QT-SOVELLUKSEN UUDELLEENKIRJOITTAMINEN ANDROID-KÄYTTÖJÄRJESTELMÄLLE

Joonas Lehtonen

Opinnäytetyö
Toukokuu 2012
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

LEHTONEN, JOONAS:
Qt-sovelluksen uudelleenkirjoittaminen Android-käyttöjärjestelmälle

Opinäytetyö 39 sivua
Toukokuu 2012

Tässä opinnäytetyössä selvitettiin, millaisia haasteita ja mahdollisuuksia Android-alusta tarjoaa Qt-sovelluskehittäjälle. Näkokulmaksi otettiin tilanne, jossa olemassa oleva Symbianille tehty Qt-sovellus kirjoitetaan alusta alkaen uudelleen Androidille.

Qt on sovelluskehyksenä menettänyt merkitystään sen jälkeen, kun Nokia ilmoitti aloittaneensa yhteistyön Microsoftin kanssa ja samalla hylkäävänsä hiljalleen Symbian-alustan. Samaan aikaan Android on parissa vuodessa kasvanut merkittäväksi, ellei merkittävimmäksi, älypuhelinalustaksi. Näin ollen panostuksen siirtäminen Symbianista Androidiin on looginen askel sovelluskehittäjille.

Työssä käydään läpi Android-sovelluskehityksen merkittävimpiä eroavaisuuksia verrattuna Qt-sovelluskehitykseen sekä Android-alustan uniikkeja piirteitä. Tämän lisäksi tarkastellaan, millaisia Android-laitteet ovat fyysisiltä ominaisuuksiltaan ja miten tämä vaikuttaa sovelluskehitykseen. Lopuksi käydään lävitse Android-kehityksen ongelmia ja haasteita.

Asiasanat: Qt, Android, sovelluskehitys, uudelleenkirjoittaminen.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

LEHTONEN, JOONAS:
Rewriting a Qt Application for Android Operating System

Bachelor's thesis 39 pages
May 2012

This thesis explores application development on Android platform. This is done from viewpoint of a developer who has previously done an application for Symbian platform and wants to rewrite this same application for Android starting from scratch.

Android has quickly risen to be one of the most important if not the most important mobile platform. At the same time Symbian has lost its market position due to Nokia's decision to partner with Microsoft and to start slowly pulling support from Symbian. Switching from Symbian to Android feels like a logical action for developers based on these recent events.

This thesis focuses on the most significant differences between these two platforms and the unique features of Android as a development platform. Different physical devices running Android are also given a quick look. Finally some of the major problems and challenges in Android development are listed.

Key words: Qt, Android, application development, rewriting.

SISÄLTÖ

LYHENTEET JA TERMIT

Android	Mobiililaitteille suunnattu ohjelmistopino, joka sisältää käyttöjärjestelmän, väliohjelmiston ja tarvittavat perussovellukset
Apache Harmony	Ilmainen, nykyään jo lakkautettu Java-toteutus
Apache v2	Apache Software Foundationin luoma vapaan ohjelmiston lisenssi
C++	C-kieleen pohjautuva, yleisesti käytetty ohjelmointikieli
CSS	Cascading Style Sheets; tyylimäärittelyspesifikaatio
I/O	Input/Output; tiedon siirtäminen, tässä yhteydessä tiedon siirto laitteen sisäisestä tai ulkoisesta lähteestä laitteen RAM-muistiin
iOS	Ohjelmistoyritys Applen kehittämä mobiililaitteiden käyttöjärjestelmä
Java	C-sukuinen, virtuaalikoneessa ajettava oliopohjainen ohjelmointikieli
Java SE	Java Standard Edition; työasemaympäristöihin suunnattu versio Javasta
JNI	Java Native Interface; ohjelmistokehys, jonka avulla Java-sovellukset voivat kutsua muilla kielillä tehtyjä kirjastoja
Lua	Kevyt, laajennuksien tekoon kehitetty skriptikieli
Maemo	Nokian käyttämä, Linux-pohjainen kehitysalusta mobiililaitteille
Python	Alustariippumaton, tulkattava ohjelmointikieli
QML	Qt Meta Language; Javascript-kieleen pohjautuva, käyttöliittymien ohjelmointiin kehitetty kieli
Qt	Alustariippumaton ohjelmistokehitysympäristö
SDK	Software Development Kit; kehitysympäristö
SVG	Scalable Vector Graphics; vektorikuvien kuvauskieli
Symbian OS	Nokian käyttämä älypuhelinkäyttöjärjestelmä
UI	User Interface; sovelluksen käyttöliittymä
XML	Extensible Markup Language; tiedon tallentamiseen ja välittämiseen käytettävä merkintäkieli

1 JOHDANTO

Viimeisen kahden vuoden aikana älypuhelinmarkkinat ovat muuttuneet merkittävästi. Nokian ikääntyvä Symbian-käyttöjärjestelmä on menettänyt nopeaan tahtiin markkinaosuuttaan ja sen ohi ovat nousseet Applen puhelimissaan käyttämä iOS sekä Googlen Android. Nokian alkuvuodesta 2011 ilmoittama päätös siirtyä käyttämään uusissa älypuhelimissaan Microsoftin kehittämää Windows Phone -käyttöjärjestelmää heikensi Symbianin asemaa entisestään.

Yksi vaihtoehto tähän asti Symbian-sovelluksia Qt-kehitysympäristössä tehneille sovelluskehittäjille on siirtyminen Androidiin. Android on nopeassa ajassa kasvattanut suosiotaan ja on tällä hetkellä maailmanlaajuisesti mitattuna markkinaosuudeltaan suurin älypuhelinkäyttöjärjestelmä. Sitä hyödyntäviä älypuhelimia valmistavat esimerkiksi Samsung, HTC ja Motorola.

Tässä työssä perehdytään Androidin tarjoamiin mahdollisuuksiin aikaisemmin Qt-sovelluskehitystä tehneen ohjelmistokehittäjän näkökulmasta. Erityisesti huomiota kiinnitetään asioihin, jotka poikkeavat merkittävästi Qt-kehityksessä totutuista. Lisäksi selvitetään, mitkä ovat Android-sovelluskehityksen suurimmat haasteet ja miten niihin voi varautua.

2 TEKNOLOGIAT

2.1 Qt

Qt (lausutaan *cute*) on alustariippumaton kehitysympäristö, jota käytetään yleisimmin graafisten sovellusten luomiseen. Qt:a hyödyntävät esimerkiksi mediasoitinsovellus VLC, verkkopuhelusovellus Skype sekä KDE-ohjelmistokokonaisuus (Qt in use: Desktop). Qt on myös ensisijainen sovelluskehitysalusta Nokian Symbian- ja Maemo-älypuhelimille sekä Meego-yhteensopivalle Nokia N9 -älypuhelimelle (Develop for the Nokia N9).

2.1.1 Historia

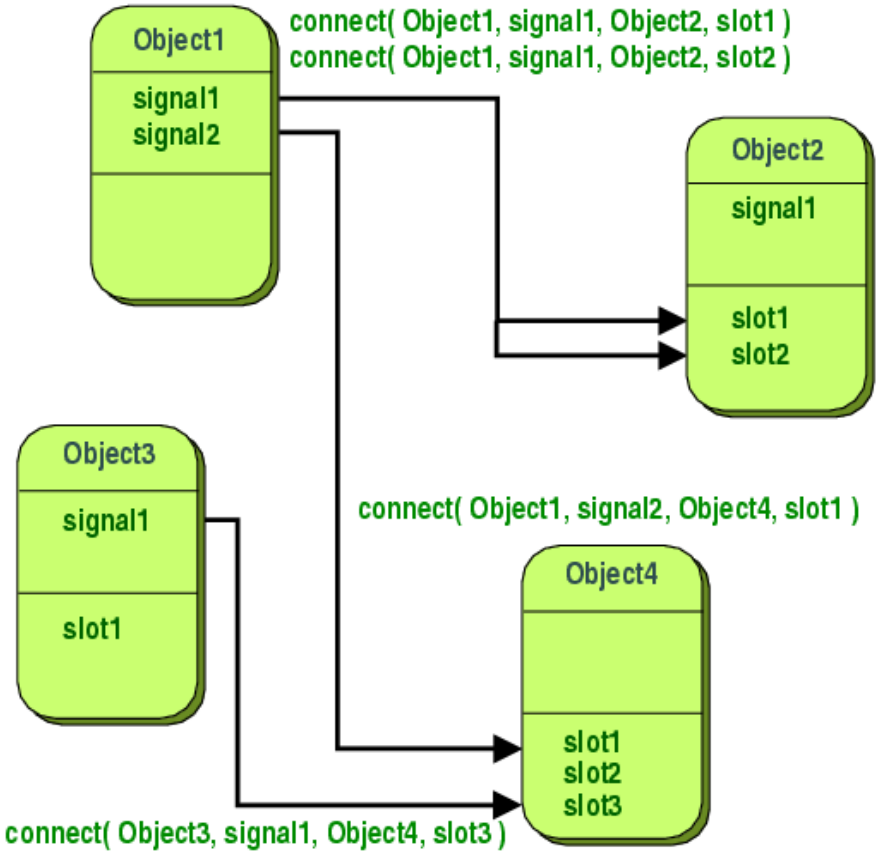
Qt:n kehityksestä on ollut vastuussa vuodesta 1994 asti norjalainen Qt Development Frameworks (vuoteen 2008 asti nimellä Trolltech). Vielä alkuaikoinaan Qt tuki pelkästään eri työpöytäympäristöjä (Windows, Unix ja Mac OS). Nokia osti Trolltechin kesäkuussa 2008 ja yrityksen painopiste siirtyi Qt:n sovittamiseen Nokian älypuhelinalustoille Symbianille ja Maemolle. (Who we are.)

1.12.2009 julkaistiin Qt:n versio 4.6, joka lisäsi tuen aiemmin tuettujen työpöytäympäristöjen lisäksi myös Symbianille ja Maemolle. Myöhemmin julkaistu versio 4.7 lisäsi tuen Qt Quick -käyttöliittymätyökalulle.

2.1.2 Sovelluskehitys

Qt-sovellus voidaan kirjoittaa useilla eri tavoilla. Ennen version 4.7 julkaisua koko sovellus kirjoitettiin usein pelkästään C++:lla, mutta nykyään Qt Quick mahdollistaa sovelluksen käyttöliittymän kirjoittamisen Javascriptia muistuttavalla QML-kielellä. Koko sovelluksen kirjoittaminen pelkästään QML:llä on myös mahdollista, jos sovelluslogiikka toteutetaan JavaScriptilla.

Qt laajentaa C++:aa useilla omilla konsepteillaan. Näkyvin näistä on olioiden väliseen kommunikointiin käytettävät signaalit (signals) ja lokerot (slots), joiden toiminta vastaa Observer-suunnittelumallia. Kuviossa 1 on esimerkki olioiden välisistä suhteista tätä mekanismia käytettäessä. Qt:n Meta Object Compiler (moc) käsittelee Qt-makroja sisältävät kooditiedostot ja generoi niiden pohjalta standardia C++-koodia.



KUVIO 1. Olioiden väliset suhteet, kun käytetään signaaleja ja lokeroita (Qt Developers)

Vaikka Qt:sta löytyy sisäänrakennettu tuki pelkästään C++:lle, on sille tehty monia kolmannen osapuolen laajennuksia eri ohjelmointikielille, kuten esimerkiksi Javalle (Qt Jambi), Pythonille (PyQt) ja Lualle (lqt). (Programming Language Support.)

2.2 Android

Android on pääasiassa mobiililaitteille suunnattu avoimen lähdekoodin ohjelmistopino, johon kuuluu virallisen määritelmän mukaan käyttöjärjestelmän lisäksi myös siihen liittyvä väliohjelmisto sekä joukko tärkeimpiä sovelluksia. Useimmissa yhteyksissä

Androidista puhuttaessa tarkoitetaan kuitenkin pelkästään sen käyttöjärjestelmäosaa. Android on helppo mieltää pelkästään Googlen hallinnoimaksi projektiksi, mutta todellisuudessa sen takana on useita kymmeniä teknologia-alan yrityksiä. (What is Android?)

2.2.1 Historia

Android sai alkunsa, kun ohjelmistojätti Google osti vuoden 2005 elokuussa kalifornialaisen startup-yrityksen Android Incin. Tällöin ei vielä ollut saatavilla tarkempaa tietoa startup-yrityksen salaisesta projektista, mutta sen uskottiin liittyvän täysin uuteen mobiililaitteiden käyttöjärjestelmään. (Elgin 2005.)

Marraskuussa 2007 joukko johtavia langattoman teknologian yrityksiä ilmoitti perustaneensa Open Handset Alliancen (OHA). Googlen johtaman yritysliittouman perustajajäseniin kuului sen itsensä lisäksi yli 30 suurta teknologiayritystä, kuten esimerkiksi matkapuhelinvalmistajat HTC, LG, Motorola ja Samsung, sekä teleoperaattorit T-Mobile ja Sprint Nextel. Yritysliittouma ilmoitti tavoitteekseen uusien innovaatioiden edistämisen sekä paremman käyttökokemuksen tuomisen mobiililaitteisiin. Samalla se julkisti kehittävänsä uutta Androidiksi nimeämäänsä käyttöjärjestelmää, joka olisi pääroolissa näiden tavoitteiden saavuttamisessa. (Open Handset Alliance 2007.)

Androidin kehityksen tärkeimmäksi periaatteeksi otettiin avoimuus, mikä näkyy selvimmin siinä, että koko Android-ohjelmistopino on avointa lähdekoodia. Tämä yhdistettynä sallivaan Apache v2 -lisenssiin antaa teleoperaattoreille ja matkapuhelinvalmistajille hyvin vapaat kädet muokata käyttöjärjestelmää haluamansa kaltaiseksi. (Apache License, Version 2.0.)

Ensimmäinen versio Androidista, versionumerolla 1.0, julkaistiin 23.9.2008. Samana päivänä julkaistiin myös Android SDK:n versio 1.0. Tämän jälkeen Androidiin on laskentatavasta riippuen julkaistu kahdeksasta yhteentoista suurempaa päivitystä. Yksi tapa on laskea julkaistut pääversiot, jolloin tämän hetkinen uusin julkaistu versio on järjestyksessään seitsemäs. Google kutsuu pääversioitaan myös koodinimillä, joiden teemana ovat erilaiset jälkiruuat. Esimerkiksi Androidin version 2.3 koodinimi on Gingerbread (piparkakku). Toinen tapa laskea eri versiot on sovelluskehityksessä versionumeroinnin

sijaan käytettävä API-taso, joka kertoo, kuinka monennetta ohjelmointirajapinnan versiota julkaistu päivitys tukee. Esimerkiksi Androidin versio 3.2 tukee API-tasoa 13. (Android API Levels.)

Ensimmäinen Androidia hyödyntänyt kuluttajakäyttöön julkaistu laite oli taiwanilaisen matkapuhelinvalmistaja HTC:n älypuhelin HTC Dream, joka saapui myyntiin Yhdysvalloissa lokakuussa 2008 teleoperaattori T-Mobilen kautta nimellä T-Mobile G1. Vuoden 2009 aikana useat eri matkapuhelinvalmistajat julkaisivat oman Android-älypuhelimensa, ja vuoden lopussa myynnissä oli jo 16 Android-käyttöjärjestelmää hyödyntävää puhelinmallia (All Android phones of 2009). Vuonna 2009 myydyistä älypuhelimista 3.9 % käytti Androidia, kun edellisvuonna määrä oli vielä 0.5 % (Gartner 2010).

Vuonna 2010 Android teki todellisen läpimurtonsa mobiililaitteiden käyttöjärjestelmänä. Vuoden aikana myydyistä älypuhelimista 22.7 % käytti Androidia ja sen edellä myyntiluvuissa oli vain Symbian (Gartner 2010). Myyntiä vauhdittivat erityisesti useat hyviä arvioita saaneet kalliimman hintaluokan Android-älypuhelimet, kuten HTC Desire, Samsung Galaxy S ja Motorola Droid X. Samana vuonna julkaistiin ensimmäiset Androidia käyttäneet tablet-tietokoneet.

Helmikuussa 2011 Google julkaisi Androidista version 3.0 (Honeycomb), joka on suunniteltu pelkästään tablet-laitteilla käytettäväksi. Maaliskuussa markkinatutkimusyhtiö comScore julkisti tutkimuksensa, jonka mukaan Android on ohittanut RIM:n Blackberryn käytetyimpänä älypuhelinalustana Yhdysvalloissa (comScore 2011). Lokakuussa 2011 julkistettiin Androidin versio 4.0 (Icecream Sandwich). Versiosta 3.0 poiketen se toimii sekä tablet-tietokoneissa että älypuhelimissa.

2.2.2 Laitteista

Android-laitteiden kirjo kattaa älypuhelinten ja tablet-tietokoneiden lisäksi esimerkiksi netbookit ja E-kirjojen lukulaitteet. Tässä opinnäytetyössä oletetaan kuitenkin, että sovellusta ollaan kehittämässä pelkästään älypuhelimille, eikä sen kehityksessä tarvitse huomioida muita laitteita.

Android-laitteiden ominaisuudet poikkeavat merkittävästi toisistaan. Taulukossa 1 on listattuna neljän erilaisen Androidia käyttävän älypuhelimen tärkeimpiä ominaisuuksia. Puhelimista ensimmäinen edustaa ominaisuuksiltaan tyypillistä kalliimman hintaluokan Android-älypuhelinta, kun taas loput kolme mallia ovat hintaluokan alapäästä. Puhelinten näyttökoot vaihtelevat noin kahdesta ja puolesta tuumasta lähemmäs viiteen tuumaan. Samaten suoritinydinten lukumäärä ja kellotaajuus, näytön tarkkuus, sisäisen muistin määrä sekä muut laitteiden ominaisuudet vaihtelevat merkittävästi eri laitemallien välillä. Tarkemmin laiteskaalan laajuudesta aiheutuvista ongelmista kerrotaan kappaleessa 4.1.

TAULUKKO 1. Esimerkkejä Androidia käyttävistä puhelinmalleista

Nimi	Android-versio	CPU	Näytön koko	Näytön tarkkuus	Fyysinen näppäimistö
Sensation 4G	2.3.3	2x1,2 GHz	4.3 tuumaa	540x960	Ei
Blade	2.2	600 MHz	3.5 tuumaa	480x800	Ei
XPeria X10 Mini	2.1	600 MHz	2.55 tuumaa	240x320	Kyllä
Galaxy 5	2.2	600 MHZ	2.8 tuumaa	240x320	Ei

Android-älypuhelimissa on vähintään kuusi, mutta yleensä ainakin seitsemän näppäintä. Laitteiden minimivaatimuksissa on listattuna virtanäppäin sekä näppäimet äänenvoimakkuuden säädölle. Lisäksi Android-yhteensopivuuden saavuttamiseksi laitteen täytyy jossain muodossa toteuttaa navigointipainikkeet, joista pakollisia on kolme (Android 2.3 Compability Definition). Aikaisemmin nämä näppäimet löytyivät laitteista poikkeuksetta fyysisinä painikkeina, mutta tilanne on muuttumassa ja nykyään markkinoilla onkin laitteita, joissa osa näppäimistä on toteutettu ohjelmallisesti. Kuvassa 1 on esimerkki navigointipainikeasettelusta.

KUVA 1. Navigointinäppäimet Google Nexus -puhelinmallissa

Yhteensopivuusvaatimuksissa määritellyt neljä navigointinäppäintä ovat:

- Koti-näppäin, jota painamalla aktiivinen sovellus siirtyy näkymättömäksi taustalle ja puhelin siirtyy kotinäkymään. Aktiivisena oleva sovellus ei voi käsitellä tätä näppäinpainallusta, vaan se siirtyy aina suoraan käyttöjärjestelmän käsiteltäväksi.
- Valikko-näppäin, jota painaessa sovellus voi näyttää valintavalikon. Tarkemmin valintavalikosta kappaleessa 3.4.
- Takaisin-näppäin, jota painaessa sovellus siirtyy yhden näkymän taaksepäin.
- Haku-näppäin, jota painaessa sovellus voi näyttää jonkinlaisen hakunäkymän. Oletuksena näppäin ei tee mitään, vaan sovelluskehittäjän on itse tehtävä sille toteutus tapauskohtaisesti. Haku-näppäin on ainoa neljästä navigointinäppäimestä, joka ei ole pakollinen.

2.2.3 Alustasta

Androidin perustan muodostaa Linux-ytimen versioon 2.6 pohjautuva järjestelmäydin. Käyttöjärjestelmäydin tarjoaa Android-järjestelmän ydinpalvelut, kuten muistin ja prosessien hallinnan, ja toimii abstraktiotasona laitteiston ja muun ohjelmistopinon välillä.

Android sisältää kokoelman C/C++ -kirjastoja, joita järjestelmän komponentit käyttävät ja joita sovelluskehittäjät voivat hyödyntää näiden komponenttien tarjoamien rajapintojen kautta. Näitä kirjastoja ovat esimerkiksi

- relaatiotietokantamoottori SQLite
- C-ohjelmointikielen standardikirjasto glibc
- kirjasinrasterointimoottori FreeType.

Kaikki Android-sovellukset toimivat omina prosesseinaan, joista jokaisella on käytössään oma ilmentymä Androidin käyttämästä Dalvik-virtuaalikoneesta. Tällöin jokaisen sovelluksen koodi suoritetaan erillään muista sovelluksista, mikä lisää tietoturvaa. Dalvik hyödyntää rekisteripohjaista arkkitehtuuria ja se on optimoitu tilanteisiin, joissa useita virtuaalikoneen ilmentymiä ajetaan samanaikaisesti. Androidin versiosta 2.2 alkaen Dalvik käyttää ajonaikaista kääntämistä (JIT).

Android-sovellukset kirjoitetaan pääsääntöisesti Javalla. Androidin käyttämä, Apache Harmonyyn pohjautuva Java-implementaatio vastaa pääpiirteiltään Java SE:tä. Se ei kata kuitenkaan kaikkea mitä Java SE kattaa, vaan esimerkiksi käyttöliittymäkirjastot Swing ja AWT eivät sisälly siihen. Käännetyt Java-luokat muunnetaan Dalvik Executable -muotoon (*.dex*), jolloin niitä voidaan ajaa Dalvik-virtuaalikoneessa.

2.2.4 Asentaminen laitteeseen

Käännetty sovellus paketoidaan kaikkien sen tarvitsemien resurssitiedostojen kanssa Android-pakettiin, jonka tunnistaa tiedostopäätteestä *.apk*. Android-paketti kelpaa sellaisenaan asennettavaksi Android-laitteeseen. Käyttöjärjestelmä luo jokaiselle asennettavalle sovellukselle oman käyttäjä-id:n. Sovellukselle kuuluville tiedostoille asetetaan käyttöoikeudet siten, että vain sovellus itse pääsee lukemaan niitä.

2.2.5 Sovelluskomponentit

Android-sovellukset rakennetaan sovelluskomponenteista. Jokaisella komponentilla on sovelluksen toiminnan kannalta selkeä yksilöllinen roolinsa ja teoriassa mikä tahansa komponenteista voi toimia pisteenä, jonka kautta sovellukseen siirrytään. Komponentit ovat myös usein riippumattomia toisistaan. Erilaisilla komponenteilla on myös toisistaan poikkeavat elinkaarensa. (Application Fundamentals.)

Sovelluskomponentti aktivoidaan lähettämälle sille viesti, joka sisältää abstraktin kuvauksen operaatiosta, jonka komponentin on tarkoitus suorittaa. Viestiin sisältyy aina suoritettavan operaation tyyppi sekä mahdollisesti operaation suorittamiseen tarvittavaa tietoa. Intent (suom. aikomus, aie) on passiivinen tietorakenne, joka kapseloi nämä komponenteille välitettävät operaatiopyynnöt. Mikäli Intent ei suoraan määrittele minkä komponentin se haluaa suorittavan operaation, valitsee käyttöjärjestelmä tarkoitukseen parhaiten soveltuvan komponentin. Sovellukset voivat rekisteröidä komponenttejaan koko järjestelmän käytettäväksi, jolloin muut sovellukset voivat hyödyntää niitä vapaasti.

Esimerkiksi jos sovellus tarvitsee valokuvausominaisuutta, voi se lähettää järjestelmälle pyynnön, jossa operaatioksi on määriteltynä valokuvan ottaminen. Tämän jälkeen käyttäjä voi valita asennetuista valokuvaussovelluksista mieleisensä, ja kun valittu sovellus on toteuttanut pyynnön, palauttaa se valmiin valokuvan pyynnön lähettäjälle. Näin sovelluksen ei tarvitse toteuttaa itse valokuvausominaisuutta, vaan se voi käyttää hyödyksi puhelimen muiden sovellusten tarjoamia palveluita. Lisäksi käyttäjälle annetaan vapaus hyödyntää asentamiaan sovelluksia Androidin omien vakiosovellusten sijaan.

Activity (suom. aktiviteetti) on komponentti, jota käytetään sovelluksen graafisen käyttöliittymän rakennuspalasena. Karkeasti ottaen jokaista näkymää vastaa yksi Activity. Activityja voi olla kerralla luotuna useampia, jolloin järjestelmä pitää niitä pinossa. Pinon päällimmäisenä oleva Activity poistetaan, kun käyttäjä painaa Takaisin-näppäintä.

Service (suom. palvelu) on taustalla toimiva komponentti, joka suorittaa pitkäkestoisia operaatioita. Tällaisia ovat esimerkiksi tiedostojen lataaminen, musiikin toistaminen tai muut raskaat I/O-operaatiot. Palvelut jatkavat toimintaansa taustalla vaikka sovelluksen käyttöliittymä ei olisikaan enää aktiivisena. Palveluilla ei koskaan ole omaa käyttöliittymää. Palveluita voidaan suorittaa myös ajastettuina.

Content provider (suom. sisällöntarjoaja) kapseloi ja hallinnoi pääsyn sovelluksen tallentamaan tai tuottamaan tietoon. Sisällöntarjoaja vastaanottaa muilta sovelluksilta pyyntöjä lukea tai muokata tietovarastoa, joka voi sijaita paikallisesti laitteella esimerkiksi SQLite-tietokannassa, mutta myös etäpalvelimella. Esimerkiksi suurimpaan osaan järjestelmän tallettamasta tiedosta pääsee käsiksi järjestelmän omien sisällöntarjoajien välityksellä.

Broadcast receiver (suom. tiedotusten vastaanottaja) reagoi tuleviin tiedotuksiin. Tällainen voi esimerkiksi olla ilmoitus siitä, että laitteen akku on lähes tyhjä. Useimmiten tiedotusten lähettäjänä on käyttöjärjestelmä, mutta sovellukset voivat myös lähettää ja vastaanottaa tiedotuksia keskenään.

AndroidManifest.xml on sovellukseen pakollisena osana kuuluva XML-tiedosto, johon listataan sovellukseen kuuluvat sovelluskomponentit. Se sisältää myös paljon muuta vaihtoehtoista tai pakollista tietoa sovelluksesta, kuten sovelluksen version, sen tukemat

näyttökoot, sen tukemat alustaversiot ja sen käyttämät laitteen ominaisuudet. Tätä tietoa käyttävät hyödykseen esimerkiksi käyttöjärjestelmä itse sekä erilaiset sovellusten jakelupaikat (mm. Google Play).

Koodiesimerkissä 1 on yksinkertaisen tekstinkäsittelysovelluksen *AndroidManifest.xml* -tiedosto. Sovellus koostuu kahdesta Activitysta, joista toista käytetään tekstisisällön esittämiseen ja toista tekstin editoimiseen. Lisäksi sovellukseen kuuluu Content provider, jonka kautta tekstitiedostoja luetaan ja talletetaan.

KOODIESIMERKKI 1: Yksinkertaisen sovelluksen AndroidManifest.xml-tiedosto

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.notepad">

    <application android:icon="@drawable/app_notes"
        android:label="@string/app_name">

        <provider android:name="NotePadProvider"
            android:authorities="com.google.provider.NotePad" />

        <activity android:name="NotesList" android:label="@string/title_notes_list">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
            </intent-filter>
        </activity>

        <activity android:name="NoteEditor"
            android:theme="@android:style/Theme.Light"
            android:label="@string/title_note"
            android:configChanges="keyboardHidden|orientation">

            <intent-filter android:label="@string/resolve_edit">
                <action android:name="android.intent.action.EDIT" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:targetSdkVersion="4" android:minSdkVersion="3"/>
</manifest>
```

2.2.6 Resurssit

Tässä yhteydessä resursseilla tarkoitetaan kaikkia muita sovelluksen tarvitsemia tiedostoja kuin kooditiedostoja. Näitä ovat esimerkiksi kuva-, ääni- ja kirjasintiedostot sekä sovelluksen käyttöliittymän XML-määrittelytiedostot. Sovelluksen käyttämät resurssit tallennetaan */res/*-kansioon. Sen sisällä ne jaetaan vielä alikansioihin resurssin tyypin mukaan. Kuten taulukosta 2 näkyy, suurin osa sovelluksen käyttämistä resursseista on XML-muotoista tietoa.

TAULUKKO 2: Sovelluksen tiedostohierarkia

Kansio	Sisältö	Tiedostoformaatit
assets/	Tiedostot, joihin halutaan viitata suoraan alkuperäisellä tiedostonimellä	Kaikki
anim/	Animaatiot	XML
color/	Lista värejä, joista valitaan yksi tilan perusteella	XML
drawable/	Grafiikkatiedostot	XML, PNG, JPG..
layout/	Asetelmatiedostot	XML
menu/	Valikkotiedostot	XML
raw/	Tiedostot, joille halutaan luoda ID, mutta joita ei haluta käsiteltävän sen enempää	Kaikki
values/	Tiedostoja, jotka sisältävät yksinkertaisia arvoja, esimerkiksi yksittäisiä värejä, tekstiä tai kokoarvoja	XML
xml/	XML-muotoinen tieto, joka ei kuulu mihinkään muualle	XML

Kaikille muille resursseille, paitsi niille jotka on talletettu *res/assets/*-kansioon, generoidaan sovelluksen käännösvaiheessa oma id-tunniste. Resursseihin viitataan koodissa aina niiden tunnisteen kautta. Jokaisesta resurssista on mahdollista tarjota oletusversion lisäksi myös erityisesti tietylle kielelle, näyttökoolle, näyttötyypille tai muulle vastaavalle laitteen ominaisuudelle räätälöity versio. Käyttöjärjestelmä valitsee sovelluksen suorituksen aikana resurssin eri versioista käytettäväksi sen, joka vastaa parhaiten käytetyn laitteen ominaisuuksia.

3 SOVELLUKSEN UUDELLEENKIRJOITTAMINEN

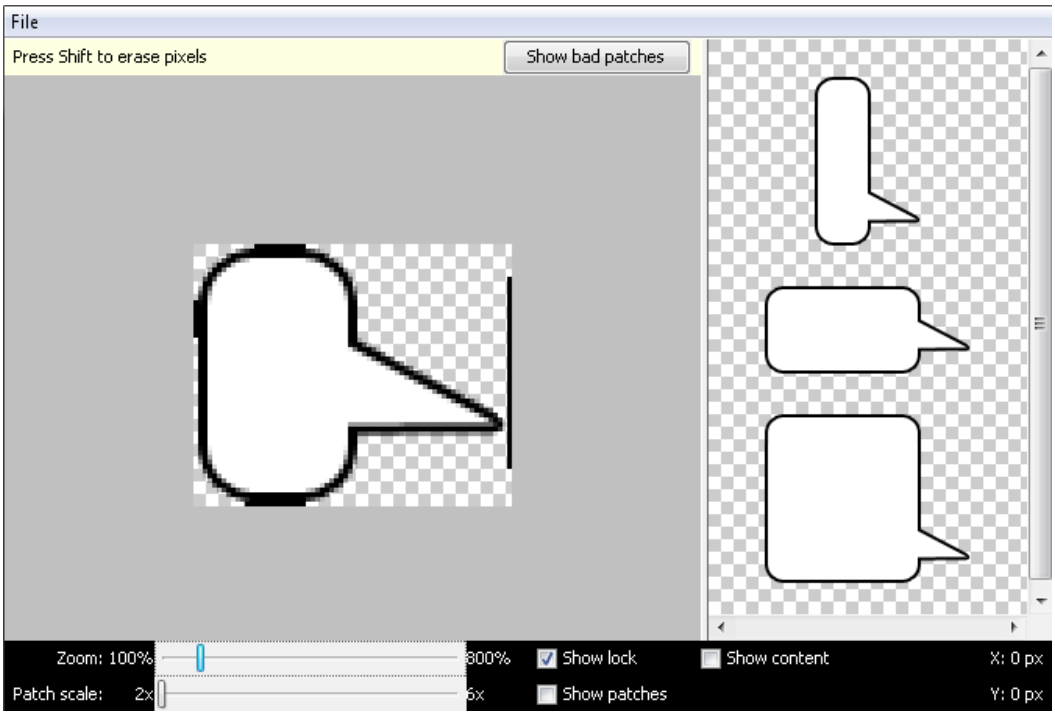
3.1 Media- ja kirjasintiedostojen uudelleenkäyttö

Sovelluskehittäjän työtaakkaa helpottaa, jos aiemmin eri alustalla käytettyjä kuva-, ääni-, video- ja kirjasintiedostoja voi käyttää uudelleen sovelluksen Android-versiossa. Useiden tiedostotyyppien kohdalla näin onkin, mutta tiettyjä puutteita ja merkittäviä eroavaisuuksia on silti olemassa.

3.1.1 Kuvatiedostot

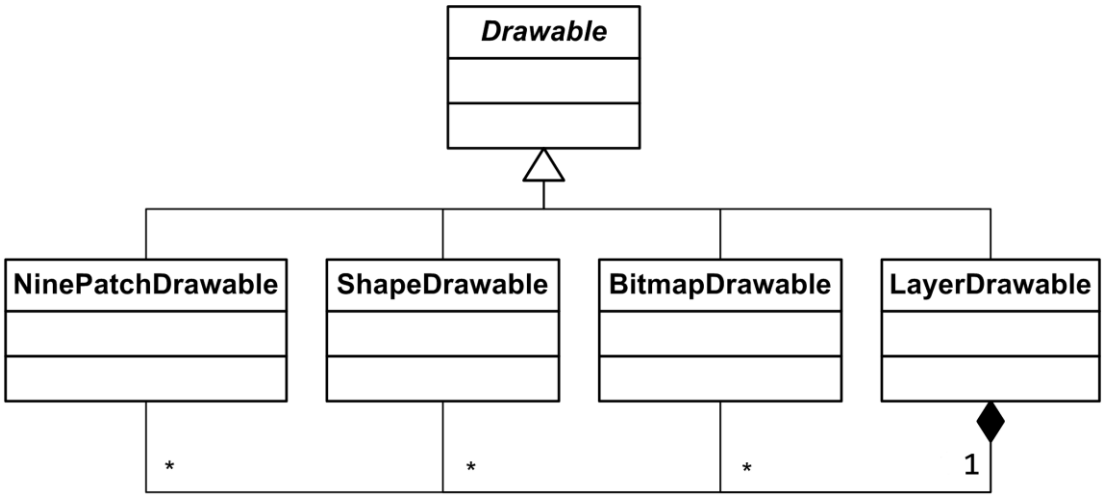
Androidin tukemat kuvatiedostoformaatit ovat JPG, GIF, PNG ja BMP. Tuettujen formaattien listalta puuttuu vektorigrafiikkaformaatti SVG, jota käytetään usein Qt-sovelluskehityksessä. Vektorigrafiikkaformaatin puuttumisen vuoksi käytetyistä kuvista tarvitsee usein olla useampi eri versio eri näyttötarkkuuksia varten. Järjestelmä osaa kuitenkin suorittaa automaattisesti kuvien skaalaamisen, joten suurin painoarvo on kuvien korkearesoluutioisilla versioilla. Android käyttää kuvatiedostoista nimeä *Drawable*. *Drawableja* on useaa eri tyyppiä, ja niitä voi yhdistellä samaan kuvaan useiksi eri tasoiksi käyttämällä *LayerDrawable*-tyyppiä.

9PatchDrawable korvaa osittain vektorigrafiikkatuen puuttumista esittelemällä 9-patch-kuvatyypin. 9-patch on tavallinen PNG-kuva, jonka reunoille lisätään yhden pikselin levyinen ylimääräinen alue. Tälle alueelle merkataan mustalla värillä, mitä kuvan osia voidaan tarvittaessa skaalata. Kuvassa 2 on esimerkki 9-patch-kuvan luomisesta, kun apuna käytetään SDK:hon sisältyvää Draw 9-patch -työkalua. 9-patch-kuvat tallennetaan tiedostopäättellä .9.patch */drawable/*-kansioon. (Nine-patch.)



KUVA 2. 9-patch-kuvan luominen Draw 9-patch -työkalulla

Valinta eri kuvatyyppien välillä tehdään tapauskohtaisesti. Koodissa kuviin viitataan niiden *Drawable*-kantaluokan kautta. Näin ollen sovelluksen kehittämisen kannalta ei ole merkitystä sillä, millaisessa muodossa käytetyt kuvat ovat. Useimmiten kannattaa kuitenkin pyrkiä välttämään bittikarttakuvien käyttöä, jos vain mahdollista. Bittikartta-kuvat joudutaan aina skaalaamaan eri näyttötiheyksille ja mikäli järjestelmän itse suorittaman skaalaamisen tulos ei ole tarpeeksi hyvä, tarvitsee kuvista luoda erikseen useita eri versioita eri näyttötiheyksille. Kuviossa 2 on esitettynä eri *Drawable*-luokkien yhteydet toisiinsa. *LayerDrawable*-luokkaa käyttämällä voi kuvion koostaa useammista toisista kuvioista.



KUVIO 2. Drawable-luokkien yhteydet toisiinsa

3.1.2 Ääni- ja videotiedostot

Android tukee useita yleisimpiä äänitiedostoformaatteja, kuten MP3, Ogg Vorbis ja MIDI. Tuki häviöttömällä FLAC-tiedostoformaatille lisättiin kuitenkin vasta Androidin versioon 3.1, joten sitä aiemmilla versiolla täytyy tehdä muunnos johonkin tuetuista tiedostomuodoista. Äänitiedostot sijoitetaan */res/raw/*-kansioon. Videotiedostoformaateista tuettuja ovat H.263, H.264 AVC, MPEG-4 SP ja VP8. Videotiedostot sijoitetaan samaan paikkaan kun audiotiedostot. (Android Supported Media Formats.)

3.1.3 Kirjasintyylit

Android tukee TrueType- ja OpenType-kirjasinformaatteja. Kirjasintyylitiedostot sijoitetaan */assets/*-kansioon. Androidista vakiona löytyvät fontit ovat Droid Sans, Droid Sans Mono ja Droid Sans Serif. (Droid Fonts Overview.)

3.2 Lokalisointi

Qt-sovelluksissa lokalisointi tehdään käyttämällä erillistä Qt Linquist -työkalua. Käännettävät kohdat merkitään koodissa ja tätä tietoa käyttäen generoidaan käännöstiedostot, joita Qt Linquist -työkalu lukee. Qt Linquist generoi valmiista käännöksistä jokaiselle

käännökselle oman käännöstiedoston, joista ladataan suorituksen aikana dynaamisesti käytettäväksi laitteen kieltä vastaava versio (Internationalization with Qt.)

Android-sovelluksissa käännösten tekemisessä käytetään hyödyksi resurssien dynaamista latausta. Jokaiselle sovelluksen tukemalle kielelle tehdään oma version XML-tiedostosta, joka sisältää sovelluksen käyttämät tekstit. Näistä järjestelmä valitsee käytettäväksi sen, joka vastaa parhaiten laitteen kieltä. Koodiesimerkissä 2 on esitettynä XML-tiedosto, joka sisältää sovelluksen käyttämät tekstit suomen kielellä.

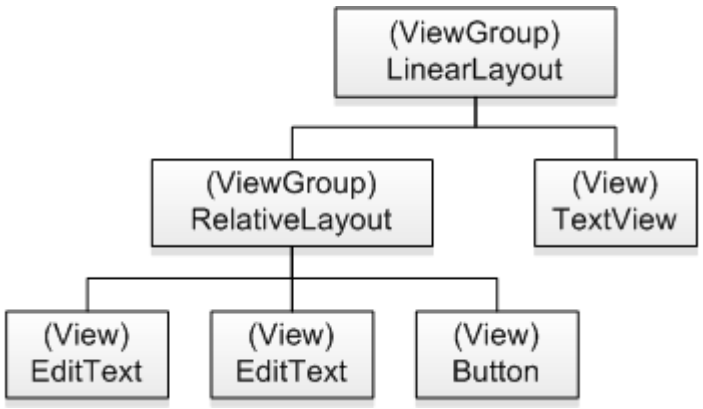
KOODIESIMERKKI 2: Kielimääritykset XML:nä

```
res/values-fi/strings.xml

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello_world">Terve maailma</string>
</resources>
```

3.3 Käyttöliittymien toteuttaminen

Android-sovellusten käyttöliittymät rakennetaan käyttöliittymäelementeistä, jotka perivät aina kantaluokan *View*. *View*-luokasta periytyy myös luokka *ViewGroup*, joka toimii kantaluokkana kaikille asetelmaluokille (eng. layout). Elementtihierarkian ylimpänä on aina jokin asetelmaluokka, kuten kuvion 3 hierarkiassa.



KUVIO 3. Esimerkki käyttöliittymäkomponenttihierarkiasta

Qt-käyttöliittymät toteutetaan vastaavanlaisena hierarkiana, jossa *Viewiä* vastaava luokka on *QWidget* ja *ViewGroupia* vastaa *QLayout*. Suurimmalle osalle Qt:n asetelmaluokkatoteutuksista löytyy vastaavan tai lähes vastaavan toiminnallisuuden toteuttava luokka tai luokat Androidin käyttöliittymäkirjastosta. Näistä muutamia on esiteltynä taulukossa 3.

TAULUKKO 3. Vertailu Layout-luokkien välillä

Qt:n Layout-luokka	**Vastaava Androidin Layout-luokka**
QGridLayout	*GridView*
QHBoxLayout, QVBoxLayout	*LinearLayout, RelativeLayout*
QStackedLayout	*ViewFlipper*
QTabWidget	*TabView*

Eniten käytetty Androidin asetelmaluokka on *RelativeLayout*, jolle ei löydy suoraa vastinetta Qt:n asetelmaluokista. *RelativeLayoutia* käytettäessä jokaiselle sen sisältämälle käyttöliittymäelementille voidaan asettaa sijainti suhteessa muihin käyttöliittymäelementteihin tai *RelativeLayoutiin* itseensä. Näin voidaan luoda asetelmia, jotka ovat riittävän joustavia useimpiin käyttötarkoituksiin.

Androidin käyttöliittymäkirjasto sisältää korvaavat komponentit asetelmaluokkien lisäksi myös monille muille Qt:n käyttöliittymäelementeille. Taulukossa 4 on esimerkkejä näistä vastaavuuksista.

TAULUKKO 4. Vertailu käyttöliittymäkomponenttien välillä

Qt-käyttöliittymäkomponentti	**Vastaava Android-käyttöliittymäkomponentti**
QPushButton	*Button*
QSpinBox	*Spinner*
QLineEdit	*EditText*
QSlider	*SeekBar*
QProgressBar	*ProgressBar*
QTimeEdit	*TimePicker*

Jos Androidin käyttöliittymäkirjastosta ei löydy tarkoituksiin sopivaa valmista komponenttia, voi sellaisen kirjoittaa itse. Tämä tapahtuu periyttämällä luokka *View* ja toteuttamalla tarvittavat metodit. Useimmat näistä metodeista liittyvät komponentit piirtämiseen sekä sen tarvitseman tilan laskentaan.

Androidilla jokaisen näkymän käyttöliittymäkomponenttien luonti voidaan tehdä suoraan koodissa *Activity*-luokan *onCreate*-metodissa luomalla jokainen komponenttiolio erikseen ja asettamalla sille oikeanlaiset parametrit. Käytännöllisempää on kuitenkin kirjoittaa ohjeet käyttöliittymän luontiin XML-tiedostona. Koodiesimerkissä 3 määritellään elementtihierarkia, jossa ylimpänä on *LinearLayout*. Se sisältää tekstin esittämiseen käytettäväm *TextView*-komponentin, jonka id:ksi asetetaan *hello_text*. Tämän id:n kautta voidaan tunnistaa myöhemmin luotu *TextView*-instanssi.

KOODIESIMERKKI 3. XML-tiedosto, jossa määritellään näkymän layout

```
res/layout/my_layout.xml

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/hello_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Koodiesimerkissä 4 käytetään kirjoitettua XML-mallia Activityn käyttöliittymän luomiseen. Esimerkissä kutsuttu *Activity*-luokan metodi *setContentView* käyttää järjestelmän *LayoutInflater*-nimistä palvelua, joka lukee XML-tiedoston ja luo siinä määritellyn mallin mukaisen oliohierarkian. Activity käyttää luotua hierarkiaa graafisena käyttöliittymänään. Metodi *findViewById* hakee hierarkiasta olion annetun id:n perusteella.

KOODIESIMERKKI 4. Activity, jossa luotua XML-tiedostoa käytetään

```
public class MyActivity extends Activity {
    /** Kutsutaan, kun Activity on luotu. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* luodaan Activityn käyttöliittymä
         * layout-tiedostosta 'my_layout.xml' */
        setContentView(R.layout.my_layout);
        /* kun käyttöliittymä on luotu, etsitään komponentti,
         * jonka id on 'hello_text' */
        TextView view = (TextView) findViewById(R.id.hello_text);
        /* asetetaan tekstiksi "Hello world!" */
        view.setText("Hello world!");
    }
}
```

Qt tarjoaa useita eri tapoja käyttöliittymän ulkoasun räätälöintiin. Eräs tehokas tapa on käyttää Qt Style Sheet -tyylitiedostoja. Ne muistuttavat syntaksiltaan hyvin paljon CSS-tyylitiedostoja, mutta niitä käytetään Qt-käyttöliittymäkomponenttien räätälöintiin. Toinen vaihtoehto on luoda oma *QStyle*-luokan toteutus, jota käyttöliittymäkomponentit käyttävät apuna piirtäessään itsensä. (Qt Style Sheets.)

Androidin tarjoamat mahdollisuudet käyttöliittymän räätälöintiin eivät ole yhtä monipuoliset, mutta riittävät useimmissa tapauksissa. Useimmille komponenteille voi asettaa taustakuvan, jota muuttamalla komponentit ulkonäköön voi vaikuttaa helpoiten. Taustakuva voi olla joko kuvatiedosto tai XML-formaatissa kuvattu muoto. Koodiesimerkissä 5 on määritelty suorakulmiomuoto ja sille liukuväritäyttö (*gradient*), etäisyys sisältöön (*padding*), reunaviiva (*stroke*) sekä kulmien pyöristys (*corners*). Esimerkissä olevalle muodolle ei ole asetettu staattista kokoa, vaan se skaalaa itsensä sitä käyttävän komponentit koon mukaan.

KOODIESIMERKKI 5. Suorakulmiomuodon määrittely XML:nä

res/drawable/customized_button_background.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:startColor="@color/custom_button_top"
        android:endColor="@color/custom_button_bottom"
        android:angle="-90" />
    <padding
        android:left="7dp"
        android:top="7dp"
        android:right="7dp"
        android:bottom="7dp" />
    <stroke
        android:color="#000000"
        android:width="1dp" />
    <corners
        android:radius="5dp" />
</shape>
```

Mikäli useat komponentit käyttävät samanlaista ulkoasuräätälöintiä, on suositeltavaa määritellä tämä räätälöinti erillisessä tyylitiedostossa. Tällöin sisällyttämällä tyylitiedoston komponentin määrittelyyn kaikki tyylitiedostossa esitellyt ominaisuudet tulevat voimaan komponentille. Koodiesimerkin 6 tyylitiedostossa määritelty tyyli sisältää valmiit arvot komponentin käyttämälle kirjasimelle, kirjasinkoolle sekä kirjasimen värille.

KOODIESIMERKKI 6. Tyylimääritelmä XML-tiedostona

res/values/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CustomStyle"
        parent="@android:style/TextAppearance.Medium">
        <item name="android:textColor">#000000</item>
        <item name="android:typeface">monospace</item>
        <item name="android:textSize">15dp</item>
    </style>
</resources>
```

Koodiesimerkissä 7 määritellään *Button*-komponentti, joka käyttää aiemmin määriteltyä tyyliä, ja jonka taustakuvana on aiemmin määritelty suorakulmiomuoto. Molempiin viitataan Androidin resurssinotaation mukaisesti. Lisäksi komponentille asetetaan tekstisisältö, johon viitataan myös resurssinotaation kautta.

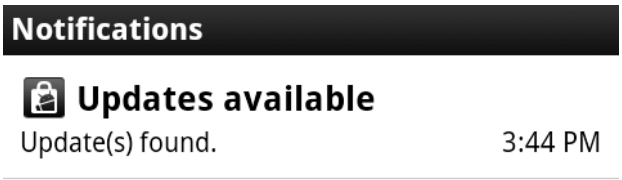
KOODIESIMERKKI 7. Komponentti, joka käyttää tyyli- ja muotomäärityksiä

```
<Button
    style="@style/CustomStyle"
    android:id="@+id/customized_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/customized_button_text"
    android:background="@drawable/customized_button_background" />
```

3.4 Androidin käyttöliittymäkonsepteja

Android toteuttaa joitain käyttöliittymäkonsepteja, joita Symbian ja Maemo eivät tue ja joita myöskään Qt ei suoraan toteuta. Näitä ei ole välttämätöntä hyödyntää silloin, kun sovelluksen käyttöliittymä sovitetaan uudelleen Android-ympäristöön, mutta niiden hyödyntäminen on helppoa ja tekee sovelluksesta miellyttävämmän käyttää.

Tilapalkki-ilmoitus on ilmoitus, joka näkyy puhelimen tilapalkissa ja johon liittyy yleensä tekstiä sekä ilmoituksen sisältöä kuvaava ikoni (Kuva 3). Tilapalkki-ilmoituksella taustalla oleva sovellus viestittää, että on tapahtunut jotain, josta käyttäjän tulisi tietää. Esimerkiksi sähköpostisovellukset viestittävät usein saapuneista viesteistä tilapalkki-ilmoituksilla. Tilapalkki-ilmoituksen näyttäminen on pakollista silloin, kun sovellus suorittaa jotain pitkäkestoista operaatiota ja halutaan, ettei järjestelmä sammuta sovellusta kesken suorituksen.



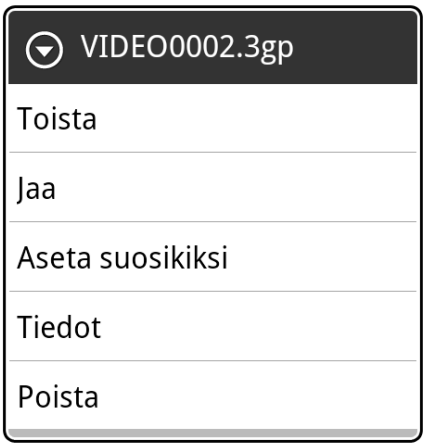
KUVA 3. Google Play -sovelluksen käyttämä tilapalkki-ilmoitus

Tilapalkki-ilmoitusten luonnista ja hallinnasta vastaa järjestelmäpalvelu *Notification-Manager*. Ilmoitusten yhteyteen voi lisätä myös värinä-, ääni- tai led-tehosteen korostamaan ilmoituksen tärkeyttä.

Toast on ilmoitus, joka näkyy aktiivisena olevan näkymän päällä, mutta ei vaikuta sen toimintaan eikä reagoi käyttäjän painalluksiin. Toast näkyy ruudulla muutaman sekunnin ajan valitussa sijainnissa, jonka jälkeen se häivytetään pois. Toasteja käytetään usein sellaisten lyhyiden viestien esittämiseen, joita on hankala muuten sovittaa käyttöliittymään. Kuvassa 4 herätyskellosovellus ilmoittaa Toastilla, että herätys on asetettu päälle.

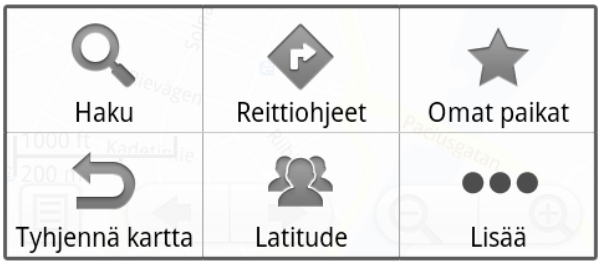
KUVA 4. Puhelinvalmistaja HTC:n Hälytykset-sovelluksessa käytetty Toast

Kontekstivalikko on valikko, joka ponnahtaa esille silloin, kun käyttäjä painaa pitkään jotakin komponenttia. Käyttötarkoitukseltaan se vastaa useista PC-ohjelmista tuttua hiiren oikeanpuoleisen näppäimen painalluksesta aukeavaa valikkoa. Kontekstivalikko sopii hyvin tärkeydeltään toissijaisten toimintojen esittämiseen. Kontekstivalikko vapauttaa tilaa käyttöliittymästä, kun harvoin käytetyt toiminnot piilotetaan sen taakse. Kuvassa 5 on esimerkki kontekstivalikosta.



KUVA 5. Puhelinvalmistaja HTC:n Galleria-sovelluksessa käytetty kontekstivalikko

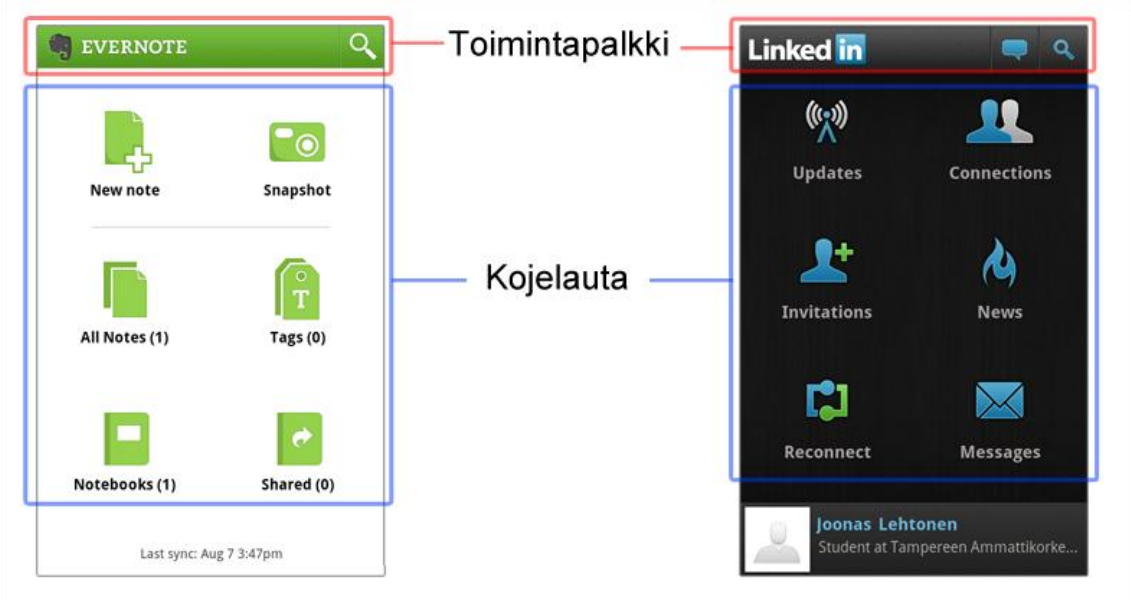
Valintavalikko sisältää listan toimintoja (Kuva 6). Se näytetään ruudun alalaidassa, kun käyttäjä painaa useimmissa laitteissa fyysisenä olevaa Valikko-näppäintä. Toimintojen määrää ei ole rajoitettu, mutta kerrallaan niistä näkyy niin monta kuin laitteen näytön koko huomioon ottaen on mahdollista esittää kahdella rivillä. Kuten kontekstivalikko-kin, valintavalikko sopii hyvin harvoin tarvittujen toimintojen piilottamiseen. Androidin versiossa 3.0 ja sitä uudemmissa valintavalikon ulkoasu poikkeaa merkittävästi aikaisemmasta, mutta sen toimintaperiaate on pysynyt samana.



KUVA 6. Maps-sovelluksen valintavalikko, kun karttanäkymä on aktiivinen

3.5 Käyttöliittymäsuunnittelumallit

Google on esitellyt joitakin käyttöliittymäsuunnittelumalleja, joita se suosittelee käytettäväksi sovelluksissa. Kaksi usein käytettyä suunnittelumallia ovat toimintapalkki (action bar) sekä kojelauta (dashboard). Kuviossa 4 on esimerkki kahdesta sovelluksesta, jotka toteuttavat molemmat näistä suunnittelumalleista. Myös useimmat Googlen itsensä kehittämät sovellukset hyödyntävät näitä suunnittelumalleja (esim. Youtube, Google Maps ja Google Translate).



KUVIO 4. Evernote- ja LinkedIn-sovellusten toteutukset toimintapalkista ja kojelaudasta

Toimintapalkki on ruudun ylälaidassa oleva palkki, jonka kautta käyttäjä voi navigoida sovellusten eri osien välillä ja käyttää usein tarvittuja toimintoja. Toimintapalkin pitäisi löytyä jokaisesta näkymästä ja sen toimintalogiikan pitäisi olla aina sama. Androidin versioon 3.0 lisättiin valmis API toimintapalkin toteuttamista varten.

Kojelauta on sovelluksen aloitusnäkymä, josta on pääsy sovelluksen eri osioihin. Kojelaudassa näkyvien toimintojen suositeltu määrä on kolmesta kuuteen kappaletta. Lisäksi kojelaudalla suositellaan näytettäväksi tietoja uusista tapahtumista. Jos sovellus toteuttaa kojelaudan lisäksi myös toimintapalkki-suunnittelumallin, pitäisi toimintapalkista löytyä pääsy kojelaudalle. (Nesladek, Bauer, Fulcher & Robertson 2010)

3.6 Julkaisu Google Playssa

Google Play (aikaisemmin Android Market) on Googlen ylläpitämä levityskanava Android-sovelluksille. Google Playta voi käyttää joko selaimella tai puhelimiin esiasennetulla Play Shop -sovelluksella. Google Play on käytettävissä yli 130 eri maassa. Julkaistut sovellukset voivat olla joko maksuttomia tai maksullisia.

Sovelluksen julkaiseminen Google Playssa on tehty kehittäjille mahdollisimman yksinkertaiseksi. Julkaiseminen edellyttää maksullisen kehittäjätilin luomista (5.4.2012 tilin avaaminen maksoi $25). Lisäksi sovelluksen manifestitiedostossa tarvitsee olla täytettynä tietyt vaaditut kohdat, sekä sovelluksen oltava käännetty julkaisutilassa. Tämän lisäksi mukaan tarvitsee liittää lyhyt kuvaus sovelluksesta, kuvankaappauksia ja jonkin verran pakollista grafiikkaa. Julkaistulle sovellukselle voi asettaa laite-, alustaversio- tai maakohtaisia rajoituksia. (Sovellusten lähettäminen.)

Sovelluksen voi julkaista myös ilman, että käyttää Google Playta julkaisukanavanaan. Tällöin jää kuitenkin ilman monia Google Playn tuomia etuja, kuten sen antamaa näkyvyyttä, automaattisia tilastointiominaisuuksia sekä valmiita maksusuoritusten käsittelyjä.

3.7 Vaihtoehtoisia tapoja

On olemassa myös muita tapoja sovelluksen luomiseksi Androidille, kuin sovelluksen kirjoittaminen Javalla käyttäen Android-kirjastoja. Näissä tavoissa on kuitenkin omat ongelmansa tai niiden käyttökohteet ovat hyvin rajattuja.

3.7.1 Android NDK

Android NDK, eli Android Native Development Kit, on kehitysympäristö joka mahdollistaa sovellusten osien tai kokonaisten sovellusten kirjoittamisen C:llä tai C++:lla. Tämä on hyödyllistä silloin, kun halutaan hyödyntää aikaisemmin kirjoitettua koodia tai saavuttaa natiivikoodin tuoma nopeusetu. (What is the NDK?)

Android-sovellus voi hyödyntää natiivikoodia kahdella eri tavalla. Sovellus voi olla kirjoitettu osaksi käyttäen Android-sovelluskehystä ja osaksi natiivikoodina, jolloin natiivikoodia kutsutaan Javasta JNI-rajapinnan kautta. Tämä toteutustapa on tuettu Androidin versiosta 1.5 eteenpäin. Toinen vaihtoehto on sovellus, joka koostuu pelkästään natiivikoodista. Tällöin ainoa sovelluksen yhteys Javaan on *NativeActivity*-luokka, joka toimii rajapintana ja välittää natiivikoodille tiedot sovelluksen elinkaareen liittyvistä tapahtumista. Tämä ominaisuus on tuettu Androidin versiosta 2.3 eteenpäin.

Natiivikoodin hyödyntäminen sopii vain hyvin rajatulle osalle sovelluksista. Mikäli Qt-sovelluksessa on käytetty puhdasta kirjastoriippumatonta C-koodia esimerkiksi raskaan laskennan suorittamiseen, voidaan tätä samaa koodipohjaa hyödyntää suoraan Android-sovelluksessa JNI-rajapinnan kautta. Toinen käyttökohde on suorituskykyä vaativat sovellukset, joissa laskentaa halutaan suorittaa natiivikoodina esimerkiksi siksi, että voidaan hallita muistinkäyttöä tehokkaammin kuin Javan virtuaalikonepohjaisessa järjestelmässä.

3.7.2 Necessitas

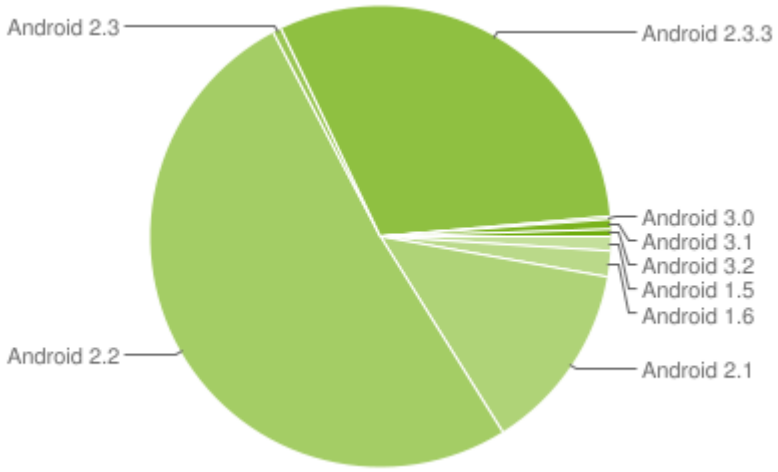
Qt Lighthouse -projektiin kuuluva Necessitas on yritys siirtää koko Qt-sovelluskehys Androidille. Necessitas ei teoriassa vaadi muutoksia olemassa olevan C++:lla kirjoitetun sovelluksen lähdekoodiin, vaan pelkkä uudelleen kääntäminen riittää. Necessitas hyödyntää Android NDK:ta toteutuksessaan. Necessitakseen kuuluu osana myös Qt Creator IDE:n versio Qt Creator for Android sekä Ministro, joka on työkalu sovelluksen vaatimien Qt-kirjastojen lataamiseen Android-laitteelle. (Vatra 2011.)

Necessitaksen käyttöön liittyy kuitenkin monia ongelmia. Projekti on vielä hyvin varhaisessa alpha-vaiheessa ja monilta osin keskeneräinen. Lisäksi Necessitas on yhteisöprojekti, josta on vastuussa pääosin yksi kehittäjä, eikä sille ole virallista tukea tai sen jatkokehityksestä ole takeita. Sovellukset istuvat myös huonosti Androidille. Ne eivät muistuta ulkoasultaan tai toiminnallisuudeltaan muita Android-sovelluksia, eikä niiden kanssa ole mahdollista käyttää monia Androidin omia konsepteja. Sovellukset toimivat myös yhtenä Activityna, eivätkä ne kykene tallentamaan tilaansa. Necessitas soveltuu nykyisessä muodossaan kuitenkin jossain määrin yksinkertaisten, QML:nä toteutettujen sovellusten tuomiseen Androidille. Sovelluksen toimiminen kuitenkin edellyttää, että laitteeseen on asennettu tarvittavat Qt-kirjastot, mikä voi olla yksinkertaisen sovelluksen tapauksessa liian työläs edellytys täytettäväksi.

4 HAASTEITA

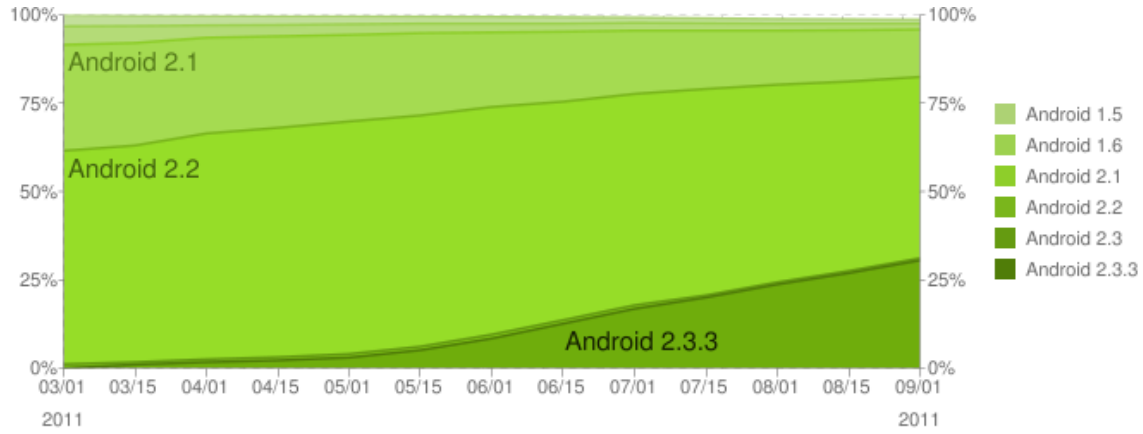
4.1 Laitekannan hajanaisuus

Suuren haasteen Android-sovelluskehityksessä muodostaa erilaisten laitteiden ja käytössä olevien alustaversioiden suuri kirjo. Jo fyysisiltä ominaisuuksiltaan käytössä olevat laitteet ovat hyvin erilaisia. Skaalan toisessa päässä on juuri ja juuri alustan minimivaatimukset täyttävä, halpahintainen älypuhelin ja toisessa 10-tuumaisella näytöllä varustettu, tehokas tablet-tietokone. Toinen merkittävästi eron laitteiden välillä muodostaa laitteen käyttämä versio Androidista. Kuviossa 5 on käytettyjen versioiden jakauma, joka on laadittu Google Playn käyttötilastojen pohjalta. Kuten kuviosta näkyy, tukemalla laitteita versiosta 2.1 eteenpäin saavuttaa yli 95 % käyttäjistä.



KUVIO 5. Käytössä olevien Android-versioiden jakauma (Android Developers 2011)

Kuviossa 6 nähdään eri alustaversioiden prosentuaalinen osuus, kun tarkasteluajanjaksona on maaliskuu 2011 - syyskuu 2011. Lähteenä ovat Google Playn käyttäjämäärät. Kuten kuviosta näkyy, uudet versiot valtaavat hitaasti tilaa vanhemmilta. Tämä on hyvä huomioida, kun sovelluskehitys aloitetaan ja tehdään päätöksiä vanhimmasta tuetusta Android-versiosta. Jos sovelluksen kehitysaika on pitkä, on jakauma saattanut muuttua jo merkittävästi siinä vaiheessa, kun sovellus on valmis julkaistavaksi.



KUVIO 6. Käytössä olevien Android-versioiden jakauman muuttuminen vuonna 2011 (Android Developers 2011)

On tärkeä tehdä jo ennen sovelluskehityksen aloittamista rajanveto sen suhteen, millaisia laitteita ja mitä Android-versioita sovellus tulee tukemaan. Mitä laajempi tuettu laitekanta on, sitä työläämmäksi tulee sovelluksen testaaminen eri laitteilla ja käyttöliittymän sovittaminen usealle eri näyttökoolle. Toisaalta laajan laitekannan voi nähdä myös alustan vahvuutena: muutoksilla käyttöliittymään ja pienillä kompromisseilla voidaan sovellus saada toimimaan niin halvoissa peruskäyttäjän puhelimissa kuin myös hintavimmissa tablet-laitteissa.

Taulukossa 5 on esitetty eri alustaversioiden merkittävimpiä uudistuksia. Uudistukset eivät ole taaksepäin yhteensopivia, joten mikäli niitä haluaa hyödyntää, rajaa samalla pois ne käyttäjät, joiden puhelimessa on jokin aiempi Androidin versio. Merkittävimmät uudistukset sisältää Androidin versio 3.0, joka on pelkästään tablet-laitteille suunniteltu. Sen tärkein ero aikaisempaan on uudistettu UI-kehys, joka sisältää uutena konseptina Fragmentit. Activityt jaetaan pienempiin alikomponentteihin fragmenteihin, joilla jokaisella on oma samantyyppinen elinkaarensa kuin activityilla. Suurilla näytöillä activity koostuu useammista fragmenteista, mutta pieninäyttöisillä laitteilla käyttöliittymä voi koostua pelkästään yhdestä fragmentista.

TAULUKKO 5. Android-versiot ja niiden merkittävimmät uudistukset

Versio	**Tärkeimpiä muutoksia sovelluskehityksen näkökulmasta**
2.0	Päivitetty Bluetooth API, Contacts API:in lisätty tuki useammille käyttäjätileille, päivitetty Camera API
2.2	Dalvik JIT -kääntäjä (merkittävä nopeusetu tietynlaisissa sovelluksissa verrattuna aikaisempaan kääntäjään), mahdollisuus varmuuskopioida sovelluksen tiedot Googlen pilvipalveluun, mahdollisuus asentaa koko sovellus muistikortille, päivitetty Camera API
2.3	Merkittäviä parannuksia suorituskykyyn erityisesti pelikehityksessä (säikeistetty roskienkerääjä, tehokkaampi kosketustapahtumien käsittely), mahdollisuus kirjoittaa koko sovellus natiivikoodilla, NFC-tuki, päivitetty Media API.
3.0	Uusi UI-kehys (fragmentit), tuki moniydinprosessoreille, 3D grafiikkamoottori RenderScript, tuki rautakiihdytetylle 2D-piirrolle.
4.0	Versiossa 3.0 esiteltyjen uudistusten tuominen myös puhelimille.

Taistellakseen laitekannan hajanaisuutta vastaan Google julkisti asettaneensa laitevalmistajien ja operaattorien kanssa tavoitteeksi, että kaikki julkaistut laitteet saisivat uusimmat päivitykset nopealla tahdilla 18 kuukauden ajan laitteen julkaisusta (Patel 2011). Toistaiseksi ei ole kuitenkaan pystytä vielä sanomaan, miten tässä tavoitteessa on onnistuttu.

4.2 Maksamattomuuden kulttuuri

Useissa tutkimuksissa on havaittu, että Google Play ei tuota maksullisten sovellusten kehittäjille voittoa yhtä hyvin kuin Applen App Store. Sovelluskauppoja analysoiva yhtiö Distimo julkaisi tutkimuksen, josta kävi ilmi, että silloisessa Google Playn edeltäjässä Android Marketissa tuoton tekeminen on vaikeampaa kuin App Storessa (Wauters 2011).

Tutkimuksen mukaan Android Marketissa vain kaksi sovellusta oli koko sen olemassaolon aikana päässyt yli puoleen miljoonaan myytyyn kappaleeseen, kun parhaimmillaan

App Storessa yhteensä kuusi eniten myytyä sovellusta oli päässyt samoihin myyntimääriin kahdessa kuukaudessa pelkästään Yhdysvaltojen markkinoilla. Myös myytyjen pelien määrissä oli selkeitä eroja: Android Marketissa vain viisi peliä oli yltänyt yli 250 000 kappaleen myyntiin, kun App Storessa viisi myydyintä peliä olivat päässeet samoihin lukuihin jo kahdessa kuukaudessa. Tämä kaikki huolimatta siitä, että Android Marketissa olevien sovellusten määrä oli jo ohittanut tai ollut ohittamassa App Storen.

Tutkimuksessa havaittiin myös, että ladatuimpien sovellusten vaihtuvuus on suurempaa App Storessa kuin mitä se on Android Marketissa. Kuukauden tarkastelujaksolla App Storen kymmenen myydyimmän sovelluksen listalle pääsi 94 eri sovellusta, kun vastaavasti Android Marketissa määrä oli 26 sovellusta.

Tutkimusyhtiö IHS:n tekemä analyysi vuorostaan paljastaa, että Android Marketissa vuonna 2010 sovellusten tuottama liikevaihto oli hieman yli 100 miljoonaa dollaria. Luku on verrattain pieni, kun sitä vertaa App Storen vastaavaan: yli 1,7 miljardin tuotot samalla ajanjaksolla. (Schonfeld 2011.)

JOHTOPÄÄTÖKSET JA POHDINTA

Android on houkutteleva alusta sovelluskehittäjille. Se hallitsee tällä hetkellä älypuhelinmyyntiä myytyjen laitteiden määrässä mitattuna ja ennusteiden mukaan tulee hallitsemaan vielä useiden vuosien päästäkin. Tällä hetkellä erilaisia Android-laitteita on myyty yli 300 miljoonaa kappaletta ja joka päivä niitä aktivoidaan lähes miljoona lisää. Android on lyönyt itsensä läpi myös kehittyvillä markkinoilla halpojen älypuhelinten voimin, mikä tarjoaa sovelluskehittäjille uusia, mielenkiintoisia ja mahdollisesti tuottoisiakin mahdollisuuksia.

Qt-sovelluskehittäjälle siirtyminen Androidiin on kohtalaisen helppoa, koska monet käyttöliittymäluokista ja käytetyistä konsepteista ovat samankaltaisia kuin Qt-kehityksessä. Asiaa helpottaa myös se, että Java on C++:n sukulaiskieli ja vaikkei sillä olisikaan aikaisemmin ohjelmoinut, on siirtyminen siihen helppoa. Android-kehitystyökalut ovat nykyisellään useiden julkaistujen päivitysten jälkeen riittävän monipuoliset ja sisältävät monia varsinkin aloittelevaa Android-ohjelmoijaa hyödyttäviä apuohjelmia. Esimerkiksi yleisimpien ohjelmointivirheiden, käyttöliittymän suorituskykyongelmien ja muistivuotojen löytämistä helpottamaan on olemassa omat työkalunsa.

Qt-sovelluksen uudelleenkirjoittaminen Androidille vaatii kuitenkin tarkkaa suunnittelua. Esimerkiksi käyttöliittymän näkymien osalta tarvitsee miettiä, miten ne voidaan jakaa activityiksi ja fragmenteiksi. Samaten kannattaa selvittää, löytyykö jokaiselle käyttöliittymäkomponentille vastine Androidin käyttöliittymäkirjastosta. Mikäli vastaavaa komponenttia ei löydy, on harkittava kannattaako hyödyntää jotain korvaavaa komponenttia sen sijaan, että käyttäisi aikaa komponentin toteuttamiseen itse. On myös tärkeää huomioida Androidin omat käyttöliittymäkonventiot, joihin käyttäjät ovat jo tottuneet. Niitä noudattamalla voi sovelluksesta tehdä miellyttävämmän käyttää.

Android-sovelluskehitykseen liittyvistä suuremmista haasteista toinen on hajanainen laitekanta. Olemassa olevat Android-laitteet poikkeavat merkittävästi toisistaan fyysisiltä ja teknisiltä ominaisuuksiltaan sekä käyttämiltään Android-versiolta. Tämä monimutkaistaa sovelluskehitystä, koska sovelluksen toimivuus täytyy testata erikseen useilla eri laitteilla ja laiteversioilla. Lisäksi tietyissä laiteversioissa tai laitteissa esiintyy ohjel-

mointivirheitä, joihin sovelluskehittäjän on usein mahdoton varautua. Tämän vuoksi sovelluksen toimivuus eri laitteilla selviääkin kunnolla vasta, kun tuote on jo loppukäyttäjillä.

Toinen suurista haasteista on käyttäjien haluttomuus maksaa sovelluksista. Tällä hetkellä sovellusmyynti Androidin kauppapaikassa on jäänyt selkeästi jälkeen esimerkiksi Applen vastaavista luvuista. Tämä saattaa olla rajoittava tekijä niille sovelluskehittäjille, joiden tulot ovat täysin kiinni myytyjen sovellusten määrästä. Yksi vaihtoehto on mainosrahoitteisuus, mutta se ei välttämättä sovi kaikille sovelluksille.

LÄHTEET

andro-phones.com. All Android phones of 2009. Luettu 14.3.2011.
http://www.andro-phones.com/2009-android-phones.php

Android Developers. Android API Levels. Luettu 5.4.2012.
http://developer.android.com/guide/appendix/api-levels.html

Android Developers. Android Supported Media Formats. Luettu 5.4.2012.
http://developer.android.com/guide/appendix/media-formats.html

Android Developers. Application Fundamentals. Luettu 5.4.2012.
http://developer.android.com/guide/topics/fundamentals.html

Android Developers. Nine-patch. Luettu 5.4.2012.
http://developer.android.com/guide/topics/graphics/2d-graphics.html#nine-patch

Android Developers. What is Android? Luettu 14.3.2011.
http://developer.android.com/guide/basics/what-is-android.html

Android Developers. What is the NDK? Luettu 18.4.2012
http://developer.android.com/sdk/ndk/overview.html

comScore. comScore Reports January 2011 U.S. Mobile Subscriber Market Share. Luettu 14.3.2011.
http://www.comscore.com/Press_Events/Press_Releases/2011/3/comScore_Reports_January_2011_U.S._Mobile_Subscriber_Market_Share

DroidFonts.com. Droid Fonts Overview. Luettu 20.8.2011.
http://www.droidfonts.com/droidfonts/

Elgin, B. 2005. Google Buys Android for Its Mobile Arsenal. Luettu 14.3.2011.
http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm

Gartner. Gartner Says Worldwide Mobile Device Sales to End Users Reached 1.6 Billion Units in 2010. Luettu 14.3.2011.
http://www.gartner.com/it/page.jsp?id=1543014

Gartner. Gartner Says Worldwide Mobile Phone Sales to End Users Grew 8 Per Cent in Fourth Quarter 2009. Luettu 14.3.2011.
http://www.gartner.com/it/page.jsp?id=1306513

Google Inc. Android 2.3 Compability Definition. Luettu 16.10.2011
http://source.android.com/compatibility/2.3/android-2.3.3-cdd.pdf

Google Inc. Paid App Availability. Luettu 9.10.2011.
http://www.google.com/support/androidmarket/bin/answer.py?hl=en&answer=143779

Google Play. Sovellusten lähettäminen. Luettu 5.4.2012.
https://support.google.com/googleplay/android-developer/bin/answer.py?hl=fi&answer=113469&ctx=cb&src=cb&cbid=upoy03es6rii

Nesladek, C. & Bauer, G. & Fulcher, R. & Robertson, C. 2010. Android UI Design Patterns. Katsottu 27.8.2011.
http://www.google.com/events/io/2010/sessions/android-ui-design-patterns.html

Nokia Developer. Develop for the Nokia N9. Luettu 5.4.2012.
http://www.developer.nokia.com/Devices/MeeGo/

Open Handset Alliance. 2007. Industry Leaders Announce Open Platform for Mobile Devices. Luettu 14.3.2011.
http://www.openhandsetalliance.com/press_110507.html

Patel, N. 2011. Google promises Android updates will get better, new devices will get updates for 18 months. Luettu 6.11.2011
http://www.theverge.com/2011/05/10/google-promises-android-devices-updates-18-months/

Qt Development Frameworks. Internationalization with Qt. Luettu 5.4.2012.
http://doc.qt.nokia.com/4.7/internationalization.html

Qt Development Frameworks. Programming Language Support. Luettu 5.4.2012.
http://qt.nokia.com/products/programming-language-support

Qt Development Frameworks. Qt in use: Desktop. Luettu 5.4.2012.
http://qt.nokia.com/qt-in-use/target/desktop

Qt Development Frameworks. Qt Style Sheets. Luettu 5.4.2012.
http://doc.qt.nokia.com/4.7/stylesheet.html

Qt Development Frameworks. Signals and Slots. Luettu 5.4.2012.
http://doc.qt.nokia.com/4.7/signalsandslots.html

Qt Development Frameworks. Who we are. Luettu 5.4.2012.
http://qt.nokia.com/about/who-we-are

Riskedal, E. 2009. A brief history of Qt for Symbian – and a look ahead. Luettu 30.7.2011.
http://labs.qt.nokia.com/2009/12/01/a-brief-history-of-qt-for-symbian-and-a-look-ahead/

Schonfeld, E. 2011. Despite 861.5 Percent Growth, Android Market Revenues Remain Puny. Luettu 9.10.2011.
http://techcrunch.com/2011/02/21/861-5-percent-growth-android-puny/

The Apache Software Foundation. Apache License, Version 2.0. Luettu 14.3.2011.
http://www.apache.org/licenses/LICENSE-2.0.html

Vatra, B. 2011. Necessitas. Luettu 17.4.2012

http://sourceforge.net/p/necessitas/home/necessitas/

Wauters, R. 2011. Android To Surpass Apple's App Store In Size By August 2011. Luettu 9.10.2011.
http://techcrunch.com/2011/05/05/android-to-surpass-apples-app-store-in-size-in-august-2011-report-exclusive/