Android is het eerste volledige, open en gratis mobiele platform ontwikkeld door Google. Android geeft concrete invulling aan een visie: het versnellen van innovatie in de mobiele markt en het aanbieden van een rijkere, goedkopere en betere mobiele gebruikerservaring. Het Android platform omvat een besturingssysteem, middleware en een aantal belangrijke toepassingen voor een mobiel apparaat. Dit, tezamen met het gratis en open karakter van het platform, biedt mobiele operators, mobiele hardwarefabrikanten en mobiele softwareontwikkelaars alles waarmee zij innovatieve apparaten, software en services kunnen bouwen. En het mooie is dat deze innovatieve software gebouwd zal worden met Java.

Android: Google op je telefoon

Tim Prijn
is Java ontwikkelaar bij Info Support BV in Veenendaal.

Anno 2008 zijn er anderhalf miljard televisies in omloop, één miljard mensen gebruiken internet en drie miljard mensen hebben een mobiele telefoon. Dit demonstreert hoe groot de markt voor de mobiele telefoon is. Google anticipeerde hier drie jaar geleden al op door op 28 juli 2005 het destijds net opgestarte bedrijf Android over te nemen. Dit bedrijf was bezig met de ontwikkeling van een mobiel platform, genaamd Android, dat voor iedereen gratis en volledig toegankelijk was. Een van de medeoprichters van dit bedrijf, Andy Rubin, is bij Google verantwoordelijk geworden voor het uitbouwen van dit platform.

Open Handset Alliance

Op het moment dat Google het bedrijf Android overnam, kwam de geruchtenstroom op gang dat Google tot de mobiele markt zou toetreden. De geruchten werden bevestigd toen op 5 november 2007 bekend werd dat de Open Handset Alliance (OHA) werd opgericht. Google was de initiatiefnemer van dit consortium dat inmiddels bestaat uit 34 deelnemers, die over de volgende vijf categorieën worden verdeeld: mobiele operators (T-Mobile, Telefónica), fabrikanten van mobiele apparaten (Samsung, Motorola), semiconductor bedrijven (Intel, Texas Instruments), softwarebedrijven (Google, Ebay) en tot slot de commerciële bedrijven (Wind river, Noser). Ieder lid van de OHA streeft naar meer openheid in software in de mobiele markt met als doel het sneller ontwikkelen van innovatieve software die de gebruikerservaring verbetert en verrijkt. Het begrip open wordt daarbij verklaard vanuit drie perspectieven:

- Mobiele industrie: hiermee worden de operators, apparaatfabrikanten en commerciële bedrijven bedoeld. Open betekent voor hen dat software waar mogelijk de Apache 2.0 licentie krijgt. Daar waar niet mogelijk zal GPL toegepast worden.
- Gebruikers: deze groep kan installeren wat hij/zij wenst. De functionaliteit omtrent het beheer van bijvoorbeeld contactpersonen kan vervangen worden. Verschillende applicaties die dezelfde functionaliteit aanbieden kunnen geïnstalleerd worden.
- Ontwikkelaars: er is geen toestemming nodig om een applicatie uit te rollen. Bestaande software kan geïntegreerd, uitgebreid of vervangen worden zonder hiervoor te moeten betalen.

Het eerste resultaat van de OHA was er op 22 oktober 2008 met de T-Mobile G1 mobiele telefoon. Inmiddels een zeer gewild gadget dat veel positieve recensies heeft gekregen.

Android Platform

Het Android platform bestaat uit een op Linux gebaseerd besturingssysteem, middleware en een aantal belangrijke applicaties specifiek bedoeld voor mobiele apparaten. Een paar features van dit platform zijn:

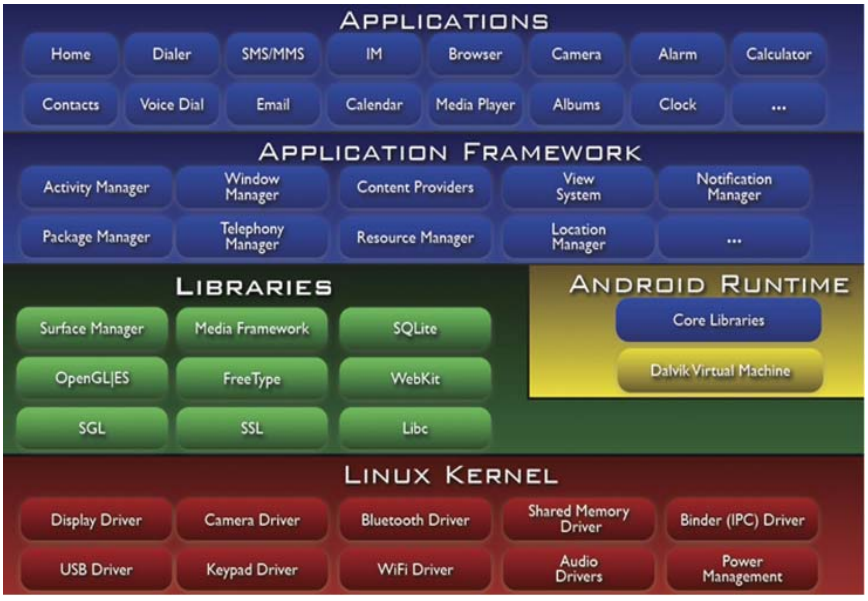
- Een ontwikkelomgeving inclusief emulator, debugging-, memory- en profiling tools en een Eclipse plugin.
- Een applicatieframework dat mogelijkheden

biedt tot integratie, uitbreiding en vervanging van componenten.
- De zogenaamde Dalvik virtual machine die is geoptimaliseerd voor mobiele apparaten.
- Een geïntegreerde browser gebaseerd op WebKit.
- Een aantal standaardapplicaties als e-mail-client, SMS, kalender, maps, browser, contactlijst.

De Android platformarchitectuur kan opgedeeld worden in verschillende lagen:

Linux Kernel

De basis van het Android platform is gebaseerd op de Linux Kernel. Het is dus geen Linux distributie. De eerste vraag die dan opkomt, is waarom men ervoor kiest veel extra werk op zich te nemen door een aparte distributie te onderhouden. De reden hiervoor is de beperkte hardware van een mobiele telefoon, wat met name tot uiting komt in de beperkte geheugencapaciteit, de beperkte CPU-snelheid en de gelimiteerde capaciteit van de batterij.
Google heeft voor de afgeleide Android Kernel zowel functionaliteit toegevoegd als weggelaten. Het bevat bijvoorbeeld geen native windowing systeem en geen glibc ondersteuning. De twee voornaamste toevoegingen betreffen een Binder IPC (inter process communicatie) driver en een agressiever power management systeem. De eerste biedt performancewinst voor het uitwisselen van objecten tussen processen doordat serialisatie niet meer nodig is. De power management wijziging is gericht op de gelimiteerde batterijcapaciteit, waardoor een proces specifieker kan aangeven welke randapparatuur wanneer benodigd is en wanneer niet.
De implementatie is open source.



Afbeelding 1. Android Platform architectuur.

Libraries

Deze laag biedt een groot aantal in C of C++ geschreven libraries aan voor lowlevel functionaliteit en rekenintensieve services. Denk hierbij aan libraries als WebKit (browser), Open Media Framework (afspelen audio, video) en SQLite (database).
Ook in deze laag zijn wijzigingen doorgevoerd om de software beter aan te laten sluiten op de beperkte capaciteit van mobiele hardware. Maar ook zijn aanpassingen gedaan, omdat een aantal bestaande standaard implementaties onder de GPL-licentie gedistribueerd worden, wat zou voorkomen dat softwarebedrijven die Android gebruiken geen closed source software kunnen toevoegen.
Om bovenstaande redenen is een eigen implementatie van libc geschreven, genaamd Bionic libc. Deze wordt gedistribueerd met de BSD-licentie waardoor bedrijven proprietary software kunnen toevoegen. En daarnaast bevat de Bionic libc optimalisaties ten aanzien van geheugengebruik (maar liefst de helft van standaard glibc) en performance.
Daarnaast heeft Google in deze laag een eigen Android HAL (Hardware Abstraction Layer) toegevoegd: een set van interfaces naar randapparatuur waaraan de drivers van hardwarefabrikanten moeten conformeren om vanuit een Android systeem gebruikt te kunnen worden. Denk hierbij aan drivers voor Bluetooth, USB, GPS maar ook de vibratie functionaliteit van een telefoon.

Android Runtime

De Android Runtime bestaat uit een Dalvik Virtual Machine met daarboven de core J2SE libraries, die gebaseerd is op de Apache Harmony open source Java implementatie. Ook hier is gekozen voor de extra kosten om de volgende redenen:
- Ten tijde van het nemen van deze beslissing was Java nog niet open source.
- Doorvoeren van optimalisaties t.o.v. de beperkte capaciteit van mobiele hardware.
- De mogelijkheid om meerdere programmeertalen te ondersteunen.

Ondanks dat de programmeur in Java ontwikkelt, executeert de Dalvik VM geen .class files maar .dex (Dalvik Executables) bestanden. Eerst worden de .java bestanden gecompileerd. Vervolgens worden de gegenereerde .class files omgezet naar .dex files die door de Dalvik VM ingeladen kunnen worden.
De .dex files zijn de helft kleiner dan .class files (en ongeveer even groot als gecomprimeerde .class files) doordat ze efficiënter opgeslagen worden. Omdat de .dex bestanden niet gecomprimeerd zijn, kunnen ze direct in en uit het geheugen worden geladen waardoor het een zeer efficiënt proces is. Ook zijn performanceverbe-

teringen doorgevoerd. Een voorbeeld hiervan is dat de .dex bestanden geschikt zijn om tijdens installatie in het Android systeem controles uit te voeren, zodat die runtime niet meer nodig zijn.

Application Framework

Deze laag in de platform architectuur is geschreven in Java. Het biedt Javaontwikkelaars een API waarmee Android applicaties ontwikkeld kunnen worden. Daarnaast biedt dit framework een aantal services, die verdeeld kunnen worden in de core platform services en de hardware services.
De platform services worden door een programmeur niet direct aangeroepen. Zo is het de verantwoordelijkheid van de Activity manager dat de gebruiker van een e-mailapplicatie naar een Android game kan wisselen. De window manager houdt bij welk scherm getoond moet worden. De resource manager zal op zijn beurt alle resources, zoals externalized strings en geïmporteerde bestanden, aan de applicatie beschikbaar stellen wanneer nodig. Tot slot is het View systeem verantwoordelijk voor het correct positioneren van componenten binnen een scherm.
De hardware services worden wel direct aangeroepen door de programmeur. Het betreft hier interfaces naar randapparatuur zoals de FM radio, GPS antenne, Bluetooth, WiFi en USB.

De bouwstenen

Een ontwikkelaar levert een zogenaamde APK (Android Package File) file op, die geïnstalleerd kan worden op het Android OS. Een APK file is niet veel meer dan een veredelde ZIP file die een aantal Android componenten bijeenhoudt. Daarbij executeert een APK file altijd in een apart Dalvik VM proces (één Linux proces) en delen de componenten binnen dit proces de resources als databases, preferences en bestandsruimte. Een van deze componenten is de Activity. Een Activity vervult een operationele taak die vaak voorzien is van een user interface. De Activity komt voor als klasse binnen het Application framework en heeft daarmee een afgebakende betekenis. Een ander begrip, de Task, is meer een design notie. Deze bestaat uit één of meerdere activities uit dezelfde of meerdere APK's en vormt een Android applicatie.
Een Activity is het object waarmee de developer het meest communiceert en die tevens de meest uitgebreide lifecycle heeft, zie afbeelding 2.
In het algemeen zijn drie categorieën te onderkennen: de opstartfase, tijdens executie en voor het afsluiten van de Activity. Met name tijdens executie zijn er een aantal callbackmethodes beschikbaar, zodat je als ontwikkelaar de juiste acties kunt ondernemen wanneer er bijvoorbeeld een telefoongesprek tussendoor komt.
Componenten tussen processen (uit verschillende APK's) hebben geen directe toegang tot elkaar of elkaars data. Daarvoor biedt het Android framework twee oplossingen middels IntentReceivers en ContentProviders.

Met IntentReceivers kunnen Activities beschrijven welk soort Intents zij kunnen afhandelen. Een voorbeeld van een Intent is "PLAY AUDIO". Voor de email client in een APK Package file zal de tweede Activity aan het systeem aangeven, middels de Intent "PICK PHOTO", dat het een foto nodig heeft. Het systeem zal dit Intent dan matchen op de derde Activity. Dit mechanisme bevordert ten eerste hergebruik van bestaande functionaliteit. Ten tweede maakt het vervanging mogelijk wanneer er bijvoorbeeld twee applicaties zijn die dezelfde Intent af kunnen handelen. Indien er twee applicaties zijn die dezelfde functionaliteit aanbieden dan moet de gebruiker aangeven welke gebruikt zal worden.
Android voorziet in een aantal standaard Intents. Door innovatie en ontwikkeling van nieuwe applicaties kan het zijn dat er nieuwe Intents bedacht gaan worden. Hoe en wanneer deze dan toegevoegd worden aan het framework is nog onduidelijk.

Codefragment 1 toont een Activity die de gebruiker een scherm toont met slechts één knop daarop. Als op de knop wordt gedrukt, dan wordt het intent 'View Contacts' afgevuurd waarop het systeem de contacten lijst zal tonen.

Met ContentProviders kunnen componenten in verschillende processen toch data delen. Stel dat vanuit de e-mail client een lijst getoond (en eventueel bewerkt) moet kunnen worden met alle namen van personen die in de contactenlijst staan. In dit geval zal de contacts APK een ContentProvider aanbieden die het mogelijk maakt voor andere activities om een Cursor op te

```
public class ViewContacts extends
   Activity implements OnClickListener {

  @Override
  public void onCreate(Bundle
                    savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button push =
     (Button)this.findViewById(R.id.ok);
    push.setOnClickListener(this);
  }

  @Override
  public void onClick(View v) {
    Intent viewContactsIntent = new
    Intent(Intent.ACTION_VIEW,
           People.CONTENT_URI);
    this.startActivity(
               viewContactsIntent);
  }
}
```

Codefragment 1. Activity en Intent

vragen om daarmee de contacts data te kunnen queryen. Daarnaast biedt de ContentProvider een API voor het toevoegen, wijzigen en verwijderen van data.

Codefragment 2 toont aan hoe de Cursor te verkrijgen en te gebruiken is.

```
String[] columnsInterestedIn = new
   String[]{ People._ID,
             People.NAME,
           };
Uri contactsUri = People.CONTENT_URI;
Cursor managedCursor =
  this.managedQuery(contactsUri,
                    columnsInterestedIn,
                    null, null,
                    People.NAME);

if (cursor.moveToFirst()) {
  String name;
  int nameColumn =
     cursor.getColumnIndex(People.NAME);

  do {
    name = cursor.getString(nameColumn);
    System.out.println("Log: " + name);
  } while (cursor.moveToNext());
}
```

Codefragment 2. Toegang tot een ContentProvider.

De variabele 'columnsInterestedIn' geeft weer in welke kolommen de aanroepende partij geïnteresseerd is. Verder biedt de Activity managed Query methode de mogelijkheid om een 'Where' clausule op te nemen (in codefragment 2 is dit 'null') en te sorteren op een bepaalde kolom. Zoals te zien is, maakt Android de toegang tot de URI's toegankelijker door een aantal Helper klassen (People). Net zoals voor Intents geldt hier dat het framework een aantal URI's voordefinieert. Hoe dit zich in de toekomst gaat ontwikkelen met betrekking tot nieuw bedachte URI's en hoe deze beheerd gaan worden, is net als voor intents nog onduidelijk.

Zoals eerder aangegeven hebben Activities veelal een bijbehorende User Interface. Android biedt twee manieren om UI te realiseren: vanuit de code of vanuit XML. Deze laatste optie biedt als voordeel dat er geen codewijzigingen nodig zijn om UI aanpassingen te doen. Codefragment 3 toont de benodigde XML om de in afbeelding zes getoonde UI te realiseren.

In deze configuratie zijn twee visuele componenten gedefinieerd: een Button en een TextView. Daarnaast is gespecificeerd hoe deze componenten ten aanzien van elkaar worden gepositioneerd, in dit geval door een LinearLayout. Dit is het basisprincipe achter het opstellen van ieder scherm in Android en op de syntax na zal dit bekend zijn voor iedere UI ontwikkelaar.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android=http://schemas.android.
com/apk/res/android"

  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  android:paddingLeft="6dip"
  android:paddingRight="6dip"
  android:paddingBottom="3dip">

  <Button android:id="@+id/ok"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="left"
    android:text="@string/button_ok" />

  <TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:autoText="true"
    android:capitalize="sentences"/>
</LinearLayout>
```

Codefragment 3. Android UI in XML.

Wat wellicht verder nog opvalt zijn verwijzingen als '@string/button_ok'. Dit is een verwijzing naar een resource file en wordt door het Android systeem op deze manier ingevuld.

Naast Activities biedt het framework ook nog het concept Services. Deze hebben geen UI en worden voornamelijk gebruikt voor het executeren van taken in de achtergrond. Het afspelen van een MP3 bijvoorbeeld zal in een service gebeuren, terwijl de selectie van de MP3 gebeurd is in een Activity. Waar Activities standaard executeren op de hoofdthread van het proces, worden services, die ook door andere componenten kunnen worden aangeroepen, uitgevoerd in een eigen thread.

Applicaties

De laatste laag van het architectuuroverzicht uit afbeelding 1 bestaat uit de applicaties zelf zoals de contactenlijst, een browser, een MP3 speler. De basisset aan applicaties kan uitgebreid, vervangen of hergebruikt worden. Zo kan een gebruiker bijvoorbeeld zijn beheer contactenlijst applicatie vervangen door een andere die meer bij zijn wensen aansluit. Dit is iets dat tot op heden niet mogelijk was. Daarnaast kan een softwareontwikkelaar de bestaande contactlijst software uitbreiden, wijzigen of bepaalde functionaliteiten hergebruiken.

Google en Sun

Google geeft de Java wereld op mobiel gebied een boost door de API voor zijn platform te baseren op Java. Maar dit is niet dezelfde Java-implementatie die al jaren gebruikt wordt voor het ontwikkelen van Java-applicaties voor een mobiel apparaat.

Ten eerste executeert er geen Java Virtual Machine maar een Dalvik Virtual Machine. Een van de voordelen, naast een betere performance en efficiënter geheugengebruik, is dat de Dalvik Virtual Machine

gedistribueerd wordt onder een licentie die toestaat dat software vendors hun eigen code proprietary mogen houden. Dit in tegenstelling tot de J2ME runtime, waarvoor een commerciële licentie gekocht moet worden. Android zal dan ook een einde maken aan de monopolie positie van Sun om Java te gebruiken voor mobiele software.

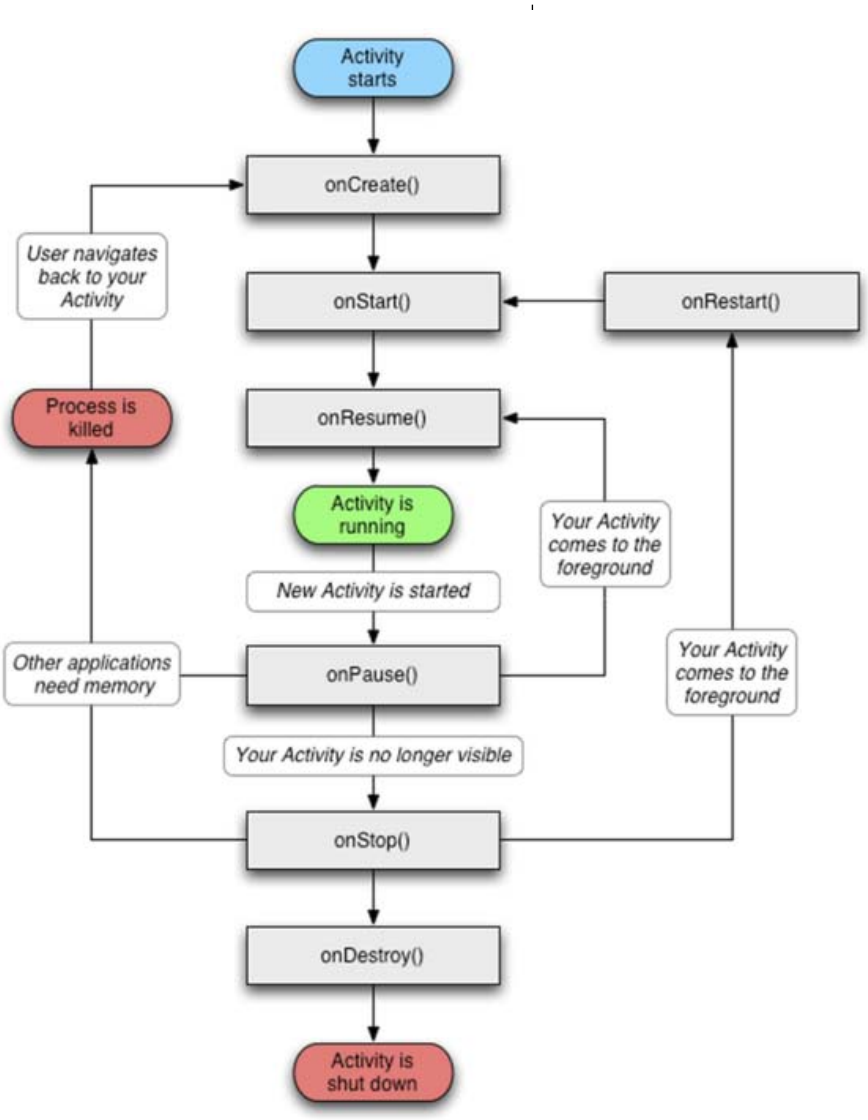
Het tweede verschil ten aanzien van de gebruikte Javaversie is de implementatie van Java. Google heeft zijn Java-implementatie gebaseerd op Apache Harmony, die buiten het Java Community Process (JCP) valt. Volgens Google sluit dit beter aan op het open karakter, omdat iedereen in staat is snel aanpassingen door te voeren. Vooralsnog heeft Google geen eenduidig antwoord op de vraag hoe het omgaat met eventuele versplintering en versies die niet compatibel zijn met elkaar. Iets wat met de JCP juist geregeld wordt. Google biedt standaard in het platform geen oplossing om bestaande J2ME-applicaties te executeren. Dit is echter opgepakt door de community en heeft geresulteerd in een Android applicatie waarbinnen J2ME Midlets uitgevoerd kunnen worden.

Conclusie

In plaats van voortborduren op bestaande ideeën omtrent mobiele besturingssystemen is Google met een clean sheet begonnen. Het heeft daarbij het Android platform gerealiseerd voor mobiele telefoons, waarbij twee keuzes centraal staan. Ten eerste is alle platformcode voorzien van een licentie die het mogelijk maakt om de bestaande code naar eigen wens aan te passen en open- of closed source verder te distribueren. Ten tweede zijn er optimalisaties doorgevoerd binnen ieder component van het platform wat betere performance, minder geheugengebruik en een langere batterijduur als gevolg heeft.

De 1.0 release van de Android SDK is al ver doorontwikkeld. Dit komt doordat Google eind 2007 al een versie beschikbaar heeft gesteld aan de community om zo vroeg mogelijk feedback te krijgen. Het bedrijf heeft tevens het gebruik van de SDK gestimuleerd met de Android challenge waarbij 10 miljoen dollar aan prijzengeld te verdelen was over twee rondes. De uitdaging was om een applicatie te ontwikkelen die innovatief is en de mogelijkheden van het platform demonstreert. Naast de feedback en publiciteit onder ontwikkelaars resulteerde deze challenge in een aantal zeer vooruitstrevende Android applicaties.

Doordat het platform open source is kan er fragmentatie ontstaan en kan het zelfs leiden tot versies die incompatibel zijn met elkaar. Google heeft aangegeven in de toekomst een compatibiliteitskit aan te bieden om te bepalen of in de afgeleide versie alles nog naar behoren werkt. Google is nu in ieder geval officieel toegetreden tot de mobiele markt met een technologisch vooruitstrevend platform. Het wordt goed ontvangen door de media, Javaontwikkelaars en de mobiele industrie. Ook de eerste telefoon, de T-Mobile G1, was snel na de release niet meer leverbaar. Wie een bijdrage wil leveren aan het succes van het platform of meer gedetailleerde informatie en codevoorbeelden zoekt, kan de Android SDK downloaden vanaf http://code.google.com/android/. «



Afbeelding 2. Lifecycle van een Activity, http://code.google.com/android/reference/android/app/Activity.html.

Referenties

http://www.openhandsetalliance.com/oha_members.html
http://git.android.com