter>
</receiver>
...
```

Intents

- Are *not explicitly* defined anywhere
  - Only in the Java code
  - (Maybe) violates point 2 of the definition of a component
  - It is up to the programmer to document the interface
  - Would be nice to see all dependencies explicitly
- Explicit vs. implicit Intents

```
/* explicit intent */
Intent ei = new Intent(org.example.MyReceiver.class);

/* implicit intent */
Intent ii = new Intent("org.example.TEST");
```

Sending an Intent

- Equipping additional data

```
Intent i = new Intent(Intent.ACTION_SENDTO);
i.setData(Uri.parse("mailto:you@mail.com"));
i.putExtra(Intent.EXTRA_SUBJECT, "Lottery");
i.putExtra(Intent.EXTRA_TEXT, "You won the Jackpot!");
startActivity(i);
```

- An appropriate application is being invoked

Outline

Realization of a Plugin System in Android

Involved Components

- Basic Application
- Plugins

Every component is realized as an own application!

Realization of a Plugin System in Android

Involved Components

- Basic Application
- Plugins

Every component is realized as an own application!

Communication

- **Basic Application**
  - Sends a broadcast on startup
  - Waits for answer from plugins
- **Plugins**
  - Wait for broadcast from basic application
  - Send answer broadcast to basic application

Basic Application

Request all plugins on startup...

```
...
public class BasicApplication extends Activity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    ...
    /* request all plugins */
    Intent i = new Intent("org.example.REQUEST_PLUGIN");
    sendBroadcast(i);
    ...
  }
  ...
}
...
```

Basic Application

Intent Filter for receiving response from plugins

```
...
<receiver android:name=".BasicApplicationResponseReceiver">
  <intent-filter>
    <action android:name="org.example.PLUGIN_RESPONSE" />
  </intent-filter>
</receiver>
...
```

Basic Application

Broadcast Receiver for receiving response from plugins

```
...
public class BasicApplicationResponseReceiver extends
     BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    /* process plugin answer */
    String packageName = intent.getStringExtra("package_name");
    ...
  }
}
...
```

Plugins

Intent Filter and Broadcast Receiver for plugins

```
...
<receiver android:name=".Plugin1RequestReceiver">
  <intent-filter>
    <action android:name="org.example.REQUEST_PLUGIN" />
  </intent-filter>
</receiver>
...
```

```
...
public class Plugin1RequestReceiver extends BroadcastReceiver {
@Override
  public void onReceive(Context context, Intent intent) {
    /* send broadcast response to basic application */
    Intent i = new Intent("org.example.PLUGIN_RESPONSE");
    i.putExtra("package_name", context.getPackageName());
    context.sendBroadcast(i);
  }
}
...
```

Android's Component System
Conclusion

- Plugin systems are easy to realize
- Plugins can be deployed independently
- So point 3 of the definition of a component is fulfilled

WebSMS

- Realizes a similiar plugin system
- Is open-source
- Text messages can be sent over online services
- Instead of the cellular phone network
- `https://github.com/felixb/websms/`

Outline

What is OSGi?

OSGi is a dynamic module system for Java and is specified by the OSGi Alliance.

> *"OSGi technology is Universal Middleware. OSGi technology provides a service-oriented, component-based environment for developers and offers standardized ways to manage the software lifecycle. These capabilities greatly increase the value of a wide range of computers and devices that use the Java™ platform."*

No further explanation will be given at this point.

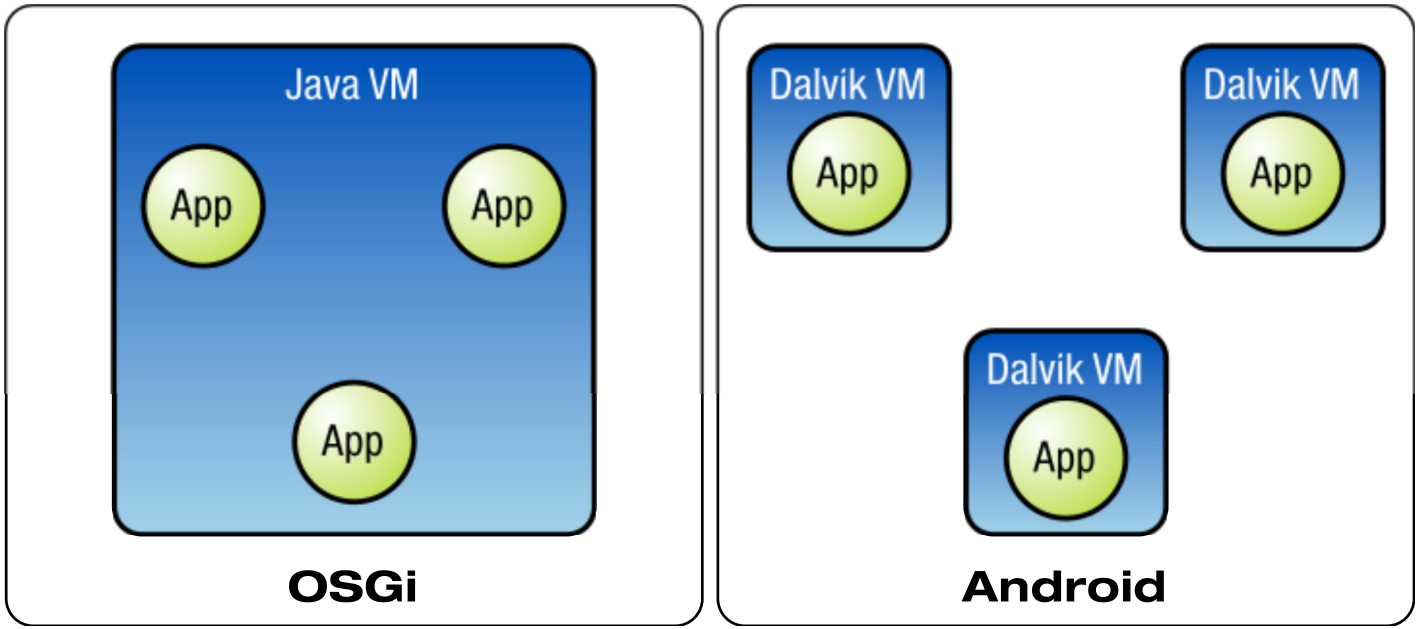
Why running OSGi on Android?

There are several advantages:

- OSGi specific features are needed
- Porting an existing OSGi application to Android
- Reusing an already existing OSGi component/bundle

Outline

OSGi vs. Android's Component System



- No need for IPC (overhead)
- On crash only one application crashes

Source: http://felix.apache.org/site/presentations.data/OSGi%20on%20Google%20Android%20using%20Apache%20Felix.pdf

Outline

OSGi Components on Android

OSGi frameworks for Android

- Apache Felix successfully ported to Android
- R-OSGi is also available

Running OSGi components on Android (I)

There are two simple steps to follow:

1. Create a normal OSGi bundle, e. g. `Dummy.jar` with following contents:
   - `META-INF/MANIFEST.MF`
   - `org/example/Dummy.class`
   - `org/example/impl/DummyImpl.class`

   The interface `org.example.Dummy` must be exposed in the manifest file
2. Convert the bundle to Dalvik format using *dx* tool:

   ```
   dx --dex --output=DummyDex.jar Dummy.jar
   ```

   The contents now should be as follows:
   - `META-INF/MANIFEST.MF`
   - `classes.dex`

Running OSGi components on Android (II)

R-OSGi needs exposed interfaces to be available as `.class` file:

- `META-INF/MANIFEST.MF`
- `classes.dex`
- `org/example/Dummy.class`

Now the JAR is ready for deployment

Outline

ROCS

What is ROCS?

ROCS (Remote OSGi Caching Service) is a remotely provisioned OSGi framework for ambient/mobile systems.

ROCS

What is ROCS?

ROCS (Remote OSGi Caching Service) is a remotely provisioned OSGi framework for ambient/mobile systems.

Why ROCS?

For providing OSGi bundles to mobile devices by directly loading them from network into the devices main memory.

ROCS

What is ROCS?

ROCS (Remote OSGi Caching Service) is a remotely provisioned OSGi framework for ambient/mobile systems.

Why ROCS?

For providing OSGi bundles to mobile devices by directly loading them from network into the devices main memory.

Advantages

- Administrators need to manage only one application repository
- Security constraints can be checked/enforced at a single point
- No need for installing software
- ...

Outline

ROCS on OSGi

- ROCS uses Java's remote class loading mechanisms
- Resources are directly loaded into memory
- Loading resources from network is similar to loading from a slow flash drive

Outline

OSGi Issues

- OSGi has one drawback:
  - Bundles are deployed into a local cache before they are loaded into the device's memory
- Normally done by using a local file system cache
- Cache stores all currently installed bundles
- Remote bundles are loaded as follows:
  - Bundle Repository $\rightarrow$ Device's Local Cache $\rightarrow$ Device's Main Memory

Outline

The ROCS Architecture

Consists of:

- Mobile devices (with ROCS OSGi frameworks)
- Remote cache servers (ROCS servers)

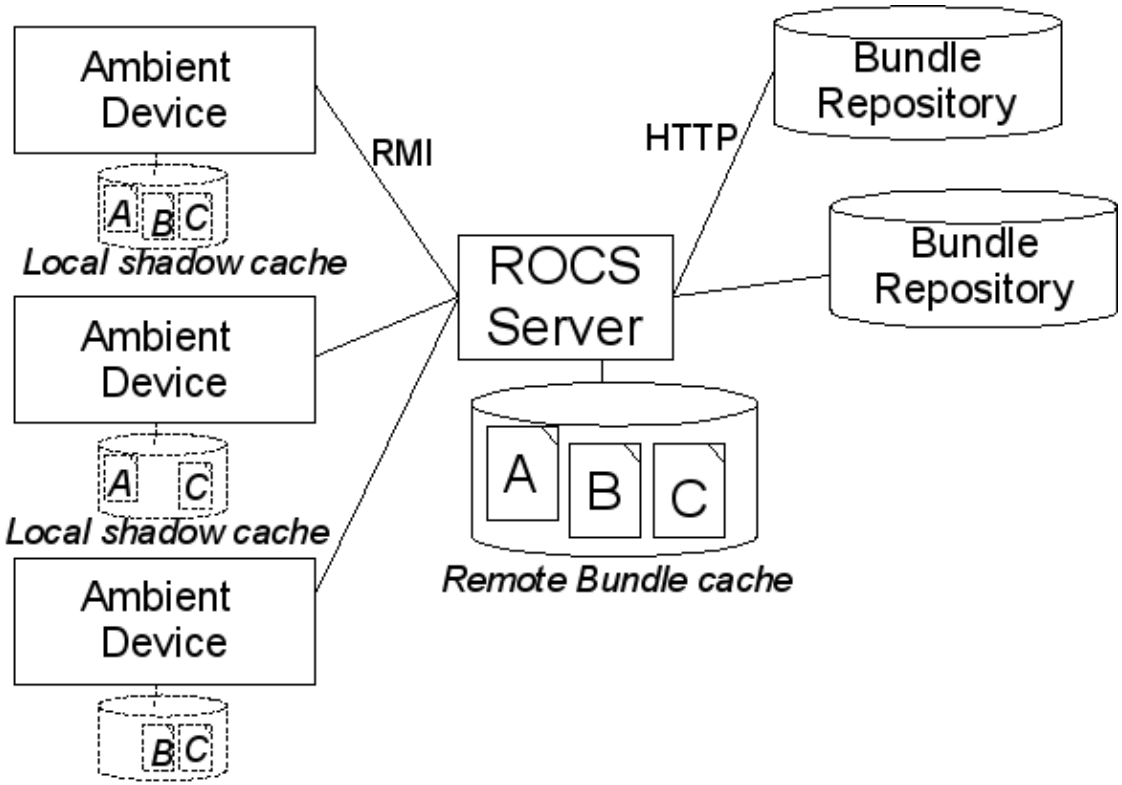


Figure: The ROCS Architecture [OP08]

ROCS Bundle Loading

- With ROCS bundles are loaded as follows:
  - Bundle Repository → ROCS Server's Local Cache → Device's Main Memory

Outline

Conclusion

Conclusion

1. **Android**
2. **Android's Component System**
   - Components in Android
   - Communication between Components
   - Realization of a Plugin System in Android

Conclusion

1. **Android**
2. **Android's Component System**
   - Components in Android
   - Communication between Components
   - Realization of a Plugin System in Android
3. **OSGi on Android**
   - Differences between OSGi and Android's Component System
   - OSGi Components on Android

Conclusion

1. **Android**
2. **Android's Component System**
   - Components in Android
   - Communication between Components
   - Realization of a Plugin System in Android
3. **OSGi on Android**
   - Differences between OSGi and Android's Component System
   - OSGi Components on Android
4. **ROCS: a Remotely Provisioned OSGi Framework**
   - ROCS on OSGi
   - OSGi Issues
   - The ROCS Solution

Conclusion

References I

[Bec] Felix Bechstein.
Sourcecode of WebSMS.
URL https://github.com/felixb/websms/.

[BP09] Arno Becker and Markus Pant.
Android – Grundlagen und Programmierung, March 2009.
URL http://www.androidbuch.de.

[Eur96] European Conference on Object-Oriented Programming (ECOOP).
Definition of a Software Component, 1996.
URL http://www.eecs.berkeley.edu/~newton/Classes/EE290sp99/lectures/ee290aSp994_1/tsld009.htm.
[Online; Accessed December 24, 2010].

[FIM+09] Stephane Frenot, Noha Ibrahim, Frederic Le Mouel, Amira Ben Hamida, Julien Ponge, Mathieu Chantrel, and Denis Beras.
ROCS: a Remotely Provisioned OSGi Framework for Ambient Systems, 2009.

References II

[Goo10a] Google Inc.
System Architecture of Android, December 2010.
URL http://developer.android.com/images/system-architecture.jpg.

[Goo10b] Google Inc.
The AndroidManifest.xml File, December 2010.
URL http://developer.android.com/guide/topics/manifest/manifest-element.html.
[Online; Accessed December 24, 2010].

[Goo10c] Google Inc.
What is Android?, December 2010.
URL http://developer.android.com/guide/basics/what-is-android.html.

References III

[Heu10] Stephan Heuser.
Entwicklung eines Frameworks zur sicheren mehrseitigen Aushandlung von Policies in Ambient-Intelligence Umgebungen.
Master's thesis, Technical University Darmstadt, April 2010.

[HKM10] Sayed Hashimi, Satya Komatineni, and Dave MacLean.
Pro Android 2.
Apress, March 2010.

[Lum07] Luminis.
Apache Felix on Google Android, November 2007.
URL http://lsd.luminis.nl/osgi-on-google-android-using-apache-felix/.

[OP08] Marcel Offermans and Karl Pauls.
OSGi on Google Android using Apache Felix, April 2008.
URL http://opensource.luminis.net.

References IV

[OSGa] OSGi Alliance.
OSGi Alliance Specifications.
URL http://www.osgi.org/Specifications/HomePage.
[Online; Accessed December 19, 2010].

[OSGb] OSGi Alliance.
OSGi Technology.
URL http://www.osgi.org/About/Technology.
[Online; Accessed December 20, 2010].

[Sch10a] Julian Schütte.
Felix OSGi on Android, 2010.
URL http://linkality.org/felix-osgi-on-android/.
[Online; Accessed November 24, 2010].

References V

[Sch10b] Julian Schütte.
R-OSGi on Android, 2010.
URL http://linkality.org/r-osgi-on-android/.
[Online; Accessed November 25, 2010].

[The] The Apache Software Foundation.
Apache Felix.
URL http://felix.apache.org/.

[Wik10] Wikipedia, The Free Encyclopedia.
Komponente (Software), 2010.
URL http://de.wikipedia.org/w/index.php?title=Komponente_(Software)&oldid=75243743.
[Online; Accessed December 24, 2010].

Thank you for listening!

Source: http://commons.wikimedia.org/wiki/File:Android_robot_skateboarding.svg

Questions?