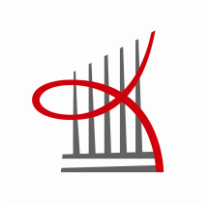


KUVAUSJÄRJESTELMÄ

Jaakko Ala-Luhtala

Opinnäytetyö
Elokuu 2012
Tietotekniikka
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautuminen

JAAKKO ALA-LUHTALA:
Kuvausjärjestelmä

Opinnäytetyö 37 sivua, joista liitteitä 0 sivua
Elokuu 2012

Kuvausjärjestelmä on järjestelmä, joka ottaa asiakkaista kuvia huvipuistolaitteessa, esittelee kuvat asiakkaille ja mahdollistaa kuvien tulostamisen ja myymisen. Vastaavat kuvausjärjestelmät ovat yleisiä isoissa huvipuistoissa maailmalla, mutta Suomessa kuvausjärjestelmien käyttö on vielä vähäisempää.

Tämän opinnäytetyön tavoitteena oli suunnitella ja luoda uusi kuvausjärjestelmä Tampereen Särkänniemi Oy:n vanhentuneen järjestelmän tilalle. Uudelta järjestelmältä vaadittiin vakautta, riippumattomuutta käyttöjärjestelmästä ja laitteistoista sekä käyttäjäystävällisyyttä.

Järjestelmän kuvausohjelma päätettiin toteuttaa käyttäen avoimen lähdekoodin Qt-ohjelmistokehystä. Qt mahdollistaa valmiin kuvausohjelman käytön Windows, Linux ja Mac OS X käyttöjärjestelmissä.

Järjestelmän suunnittelu ja ohjelman kehitys aloitettiin keväällä 2011. Ensimmäinen versio ohjelmasta valmistui 2011 Elokuussa.

Asiasanat: kuvausjärjestelmä, kuvausohjelma, qt

ABSTRACT

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

JAAKKO ALA-LUHTALA:
On-ride camera system

Bachelor's thesis 37 pages, appendices 0 pages
August 2012

On-ride camera system is a system that takes pictures of customers in amusement park rides, presents the pictures to customers and enables printing and selling the pictures. Similar on-ride camera systems are common in big amusement parks around the world, but in Finland using such systems are still uncommon.

The goal of this thesis was to design and create a new on-ride camera system for Tampereen Särkänniemi ltd to replace the old system. New system was required to be stable, not dependent on operating systems or hardware configuration and to be user friendly.

Programming part of on-ride camera system was decided to accomplish using open-source Qt-framework. Qt-framework makes possible to use the on-ride program on Windows, Linux and Mac OS X operating systems.

Designing and development of the new system was started in spring 2011. First version of the software was ready in August 2011.

Key words: On-ride, camera, qt

SISÄLLYS

LYHENTEET JA TERMIT

C++	Ohjelmointikieli
GPL	GNU General Public License
HTML	Merkintäkieli
Javascript	Ohjelmointikieli
JPEG	Joint Photographic Experts Group
Luokka	Kapseloi yhteenkuuluvat muuttujat ja metodit
Olio	Luokan instanssi
Open domain	Tekijänoikeuksien ulkopuolella
PCI	Peripheral Component Interconnect
PCI-E	Peripheral Component Interconnect Express
Qt	Käyttöliittymien kehitysympäristö
SQLite	Relaatiotietokantajärjestelmä
USB	Universal Serial Bus
X3C	World Wide Web Consortium
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

1 JOHDANTO

Kuvausjärjestelmä on järjestelmä, joka ottaa asiakkaasta kuvan huvipuistolaitteessa, esittelee kuvan asiakkaalle ja mahdollistaa kuvan tulostamisen ja myymisen. Vastaavat järjestelmät ovat yleisiä isoissa huvipuistoissa maailmalla, mutta Suomessa kuvausjärjestelmien käyttö on vähäisempää.

Tämän opinnäytetyön tavoitteena on suunnitella ja luoda uusi kuvausjärjestelmä Tampereen Särkänniemi Oy:n vanhentuneen järjestelmän tilalle. Uuden järjestelmän vaatimuksia ovat vakaus ja riippumattomuus ohjelmistoista ja käyttöjärjestelmistä.

Luvusta 2 alkaen käsitellään projektissa käytettyjä ohjelmistotekniikoita. Luvussa perustellaan tekniikoiden valintaa ja käytetään esimerkkejä havainnollistamaan niiden käyttöä.

Luvusta 3 alkaen käsitellään kuvausjärjestelmän suunnittelua ja siinä tarvittua laitteistoa. Luvussa pohditaan mitä vanhan järjestelmän ongelmista ja virheistä voidaan oppia ja kuinka toteuttaa ne toisella tavalla uudessa järjestelmässä.

Luvussa 4 esitellään järjestelmän kuvausohjelman käyttöliittymän rakentamista ja valmistuneen työn käyttöliittymää. Kappaleessa käydään myös läpi ohjelman eri vaiheita ja näytetään valmiista ohjelmakoodista osia kuinka vaihe suoritettiin. Kappaleen lopussa kerrotaan järjestelmän käyttöönotossa ilmenneistä ongelmista ja kuinka ongelmat ratkaistiin. Luvun viimeinen osa on pohdintaa mitä järjestelmään voisi lisätä tulevaisuudessa.

2 TEKNIIKAT

Tässä luvussa kerrotaan projektissa käytetyistä tekniikoista. Tarkoituksena on antaa yleiskuva tekniikoista joita projektissa tarvittiin.

2.1 Qt

Qt on alustariippumaton ohjelmistokehitysympäristö, joka käyttää tavallista C++ kieltä. Tämän lisäksi Qt käyttää myös hyväksi erityistä koodikehitintä nimeltä Meta Object Compiler ja useaa makroa tuomaan tavalliseen C++ kieleen enemmän ominaisuuksia (Qt framework 2012).

Qt:ta kehittää ohjelmistoyhtiö Digian omistama ohjelmistoyhtiö Qt Development Frameworks, aikaisemmilta nimiltään Qt Software ja Trolltech. Nokia oli ostanut Qt:n kehittäneen norjalaisen yrityksen Trolltechin kesäkuussa 2008. Maaliskuussa 2011 Nokia myi Qt:n lisensointi- ja palveluliiketoiminnan Digialle, ja elokuussa 2012 Digia osti Nokialta kaiken toiminnan liittyen Qt-kehitysympäristöön (Qt framework 2012; Qt Development Frameworks 2012; Qt kehitysympäristö 2012.).

Qt:lla toteuttettuja sovelluksia pystytään helposti siirtämään alustalta toiselle. Käytännössä tämä tarkoittaa, että yhdellä lähdekoodilla voidaan luoda sovelluksia jotka toimivat sekä Mac-, Linux- sekä Windows -käyttöjärjestelmissä. Qt:lla voidaan kehittää myös mobiilisovelluksia Symbian, Maemo ja Windows CE -käyttöjärjestelmille (Qt Products 2012).

Digian ostettua Qt Nokialta Digia kertoi että se aikoo mahdollistaa Qt:n käyttö myös Android, iOS ja Windows 8 -alustoilla (Digia to acquire Qt from Nokia 2012).

2.1.1 Qt Creator

Qt Creator on ohjelmointiympäristö (Integrated development environment (IDE)) joka sisältää kaiken tarvittavan Qt:n ohjelmoimiseen kuten: koodieditorin, debuggerin ja projektinhallintatyökalut. Samalla ohjelmistolla voidaan luoda ohjelmia työpöytä- ja mobii-

li-kohteille. Qt Creator toimii Windows, Linux ja Mac OS X käyttöjärjestelmissä (Qt Creator IDE 2012).

Qt tarjoaa myös mahdollisuuden käyttää Visual Studio Professional 2005, 2008 tai 2010 ohjelmointiympäristöä Qt Creatorin sijaan. Tämä mahdollistetaan asentamalla Visual Studioon Qt Visual Studio liitännäinen (plugin). Tämä antaa mahdollisuuden ohjelmoijille käyttää jo entuudestaan tuttuja työkaluja (Qt Visual Studio Add-in 2012).

2.1.2 Qt Designer

Qt Designer on työkalu jolla luodaan graafisia käyttöliittymiä valmiista Qt komponenteista. Komponentteja pystytään muokkaamaan, yhdistelemään ja luomaan kokonaan uusia. Qt Designerilla luodut käyttöliittymät integroituvat koodiin ja komponentteihin pystytään tekemään dynaamisesti muutoksia koodista käsin (Qt Designer 2012).

2.1.3 Signal-slot -mekanismi

Qt:n keskeinen ominaisuus on signal-slot kommunikointi, joka helpottaa suuresti olioiden välistä kommunikointia. Normaalisti olioiden välinen kommunikointi voitaisiin esimerkiksi suorittaa lähettämällä olion *raportointi*-osoitin toisen olion *tulostin*-funktiolle *tulosta(Raportointi *raportinti_)*, joka kutsuisi osoittimen kautta olion *raportointi*-funktiota *kirjaaTulostusOnnistuneeksi()*. Tällöin Tulostin-luokan täytyisi tietää Raportointi-luokan olemassaolosta.

```
Raportointi *raportointi = new Raportointi();
Tulostin *tulostin = new Tulostin();
tulostin->tulosta(raportointi);
```

KOODI 1 C++ Esimerkki: Kutsukoodi

Koodissa 1 esitellään olioiden luominen C++ kielellä. Raportointi-luokan *raportointi*-olion osoitin annetaan parametrina *tulostin*-olion funktiolle *tulosta()*.

```
#include "raportointi.h"
class Tulostin
{
public:
    void tulosta(Raportointi *raportointi_);
    bool teeTulostus();
};
```

KOODI 2 C++ Esimerkki: Tulostin-luokan otsikkotiedosto

Koodi 2 esittelee C++ esimerkkinä Tulostin-luokan otsikkotiedoston. Otsikkotiedostoon täytyy sisällyttää Raportointi-luokan otsikkotiedosto ja *tulosta()*-fuktiolle täytyy kertoa, että se saa parametrina Raportointi-luokan olion.

```
Tulostin::tulosta(Raportointi *raportointi_)
{
if(teeTulostus())
    raportointi_->kirjaaTulostusOnnistuneeksi();
}
```

KOODI 3 C++ Esimerkki: Tulostin-luokan toteutustiedosto

Koodi 3 esittelee C++ esimerkkinä Tulostin-luokan toteutustiedoston. Tulostuksen onnistuttua kutsutaan *raportointi*-olion *kirjaaTulostusOnnistuneeksi()*-funktiota. Koodissa 4 esitellään kutsuttava funktio Raportointi-luokan otsikkotiedostossa.

```
class Raportointi
{
public:
    void kirjaaTulostusOnnistuneeksi();
};
```

KOODI 4 C++ Esimerkki: Raportointi-luokan otsikkotiedosto

```
Raportointi *raportointi = new Raportointi();
Tulostin *tulostin = new Tulostin();
tulostin->tulosta();
QOb-
ject::connect(tulostin,SIGNAL(tulostusOnnistui()),raportointi,SLOT(kir
jaaTulostusOnnistuneeksi()));
```

KOODI 5 Qt Esimerkki: Kutsukoodi

Koodi 5 esittelee Qt esimerkkinä kuinka *tulostin*-oliolle ei tarvitse lähettää osoitinta *raportointi*-olioon. QObjectin *connect*-funktio liittää *tulostin*-olion signaalin *tulostusOnnistui() raportointi*-olion *kirjaaTulostusOnnistuneeksi()* slottiin.

```
class Tulostin : public QObject
{
Q_OBJECT
public:
    void tulosta();
    bool teeTulostus();
signals:
    void tulostusOnnistui();
};
```

KOODI 6 Qt Esimerkki: Tulostin-luokan otsikkotiedosto

Koodi 6 Qt esimerkissä Tulostin-luokka periytetään QObject-luokasta tämän saaden käyttöönsä Qt:n Signal-slot mekanismin. Luokkaan lisätään *tulostusOnnistui()*-signaali.

```
Tulostin::tulosta()
{
if(teeTulostus())
    emit tulostusOnnistui();
}
```

KOODI 7 Qt Esimerkki: Tulostin-luokan toteutustiedosto

Koodi 7 esittelee Qt esimerkin mukaisen Tulostin-luokan toteutustiedoston. Erona Koodi 3 esimerkkiin *tulosta()*-funktio antaa onnistuttuaan *tulostusOnnistui()*-signaalin.

```
class Raportointi : public QObject
{
Q_OBJECT
 public slots:
    void kirjaaTulostusOnnistuneeksi();
};
```

KOODI 8 Qt Esimerkki: Raportointi-luokan otsikkotiedosto

Koodi 8 esittelee Qt esimerkkinä Raportointi-luokan otsikkotiedoston. Esimerkissä Raportointi-luokka on periytetty QObject-luokasta, näin ollen saaden käyttöönsä signal-slot mekanismin. Funktio *kirjaaTulostusOnnistuneeksi()* on esitelty slottina, johon signaali voi yhdistää itsensä.

Qt esimerkissä (Koodi 5 – 8) luokkien ei tarvitse tietää toisistaan mitään. Luokkia ja funktioita voidaan luoda tarvitsematta ajatella kenelle tietoja tarvitsee välittää tai saako vastaanottaja edes tiedon perille (Qt Signals & Slots 2012).

Koodi 3 C++ koodissa ohjelma kaatuisi, mikäli *raportointi*-oliota ei olisikaan olemassa. Vastaavasti koodi 7 Qt esimerkissä ohjelma ei välitä jos kukaan ei käsittele annettua signaalia.

2.2 XML

XML eli Extensible Markup Language on merkintäkieli jolla mahdollistetaan tiedon varastointi muotoon, joka on sekä ihmisluettava että koneluettava. XML:n on kehittänyt X3C (World Wide Web Consortium). XML käytetään usein välittämään jäsenneltyä tietoa ohjelmien-, ihmisten- ja tietokoneiden -välillä (Harold & Means 2004, 3; W3C XML 2012.).

```
<?xml version="1.0"?>
<?xml-stylesheet href="Raportti.xsl" type="text/xsl" ?>
<Raportointi>
 <Vuodet vuosi="2012">
  <Kuukaudet kuukausi="4">
   <Paivat paiva="16">
    <Tulosteet>
     <Tuloste ID="658">
      <Aika>
       <HH>14</HH>
       <MM>59</MM>
       <SS>19</SS>
      </Aika>
      <KuvaNRO>0001</KuvaNRO>
     </Tuloste>
    </Tulosteet>
   </Paivat>
  </Kuukaudet>
 </Vuodet>
</Raportointi>
```

KOODI 9 Esimerkki XML-tiedostosta

Koodi 9 näyttää esimerkkinä XML-dokumentin. Dokumentti koostuu prologista ja elementeistä. Ylin rivi on prologi joka sisältää XML-version. Seuraava rivi on viittaus XSL-tyylitiedostolle. Ensimmäinen alkava elementti <Raportointi> on dokumentin juurielementti ja seuraavat sisäkkäiset elementit ovat tämän lapsielementtejä. Elementti voi sisältää tekstiä tai muita elementtejä ja attribuutteja. Elementti päättyy kun elementille vastaava lopetusmerkki löytyy. Elementti aloitetaan < >-merkeillä ja päätetään </ > -merkeillä (Harold & Means 2004, 14 – 18; XML 2012.).

Elementeillä voi olla myös attribuutteja. Attribuutti kiinnitetään elementin pariksi aloitusmerkkiin. Koodi 9 esimerkistä nähdään esimerkiksi, että Vuodet elementillä on attribuutti vuosi, jonka arvona on 2012. Sama tieto voitaisiin esittää myös lapsielementtejä

käyttäen. XML ei määrittele tiettyä sääntöä milloin käyttää lapsielementtejä ja milloin attribuutteja (Harold & Means 2004, 16 – 17).

XML-formaatille on luotu lista oikeellisuussääntöjä joilla pystytään tarkistamaan, että dokumentti on hyvin muodostettu. Muutamia oikeellisuussääntöjä on (Harold & Means 2004, 14 - 15; XML 2012.):

- Dokumentissa saa olla vain yksi juurielementti. Koodi 9 esimerkissä se on oppilas.
- Aloitusmerkit ja lopetusmerkit täytyy olla oikein sisäkkäin. Esimerkiksi <auto> <moottori></auto></moottori> olisi virheellisesti sisäkkäin. Moottori ei voi sisältää auton lopetusmerkkiä ja moottorin lopetusmerkki täytyy olla auton sisällä.
- Elementtien nimet eivät saa sisältää erikoismerkkejä ja elementit eivät saa alkaa -, . tai numeerisella arvolla
- Elementit ovat merkkikokoriippuvaisia. Jos elementti alkaa merkillä <auto> se ei voi loppua merkillä </AUTO> .

2.2.1 XSLT

XSLT eli Extensible Stylesheet Language Transformations on XML-pohjainen merkintäkieli jota käytetään XML-dokumenttien muunnoksissa. XSLT ei muuta alkuperäistä tiedostoa vaan luo uuden muunnetun tiedoston. XSLT mahdollistaa esimerkiksi XML-tiedoston tiedon esittämisen HTML- tai teksti-muodossa (Harold & Means 144; W3C XSLT 2012.).

2.3 SQLite

SQLite on täysin ilmainen relaatiotietokantajärjestelmä, joka on suunniteltu käytettäväksi projekteissa joissa on tärkeää, että tietokanta on helppo hallita, toteuttaa ja ylläpitää. Lisenssi on tekijänoikeuksien ulkopuolella (public domain), joten sitä saa vapaasti muokata ja levittää.

SQLiten etu isompiin tietokantoihin verrattuna on, että SQLite ei vaadi erillisiä servereitä tai konfigurointeja. Tietokantaa voi ajaa joko laitteen muistissa tai yhdessä tiedos-

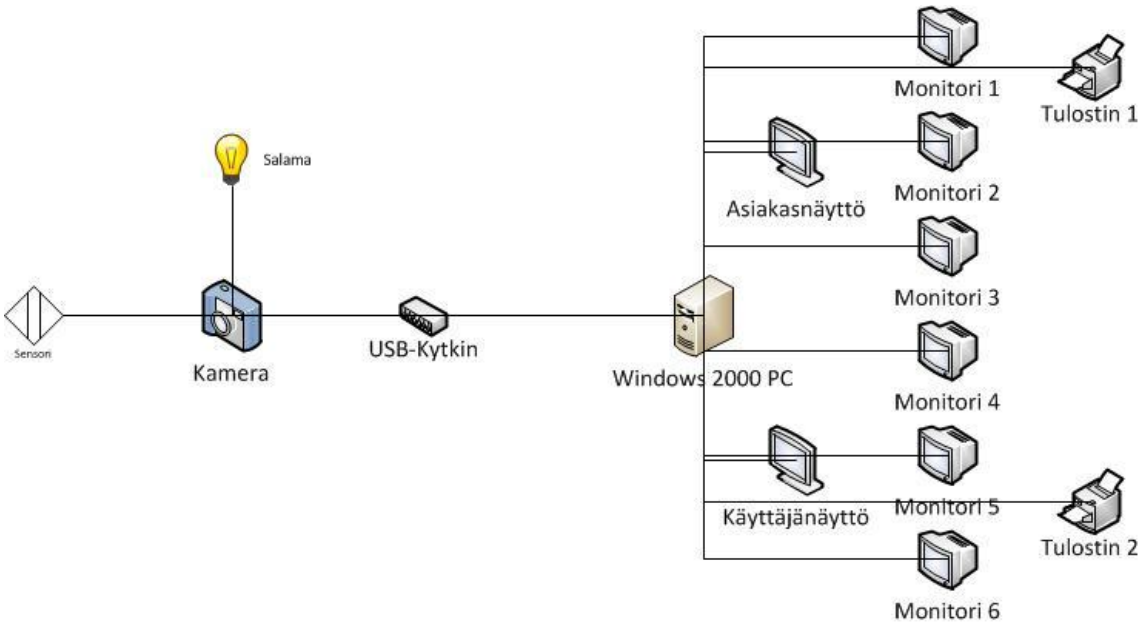
tossa. SQLite on suosittu tietokantajärjestelmä laitteissa joissa on rajallinen määrä muistia käytettävänä; kuten puhelimet ja mp3-soittimet (About SQLite 2012).

3 KUVAUSJÄRJESTELMÄ

Kuvausjärjestelmä on järjestelmä joka ottaa asiakkaista kuvia huvipuistolaitteissa, kuten vuoristoradoissa ja tämän jälkeen esittää ne asiakkaalle. Kuvausjärjestelmässä kamera sijoitetaan siten, että asiakkaasta otetusta kuvasta tulisi mahdollisimman spontaani. Kuvausjärjestelmän yksi ominaisuus on se, että asiakas voi ostaa kuvan itselleen matkamuistoksi. Vastaavanlaiset järjestelmät ovat yleisiä isoissa huvipuistoissa maailmalla. Tässä työssä käytetty sana “kuvausjärjestelmä” on vapaa käännös englanninkielisestä ”On-ride camera system” -nimestä.

3.1 Vanha järjestelmä

Huvilaitteen liikkuessa sensorin lävitse antaa sensori herätteen kameralle ja kamera salamalle. Kuva otetaan ja siirretään kameran hallintaan tarkoitetulla ohjelmalla eteenpäin tietokoneelle (Kuva 1).



KUVA 1 Vanha kuvausjärjestelmä

Vanhassa järjestelmässä (KUVA 1) otetut kuvat näytettiin asiakkaille kuudella eri monitorilla. Otetuissa kuvissa luki kuvan numero, jolla asiakkaat osasi pyytää oman kuvansa tulostusta. Käyttäjä pystyi laittamaan asiakkaan pyytämän kuvan näkyviin asiakas-

näytölle, jotta asiakas pystyi varmistamaan, että kuva on hänen. Lisäksi käyttäjälle oli oma monitorinsa, jolla kuvien myyminen pystyttiin hoitamaan.

Järjestelmän kuvausohjelma oli vahvasti riippuvainen ympärillä olevista laitteista. Kameran tai tulostimien vaihtuessa myös ohjelman lähdekoodia tarvitsi muuttaa. Kuvausohjelmaan oli lisäksi jätetty toimenpiteitä jotka olivat epäkäyttäjäystävällisiä. Päivän alussa käyttäjän piti nollata tekstitiedostosta kuvalaskuri, jotta numerointi alkaisi taas numerosta 0000. Päivän lopuksi käyttäjän täytyi tulostaa myydyistä kuvista raportti erilliselle paperille, joka syötettiin tulostimen ohisyöttölaatikkoon.

Ongelmaksi aiheutui myös tietokoneen komponenttien ja laitteiston ikääntyminen. Komponentit ja laitteisto toimivat ajureiden rajoitteiden takia pelkästään Windows 2000 -käyttöjärjestelmällä. Tietokone tarvitsi myös neljä PCI-näytönohjainta ja PCI-E teknologian syrjäyttäessä PCI-näytönohjaimet, supistui myös niiden saatavuus.

3.2 Kamera

Kamerana voi toimia mikä tahansa kamera kunhan kamera tukee sensorin avulla tehtävää laukaisua ja ulkoista salamaa. Lisäksi kameraan on oltava saatavilla jonkinlainen hallintaohjelma joka mahdollistaa kuvien siirtämisen kamerasta tietokoneelle.

Digitaalisen järjestelmäkameran ongelmaksi muodostuu sulkijan kestävyys. Kuvia saatetaan ottaa yli 1000 päivässä, joka tarkoittaa n. 120 000 kesäkauden aikana. Kameran valinnassa on siis kiinnitettävä huomiota sulkijan kestävyyteen ja vaihto-osien saatavuuteen.

Kameran hallintaan käytetään erillistä ohjelmaa. Ohjelma hoitaa kameran ohjaamisen, kuten sulkimen, valoisuuden ym. säädöt. Lisäksi ohjelma kuuntelee kameran sensoria ja uuden kuvan saapuessa voi se esimerkiksi tallentaa sen muistikortille tai lähettää eteenpäin tietokoneelle. Kameran hallintaa on vaikea integroida itse ohjelmaan koska spesifikaatiot vaihtuvat kameran valmistajien ja jopa saman valmistajan mallien kesken.

Käytettävä kameranhallinta-ohjelma riippuu kamerasta mitä sillä ohjataan. Tunnettuja ilmaisia ohjelmia esimerkiksi Canon-merkkisten kameroiden hallintaan on Canon Re-

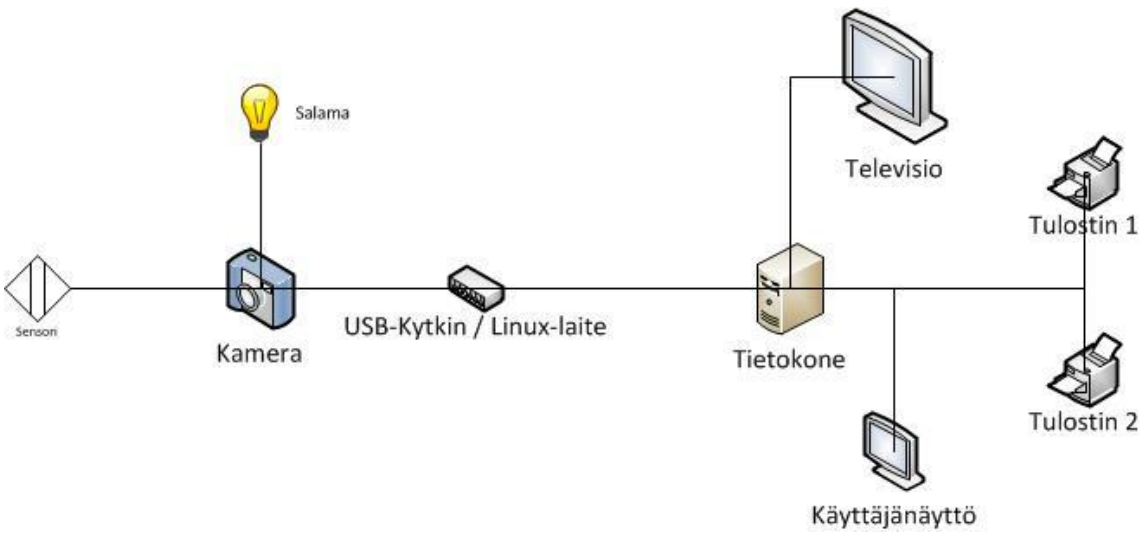
moteCapture vanhemmille kameroille ja EOS Utility uudemmille. Linuxille on olemassa gPhoto2 niminen komentorivipohjainen-ohjelma joka tukee yli 1400 kameraa (Libgphoto2 2012). gPhoto2 käyttö voidaan mahdollistaa Windows-ympäristössä esim. virtuaalikoneella.

3.3 Uuden järjestelmän suunnittelu

Kuvausjärjestelmän suunnittelu aloitettiin pohtimalla, mikä nykyisessä järjestelmässä ei toimi ja miten parhaiten ehkäistään, että uutta järjestelmää ei tarvitsisi vaihtaa samankaltaisista syistä.

Uusi järjestelmän kuvausohjelma päätettiin toteuttaa siten, että se olisi käyttöjärjestelmästä, komponenteista ja ympärillä olevasta laitteistosta riippumaton. Lähtökohtana pidettiin, että sitä voisi käyttää jopa kannettavalla tietokoneella ja sen pitäisi vaatia mahdollisimman vähän tietokoneelta resursseja.

Vanhasta järjestelmästä säilytettiin sama tekniikka joilla kuvia otettiin. Sensoriin, salamaan ja kameraan ei koskettu. Uusi järjestelmä (Kuva 2) päätettiin toteuttaa siten, että vanhan järjestelmän kahdeksan näyttöä supistetaan kahteen; yhteen isoon televisioon hoitamaan kuvien näyttäminen asiakkaille ja yhteen käyttäjänäyttöön kuvausohjelman käyttöä varten.



KUVA 2 Uusi kuvausjärjestelmä

3.3.1 USB-kytkimen korvaaminen

Kameran liittäminen tietokoneeseen ei onnistu pelkästään USB-kaapelia käyttäen. USB kaapelin standardin mukainen maksimipituus on viisi metriä. Tässä projektissa väli kameran ja tietokoneen välillä on noin 70 metriä. Tätä varten kaapelin ja tietokoneen välille on laitettu kuvan 1 ja kuvan 2 mukaisesti USB-kytkin, joka jakaa USB-yhteyden CAT-6 kaapelia pitkin tietokoneelle.

USB-kytkimien ongelmaksi muodostuu niiden korkea hinta ja huonot siirtoyhteydet. Laitteet tukevat usein vain USB 1.1 -standardia, jonka tiedonsiirtonopeus on vain 12 Mbit/s. Uudemman USB 2.0 -standardin tiedonsiirtonopeus on 480 Mbit/s (USB 2012).

Vaihtoehto USB-kytkimelle olisi tietokone, joka voisi hoitaa kameran hallinnan ja ohjelma vain jakaisi verkkolevyn tietokoneelle jolla kuvausohjelmaa käytettäisiin. Tietokoneet ovat vain liian isoja ja kömpelöitä käytettäväksi paikassa missä kamera on kiinni. Ohikulkevat laitteet aiheuttavat paljon tärinää joten tulisi myös kestävyysongelmia.

Vaihtoehtona USB-kytkimelle löytyi Linux-pohjainen reititin joka hoitaa kameran hallinnan käyttäen Linuxille saatavaa gPhotoa. Reititin saa verkkolevyjakona kuvan 2 tietokoneen ja tallentaa tulevat kuvat sinne.

3.4 Versionhallinta

Versionhallinta on tapa jolla pidetään kirjaa tiedostoihin tehdyistä muutoksista. Jokainen muutos kirjataan ylös ja se tallennetaan omana versionaan. Myös vanhat versiot tiedostoista säilytetään. Versionhallinnan tärkeys tulee erityisesti esille projekteissa joissa on useita ihmisiä työstämässä samoja tiedostoja (Revision control 2012).

Tässä projektissa käytettiin versionhallinnassa ilmaista TortoiseSVN työkalua. Työkalulla voidaan liittyä joko ulkoiseen versionhallintaa tarjoavaan palveluun tai luoda oma paikallinen versionhallintakansio. Kansioihin ja tiedostoihin tehdyt muutokset tallennetaan versionhallintaan. Ohjelma pystyy vertailemaan kahta eri versiota ja näyttämään miten versiot poikkeavat toisistaan.

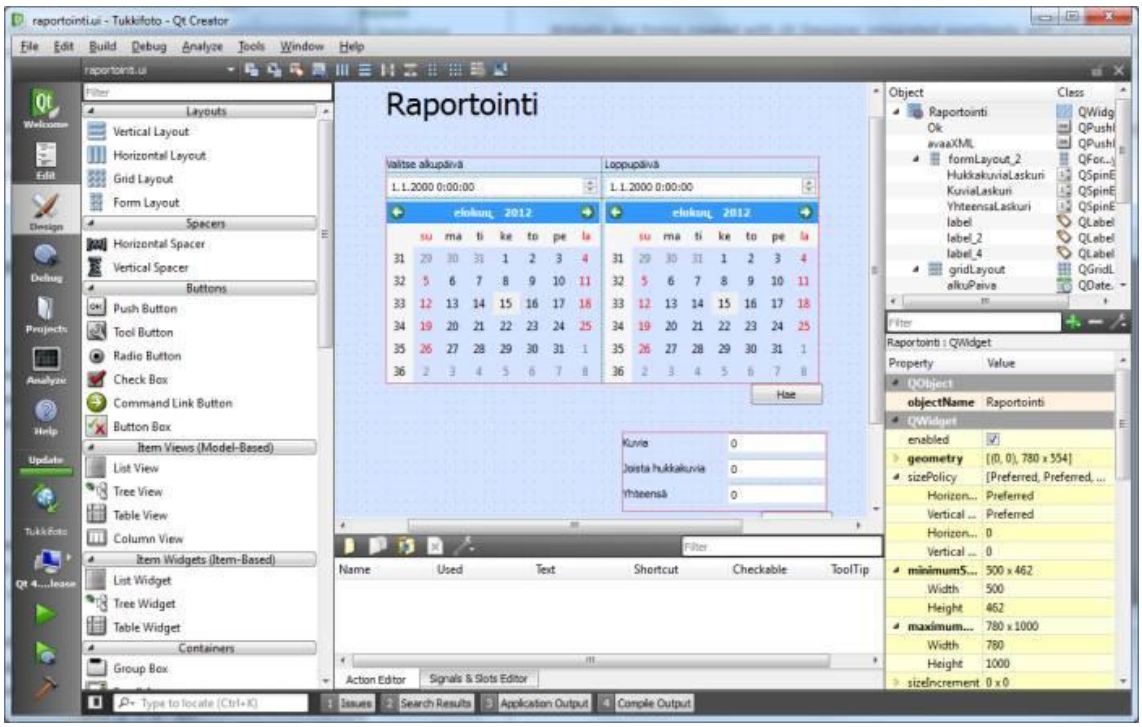
4 KUVAUSOHJELMA

Kuvausohjelma on järjestelmän tärkein osa. Ohjelma prosessoi kameralta tulevat kuvat ja esittää ne televisionäytöllä (kuva 4) asiakkaille. Käyttäjä myy kuvia ja hallitsee ohjelmaa omalla käyttäjänäytöllään (kuva 3). Ohjelma pitää sisällään myös raportoinnin myydyille kuville (kuva 7). Ohjelmointiympäristöksi valittiin Qt, koska tätä työtä tehdessä se oli yksi harvoista ilmaisista ja käyttöjärjestelmästä riippumattomista nykyaikaisista ohjelmointiympäristöistä.

4.1 Käyttöliittymä

Ohjelman käyttöliittymä koostuu käyttäjänäytöstä, televisionäytöstä, asetusikkunasta ja raportointi-ikkunasta.

Käyttöliittymä on rakennettu täysin käyttäen Qt Designeria. Qt:ssa ohjelmoijalla on vaihtoehto rakentaa käyttöliittymä käyttäen Qt Designeria jolloin ohjelmoija pystyy jo rakennusvaiheessa näkemään miltä käyttöliittymä tulee näyttämään suoritusvaiheessa (Qt Designer 2012). Esimerkkinä kuvassa 3 on Raportointi-ikkunan suunnittelua Qt-Designerilla. Vasemmalla puolella kuvassa näkyy lista käyttöliittymäkomponenteista joita voi käyttää oman käyttöliittymän toteuttamiseen. Vaihtoehtoinen tapa on luoda käyttöliittymäkomponentteja ohjelmakoodissa jolloin mitään näkyviä komponentteja ei ole ennen kuin ohjelma ajetaan. Koodissa 10 on esimerkkinä näytetty kuinka komponentin voisi luoda käyttämättä Qt-Designeria. Vaihtoehtoista tapaa voidaan käyttää esimerkiksi komponenteissa joiden ei ole tarkoituskaan aina näkyä ruudulla vaan tietyn toiminnon suorittaminen toisi komponentin hetkeksi näkyviin.



KUVA 3 Raportointi-ikkunan suunnittelua Qt-Designerilla

```
QLabel *label = new QLabel(this);
label->setText("Hei Maailma");
label->show();
```

Koodi 10 Esimerkki komponentin luomisesta koodissa

4.1.1 Käyttäjänäyttö

Käyttäjänäyttö (kuva 4) on näyttö, joka näkyy ainoastaan käyttäjälle ja jonka kautta hän hallitsee ohjelman toimintoja. Käyttäjänäyttöön on sisällytetty kaikki tarvittavat toiminnot pisteellä työskentelyyn, kuten kuvien esikatseleminen, esillepano, tulostaminen ja raportointi. Käyttäjänäyttö on pyritty pitämään mahdollisimman yksinkertaisena, koska kuvausohjelman käyttäjinä toimivat pääosin kesätyöntekijät, joilla ei ole välttämättä osaamista tietokoneiden käyttämisestä.

KUVA 4 Käyttäjänäyttö

Kuvassa 4 nähdään kahdeksan pientä ikkunaa vierekkäin. Ikkunoissa näytetään kahdeksan viimeksi tullutta kuvaa. Ikkunoiden vieressä on isompi ikkuna, joka näyttää mikä on televisioruudulla ”Asiakasnäytöllä”. Vasemmalla puolella käyttäjänäyttöä näkyy kuvien listaus ja kuvien esikatselu-ikkuna. Esikatselu-ikkuna näkyy vain käyttäjälle.

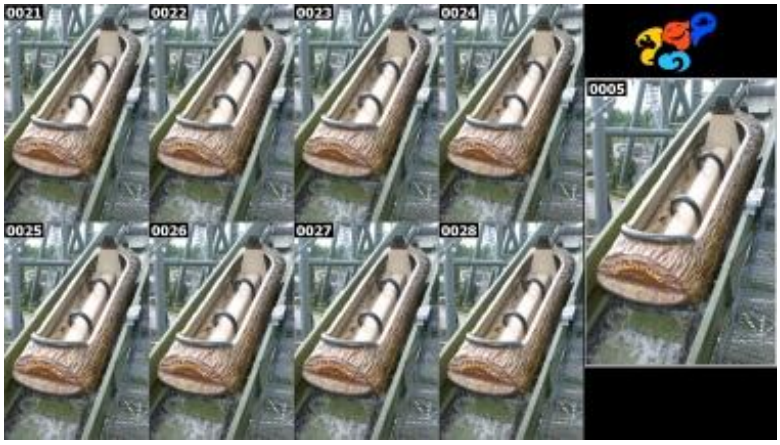
Käyttäjänäytöstä myyjä pystyy:

- selaamaan kuvia käyttäen hiirtä / näppäimistöä
    - Jokaisesta kuvasta näkyy kuvan nimi ja tieto milloin kuva on otettu. Asiakas saattaa palata myöhemmin noutamaan kuvaansa ja mikäli hän ei muista kuvan numeroa voi myyjä etsiä kuvaa ajan perusteella.
- asettamaan kuvan isommaksi televisionäytölle
    - Asiakkaan kertoessa, että hän haluaa tietyn kuvan voi myyjä varmistaa asiakkaalta, että kuva on oikea laittamalla sen näkyviin ”Asiakasnäytölle”.
- tulostamaan kuvia
- poistamaan kuvia näkyvistä
    - Joskus on tarpeellista poistaa esim. asiaton kuva näkyvistä. Ohjelma ei poista kuvaa tietokoneelta, mutta piilottaa sen näkyvistä. Kuvan saa takaisin näkymään painamalla käyttäjänäytöstä ”Päivitä kuvat”-painiketta.
- poistamaan kaikki kuvat tietokoneelta
    - Edellisen päivän kuvia ei ole tarpeellista säilyttää.
- lisäämään raporttiin merkinnän epäonnistuneesta kuvasta
    - Jos asiakas päättää olla ostamatta kuvaa lisätään tieto, että kuva oli epäonnistunut.
- avaamaan raportit-ikkunan

- avaamaan asetukset-ikkunan
- päivittämään kuvat
    - Kuvien päivitystä ei käytännössä tarvitse ikinä tehdä. Päivitys tehdään jos esimerkiksi poisti vahingossa kuvan näkyvistä ja kuva halutaan takaisin näkyviin.
- lopettamaan ohjelman

4.1.2 Televisionäyttö

Televisionäyttö on ikkuna (kuva 5), joka koostuu kahdeksasta pienemmästä ruudusta ja yhdestä isosta. Televisionäytöllä näkyy myös Särkänniemen logo, jonka ohjelma lataa automaattisesti käynnistyessään. Logon pystyy halutessaan vaihtamaan.



KUVA 5 Televisionäyttö

Televisionäyttöä esitetään televisioruudulla. Televisioruudun koolla ei väliä, mutta television täytyy tukea HD-resoluutiota.

Televisionäytön idea on automaattisesti esittää kameralta tulevia kuvia asiakkaille. Asiakkaat näkevät ruudulta oman kuvansa ja numeron kuvan vasemmassa yläreunassa. Numeron perusteella he voivat pyytää käyttäjää tulostamaan halutun kuvan. Käyttäjä pystyy siirtämään käyttäjänäytöstä minkä tahansa kuvan suuremmaksi yhteen isompaan ruutuun televisionäytölle. Isommalta ruudulta asiakas voi todeta kuvan olevan hänen.

4.1.3 Asetukset

Asetukset-ikkunasta käyttäjä pystyy vaikuttamaan ohjelman toimintaan. Asetukset-Ikkuna koostuu kahdesta välilehdestä: perusasetuksista ja lisäasetuksista. Perusasetuksista (kuva 6) käyttäjä voi valita käytettävät tulostimet ja poistaa tulostimen käytöstä. Tulostimet valitaan listavalikosta. Listavalikko näyttää automaattisesti kaikki järjestelmään asennetut tulostimet.

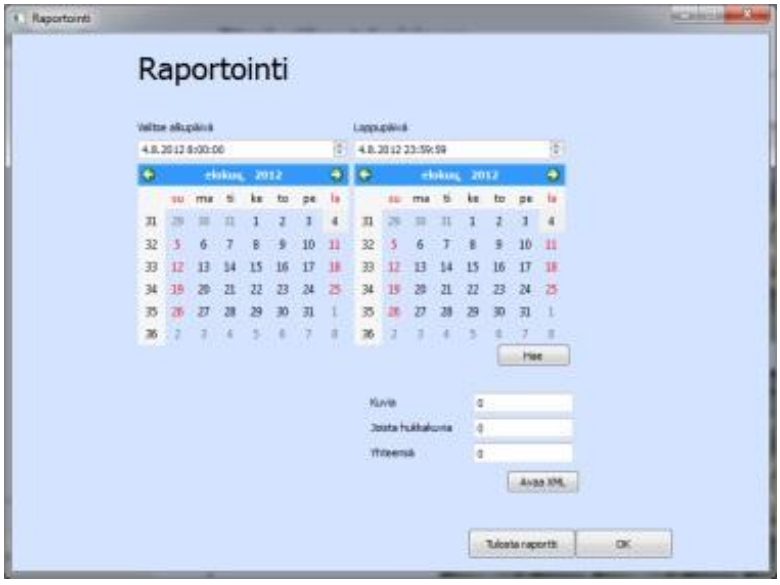
KUVA 6 Asetukset-ikkuna, perusasetukset

Lisäasetukset-ikkunasta (kuva 7) käyttäjä voi asettaa polun kameraohjelmalle, asettaa kuvan kääntämisen päälle ja valita kuvan kääntöasteen.

KUVA 7 Asetukset-ikkuna, lisäasetukset

4.1.4 Raportointi

Raportointi-ikkuna (kuva 8) näyttää käyttäjälle tietoa myydyistä kuvista ja mahdollisista hukkakuvista. Käyttäjällä on mahdollisuus tehdä haku tietokantaan millä tahansa aikavälillä joko kalenteria käyttäen tai syöttämällä haluttu päivämäärä tekstikenttään. Oletuksena ohjelma aukaisee kyseisen päivän raportin. Ikkuna sisältää myös ”Avaa xml”-painikkeen. Painike avaa raportin XML-muodossa näyttäen tiedon myös yksittäisistä tulostuksista.



KUVA 8 Raportointi-ikkuna

4.2 Kuvankäsittely

Ohjelma on rakennettu siten, että kameralta ja sen ohjelmistolta vaadittaisiin mahdollisimman vähän. Kamera ja sulkija hajoavat tasaisin väliajoin ja varmuutta seuraavan kameran ominaisuuksista ei ole. Tämän vuoksi ohjelmaan on rakennettu toimintoja jotka varmistavat, että ohjelma toimii kameran ominaisuuksista huolimatta. Vaatimuksina on, että kuvasuhde pysyy samana ja kamera tallentaa kuvat JPEG-muodossa.

Ennen kuin kuva voidaan näyttää käyttäjänäytöllä ja televisionäytöllä tarvitsee se nimetä uudelleen ja kääntää. Kuva nimetään uudelleen jotta asiakkaiden näkemä kuvanumero olisi mahdollisimman selkeä. Asiakkaiden näkemä kuvanumero muodostuu kuvan

tiedostonimestä poistettuna tiedoston loppupääte. Kuvanumero alkaa numerosta 0001, maksiminumeroa ei ole määritelty.

Koodissa 11 ohjelma asetetaan seuraamaan kansion toimintaa, johon oletetaan kameralta saapuvien kuvien tulevan. Mikäli kansiossa tapahtuu jotakin, tarkistaa ohjelma saapuiko uusia kuvia. Uuden kuvan saapuessa kuva käännetään (tarvittaessa).

```
QFileSystemWatcher *KansionKatselijaOlio = new QFileSystemWatcher();
KansionKatselijaOlio->addPath("KameraltaTulevatKuvat");
con-
nect(KansionKatselijaOlio,SIGNAL(directoryChanged(QString)),this,SLOT(
Kansiossatapahtui()));
```

KOODI 11 Kansion asettaminen seuraamista varten

4.2.1 Kuvan kääntäminen

Kameran hallintaohjelma ei välttämättä osaa kääntää JPEG-muotoista kuvaa. Tämän vuoksi kuvan kääntäminen tarvitsi tehdä itse kuvausohjelmassa.

Kuvausohjelman ensimmäisessä versiossa kuvan kääntäminen oli toteutettu Qt:n omia kirjastoja käyttäen. Pian selvisikin, että jo pelkkä JPEG-muotoisen kuvan lataaminen ja uudelleen tallentaminen huonontaa kuvanlaatua radikaalisti. Kuvan laatu pieneni n. 50 % käännöksen yhteydessä. Ratkaisuna oli käyttää C-kirjastoa joka pystyi suorittamaan JPEG-muotoisen kuvan kääntämisen ilman, että laatu huononee lainkaan.

FreeImage on avoimen lähdekoodin C -kirjasto. Kirjasto sisältää lukuisia kuvanmuokkauksiin tarkoitettuja funktioita. Kirjastoa pystyy käyttämään joko kaupallisella lisenssillä tai GNU General Public (GPLv2 / GPLv3) -lisenssillä. GPL-lisenssin käyttö tarkoittaa käytännössä, että ohjelmassa täytyy mainita kirjaston käytöstä. Lisäksi lisenssi tulee jakaa ohjelman mukana esim. tekstitiedostona. Kuvausohjelman asetuksiin kirjoitettiin myös maininta, että ohjelma käyttää FreeImage-kirjastoa kuvan kääntämisessä (kuva 7).

FreeImagesta löytyy valmis ladattava ajonaikainen kirjasto osoitteesta http://freeimage.sourceforge.net/download.html. Koodissa 12 Qt:n .pro tiedostoon lisätään merkintä ulkopuolisista lähteistä, jotta kääntäjää osaa käyttää kirjastoa oikein.

```
win32:CONFIG(release, debug|release): LIBS += -L$$PWD/FreeImage/ -
lFreeImage
else:win32:CONFIG(debug, debug|release): LIBS += -L$$PWD/FreeImage/ -
lFreeImaged
else:unix:!symbian: LIBS += -L$$PWD/FreeImage/ -lFreeImage
INCLUDEPATH += $$PWD/FreeImage
DEPENDPATH += $$PWD/FreeImage
```

KOODI 12 Qt:n .pro tiedostoon lisätty merkintä ulkopuolisesta lähteestä

```
if(kuvanKaantoAsteina == 90)
if(!FreeImage_JPEGTransform(kuvaPolku.toLatin1(),kuvaPolku.toLatin1(),
FIJPEG_OP_ROTATE_90,true))
            return false;
```

KOODI 13 Esimerkki kuvan kääntämisestä

Kuvankääntämiseen käytettiin FreeImage-kirjaston *JPEGTransform()*-funktiota koodissa 13 nähdään esimerkki kuvan kääntämisestä 90-astetta. JPEGTransform suorittaa kuvan kääntämisen siten, että JPEG:in data järjestellään uudelleen ilman, että kuvanlaatu huononee. Normaalisti JPEG:in lataaminen aiheuttaa ensin pakkauksen purkamisen, jonka jälkeen tallennettaessa JPEG pakataan uudelleen. Jokainen lataus–tallennus pakkaa tiedostoa entisestään ja huonontaa kuvanlaatua.

4.3 Kuvan tulostaminen

Tulostamisessa käytetään kahta eri tulostinta. Käyttäjältä ei kysytä missään vaiheessa mihin tulostimeen hän haluaa tulostaa. Tulostimia vaihdellaan ohjelmallisesti vuoronperään. Tämä nopeuttaa pisteellä työskentelyä ja pienentää mahdollisia virhetulostuksia. Tulostimet valmistellaan ohjelman käynnistyessä ja tulostimia vaihdettaessa. Koodissa 14 nähdään yhden tulostimen asettaminen käyttövalmiiksi. Tulostimelle asetetaan nimi saatavilla olevista tulostimista, tulostuksen resoluutio, paperikoko ja sivumarginaalit. Sivumarginaalit luodaan jotta tulostettu kuva näkyisi oikein Särkänniemen pahvisissa kuvakuorissa.

```
QSettings asetukset("Särkänniemi", "Tukkifoto");
Tulostin1Kaytossa = asetuk-
set.value("Tulostimet/Tulostin1Kaytossa").toBool();
printteri1 = new QPrinter(QPrinter::HighResolution);
printteri1-
>setPrinterName(asetukset.value("Tulostimet/Tulostin1").toString());
printteri1->setResolution(300);
printteri1->setPaperSize(QPrinter::A5);
printteri1->setPageMargins(15,25,20,10,QPrinter::Millimeter);
```

KOODI 14 Esimerkki yhden tulostimen asettamisesta käyttövalmiiksi

Kuva tarvitsee tämän lisäksi vielä piirtää alustalle, joka lähetetään tulostimelle. Koodissa 15 luodaan QPainter-luokasta *painter*-olio jonka tallennuskohteeksi valitaan tulostin johon halutaan tulostaa. *painter*-olioon ladataan tulostettava kuva ja se skaalataan tulostimen paperille sopivaksi. Lopuksi kutsutaan *painter*-olion *end()*-funktiota joka lähettää kuvan tulostimelle.

```
QPoint ylavasen(0,0);
QPoint alaoikea(printteri1->pageRect().bottomRight()-printteri1-
>pageRect().topLeft());
QRect TulostusAlue(ylavasen,alaoikea);

painter = new QPainter;
QImage TulostettavaTukkikuva("Kuvat/" + Tiedostonimi);
    if(VaihdaTulostin())
            if(tamanHetkinenTulostin == 1)
                if(!painter->begin(printteri1))
                    return false;
            if(tamanHetkinenTulostin == 2)
                if(!painter->begin(printteri2))
                    return false;
painter-
>drawIma-
ge(TulostusAlue,TulostettavaTukkikuva.scaledToHeight(TulostusAlue.heig
ht(),Qt::SmoothTransformation));
painter->end();
```

KOODI 15 Esimerkki kuvan tulostamisesta

4.4 Raportointi

Raportointi on suurin yksittäinen osa ohjelmasta. Raportointi on tärkeää koska sitä seuraamalla voidaan selvittää esimerkiksi kuinka paljon väriä, paperia ja kuoria pisteellä kulutetaan. Lisäksi voidaan vertailla sään ja viikonpäivien vaikutusta kuvien menekkiin.

Raporttiin kerätään tieto milloin kuva tulostettiin ja mikä oli kuvan numero. Lisäksi kerätään tietoa mahdollisista hukkakuvista. Hukkakuva syntyy, mikäli asiakas jättää

kuvan ostamatta. Syitä voi olla esimerkiksi kuvan heikko laatu tai epäonnistunut tulostus.

Raportointi haluttiin pitää mahdollisimman yksinkertaisena, mutta kuitenkin niin, että tietojen käyttö olisi mahdollisimman monipuolista.

Aluksi raportit tallennettiin tekstitiedostoon, josta ne käsiteltiin luettavaksi kuvan 7 mukaiseen ikkunaan. Tapa kuitenkin rajoitti raporttien käyttöä muiden sovellusten välillä, jonka takia päädyttiin käyttämään SQLite-tietokantaa ja mahdollisuutta siirtää tiedot tietokannasta XML-formaattiin.

4.4.1 SQLite tietokannan käyttö

SQLite tietokannan käyttöön päädyttiin koska se oli nopea ottaa käyttöön eikä vaatinut erillisiä palvelimia ja käyttöönottoja. Vaihtoehtona SQLitelle olisi ollut esimerkiksi MySQL-tietokanta, mutta tämä olisi vaatinut erillisen palvelimen ja jatkuvan internetyhteyden. MySQL mahdollistaisi raporttien monipuolisemman käytön, mutta samalla rajaisi ohjelman mahdollisia käyttöpaikkoja.

Koodissa 16 esitellään yhteyden luominen tietokantaan. Qt:n QSqlDatabase-luokka sisältää valmiin rajapinnan SQL-komentojen käyttämiseen.

```
mySQLitedb = QSqlDatabase::addDatabase("QSQLITE");
mySQLitedb.setDatabaseName("Raportit\\RaportitSQLiteKanta.db");
mySQLitedb.open();
```

KOODI 16 Yhteyden avaaminen tietokantaan

Koodissa 17 tietokantaan luodaan uudet taulukot, mikäli niitä ei ole olemassa ennestään. Tulosteet- ja Hukkakuvat-taulukot luodaan identtisiksi. Kuvista halutaan tallentaa taulukkoon tiedot ID, Aikaleima epochtime-muodossa ja kuvanumero. Uusi syöte saa automaattisesti tietokannalta arvot ID- ja Aikaleima-kenttiin. Koodissa 18 taulukkoon syötetään uusi syöte.

```
QSqlQuery query;
query.exec("create table Tulosteet (T_Id INTEGER PRIMARY KEY, Aikalei-
ma DATE DEFAULT (strftime('%s','now')), Kuvanro varchar(4) DEFAULT
xxxx)");
query.exec("create table Hukkakuvat (H_Id INTEGER PRIMARY KEY,
Aikaleima DATE DEFAULT (strftime('%s','now')),Kuvanro varchar(0))");
```

KOODI 17 Taulukoiden luominen SQLite tietokantaan

```
QSqlQuery query;
if(tuloste)
 query.exec("insert into Tulosteet (Kuvanro) values('"+kuvaNro+"')");
else
 query.exec("insert into Hukkakuvat (Kuvanro) values('"+kuvaNro+"')");
```

KOODI 18 Uuden arvon lisääminen taulukkoon

Tietokannassa päädyttiin käyttämään aikaleimassa epoch-aika-tyyppistä ratkaisua. Epoch-aika on aika sekunteina, siitä mitä on kulunut ajasta 1 Tammikuuta 1970 kello 00:00:00 UTC. Lukuisat järjestelmät tukevat muunnosta epoch-ajasta ihmisluettavaan muotoon.

Tässä tapauksessa epoch-aikaa käytetään mm. koska se vie huomattavasti vähemmän tilaa tietokannasta. INT-tyyppisen epoch-ajan käsittely on myös ohjelmallisesti helpompaa kuin string-tyyppisen päivä-ajan. String-tyyppisessä aikamuodossa ohjelmalle on kerrottava mitkä arvot ovat päiviä, kuukausia, vuosia, tunteja, minuutteja ja sekunteja.

Taulukko 1 esimerkin mukaisesti nähdään, että päiväys 04.08.2012 klo 15:33:16 voidaan ilmoittaa päivä-aika muodossa lyhimmillään 14 merkin pituisena. Sama aika epoch-aikana on 4 merkkiä lyhyempi. Kokoero ei yksittäisissä arvoissa ole merkittävä, mutta kun arvoja on satojatuhansia on neljällä merkillä jo eroa.

Epoch-aika -formaatti	Päivä-aika –formaatti lyhimmillään (DDMMYYYHHMMSS)
1344083416	04082012153316
= 10 merkkiä	= 14 merkkiä

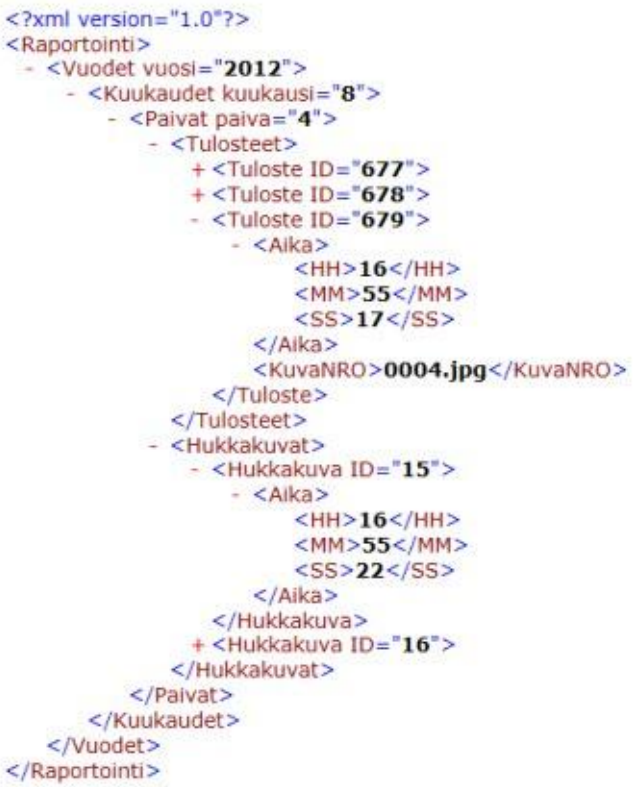
TAULUKKO 1 Esimerkki tilankäytöstä

4.4.2 XML vienti

Alun perin raportoinnissa oli tarkoitus käyttää XML-formaattia sen ihmisluettavan ominaisuuden vuoksi. Käytössä kuitenkin huomattiin, että formaatti on liian raskas jatkuvaan käyttöön. XML kuitenkin pidettiin projektissa SQLiten rinnalla mahdollistaen sekä SQLiten hyödyt että XML:n hyödyt. XML mahdollistaa tavan siirtää SQLite-tietokannan data muihin järjestelmiin, kuten Exceliin.

Joka kerta, kun käyttäjä tekee Raportointi-ikkunassa haun tietokantaan, tehdään haun tuloksista myös XML-dokumentti joka on luettavissa raportointi-ikkunan Avaa XML-painikkeen kautta.

Kuvassa 9 näytetään esimerkki luotavasta XML-dokumentista. Dokumentilla on prologi, juurielementti, lapsielementti, lapsielementin lapsielementti jne. Dokumentin elementit ja arvot ovat luotu siten, että ne olisi mahdollisimman siirrettävät järjestelmien välillä. Esimerkiksi aika-elementti sisältää tunneille, minuuteille ja sekunneille oman elementin. Tämän ansiosta voidaan esimerkiksi tiedon esityksen yhteydessä mainita, että mikäli HH-elementin arvo on pienempi kuin 16, ei tietoa esitetä.



KUVA 9 Esimerkki luotavasta XML-dokumentista

Lisäksi jotta XML-tiedosto olisi mahdollisimman luettava, luotiin sen rinnalle XSL-dokumentti, joka tekee XML-tiedostosta HTML-muotoisen dokumentin. XSL-tiedostossa yhdisteltiin sekä javascriptia ja HTML-koodia, jotta saatiin kuvan 10 näkymä.

KUVA 10 XML-tiedosto yhdistetty XSL-muotoiluun

Linkkien klikkaaminen kuvassa 10 avaa näkyviin linkin alta löytyvän tiedon. Linkin uudelleen klikkaaminen piilottaa alla olevan tiedon uudelleen. Esimerkiksi voidaan tarkastella vain vuoden 2012 elokuun neljännen päivän tietoja.

Huomion arvoista on, että oletuksena selaimet eivät yleensä suostu avaamaan XML-dokumentteja johtuen tietoturva-asetuksista. Esimerkiksi Internet Explorer versio 9 pyytää käyttäjää hyväksymään ActiveX-komponentin ja Google Chrome versio 21 ei avaa XML-muotoisia dokumentteja lainkaan ilman turva-asetuksien muuttamista. Firefox versio 14 oli testauksessa ainoa selain joka suostui aukaisemaan dokumentin ilman asetuksien muuttamista.

4.5 Asetukset

Jo ohjelman suunnitteluvaiheessa päätettiin, että ohjelman halutaan toimivan ympäristöstä riippumatta. Tietokoneen osia, oheislaitteita, käyttöjärjestelmää ja muuta laitteistoa pitää pystyä vaihtamaan ilman, että ohjelmakoodia tarvitsisi kirjoittaa uudelleen.

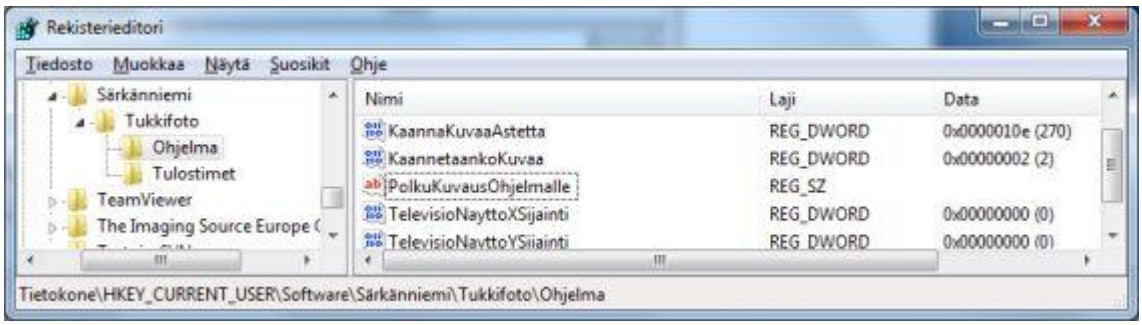
Tätä varten ohjelmaan luotiin Asetukset-luokka. Asetukset-luokka käyttää hyväksi Qt:n QSettings-luokkaa joka mahdollistaa asetuksien tallentamisen käyttöjärjestelmästä riippumatta.

Ohjelmaan voi syöttää asetuksia seuraavista tiedoista:

- Tulostin 1
- Tulostin 2
- Tulostin 1 Käytössä / Pois käytöstä
    - Joskus tulostin menee epäkuntoon, joten tällä asetuksella voi ottaa tulostimen hetkeksi pois käytöstä.
- Tulostin 2 Käytössä / Pois käytöstä
- Polku kameranhallinta-ohjelmalle
    - Tähän voi syöttää kameran hallintaan tarkoitetun ohjelman jolloin ohjelma käynnistetään ja suljetaan yhdessä kuvausohjelman kanssa. Lisäksi ohjelma seuraa pysyykö kameraohjelma päällä ja ilmoittaa jos ohjelma sattuisi esim. sulkeutumaan yllättäen.
- Käännä kuvaa kyllä/ei
    - Kamera ei välttämättä osaa kääntää kuvaa oikein. Tämän asetuksen päälle laittamalla uusi kuva käännetään aina.
- Käännä kuvaa 90- / 180- / 270-astetta
    - Asetuksella määritellään kuinka monta asetta kuvaa käännetään.

Ohjelma kerää automaattisesti asetuksiin tiedon televisionäytön sijainnista. Televisionäyttöä voi liikutella hiirellä näytöstä toiseen. Liikutuksen jälkeen ohjelma tallentaa tiedon sijainnista ja ohjelma osaa avata näytön uudelleen samassa ikkunassa uudelleenkäynnistyttyään. Käytännössä tämä tarvitsee tehdä vain kun ohjelma asennetaan uuteen tietokoneeseen.

Kuvassa 11 on näytetty kuinka asetukset tallentuu Windows-käyttöjärjestelmässä. Asetukset tallentuu rekisteriin HKCU/Software/[Yrityksen nimi]/[Ohjelman nimi]



KUVA 11 Asetukset

Linuxissa vastaavasti asetukset tallentuu tiedostoon $HOME/.config/[Yrityksen nimi]/[Ohjelman nimi].conf. Qt sulautetuissa järjestelmissä tallentaa asetukset tiedostoon $HOME/Settings/[Yrityksen nimi]/[Ohjelman nimi].conf ja Mac OS X tallentaa asetukset polkuun $HOME/Library/Preferences/com.[Yrityksen nimi].[Ohjelman nimi].plist (QSettings 2012).

4.6 Käyttöönotto

Järjestelmän suunnittelu ja ohjelman kehitys aloitettiin keväällä 2011. Ohjelman ensimmäisen version käyttöönotto aloitettiin 15.8.2011. Uusi järjestelmä tuotiin vanhan järjestelmän rinnalle, säilyttäen näin mahdollisuuden käyttää vanhaa järjestelmää jos uusi ei toimisikaan. Heti käyttöönoton jälkeen huomattiin lukuisia yhteensopivuusongelmia muun laitteiston kanssa. Vanha järjestelmä oli suunniteltu toimimaan Windows 2000 -käyttöjärjestelmän kanssa ja uusi käyttöjärjestelmä oli Windows 7. Vanhoihin laitteisiin ei löytynyt ajureita jotka toimisivat Windows 7 -käyttöjärjestelmässä. Tästä syystä käyttöjärjestelmäksi vaihdettiin Windows XP ja ajuriongelma saatiin tilapäisesti ohitettua. Ohjelman käyttöönotto kuitenkin siirtyi päivällä, kun ohjelmasta havaittiin lukuisia virheitä jotka täytyi korjata. Virheet liittyivät pääosin kuvien käsittelyyn ja kääntämiseen.

16.8.2011 saatiin seuraava versio ohjelmasta käyttöön ja kuvien myynti pystyttiin nyt hoitamaan täysin uudella järjestelmällä. Työntekijät perehdytettiin pikaisesti uuteen järjestelmään ja heitä pyydettiin päivän päätteeksi kertomaan käyttökokemuksista ja mahdollisista korjausehdotuksista. Käyttökokemuksien ja korjausehdotuksien pohjalta ohjelmasta tehtiinkin vielä elokuun 2011 aikana seitsemän uutta versiota.

Käyttäjäkokemukset olivat pääsääntöisesti positiivisia. Uudessa järjestelmässä vältyttiin tekemästä asioita joita oli joutunut tekemään vielä vanhassa järjestelmässä. Uuden järjestelmän suurimmaksi pullonkaulaksi muodostui kuvien tulostamisen hitaus. Tulostamisen hitaus johtui virheestä koodissa. Kuvat tulostettiin parhaalla mahdollisella resoluutiolla joka tarkoitti, että kuvan koko moninkertaistui matkalla ja tulostimella kesti prosessoida isokokoinen kuva. Vika korjaantui alentamalla resoluutio vastaamaan kuvan todellisia tarpeita.

Särkänniemen kesäkauden päätyttyä elokuun lopussa ei nähty tarpeelliseksi pitää enää vanhaa järjestelmää uuden rinnalla, joten vanha järjestelmä purettiin pois.

Ohjelman kehitystä jatkettiin vielä Särkänniemen talvikauden 2012 ajan. Lukuisia korjauksia ja parannuksia tehtiin kuvankäsittelyyn ja raportointiin liittyen. Järjestelmä otettiin uudelleen käyttöön Särkänniemen kesäkauden 2012 aukeamisen yhteydessä.

Tämän opinnäytetyön alueeseen ei kuulunut kamera, sensori ja salama, josta aiheutuikin, että vanha kamerajärjestelmä aiheutti jatkuvia ongelmia toimimattomuutensa vuoksi. Sensori saattoi päästää laitteita lävitseen tai huomata laitteen liian myöhään. Kamera saattoi jäädä jumiin ja vaatia, että kameran fyysistä laukaisunappia painettiin ja salama ei toiminut uuden kuvausjärjestelmän aikana kertaakaan.

4.6.1 Koulutus

Kesäkauden 2012 alussa työntekijöitä koulutettiin kuinka ohjelmaa ja järjestelmää käytetään. Pisteellä ei ole vakituista työntekijää vaan työntekijöitä voi olla monia kymmeniä kesän aikana. Tämän vuoksi vain pientä osaa työntekijöistä koulutettiin, jotka taas kouluttaisivat seuraavia työntekijöitä. Kesän alussa kattavaa työohjetta tai manuaalia ei vielä oltu saatu kirjoitettua mahdollisia ongelmia ja vikatilanteita varten. Lähinnä koska ei vielä tiedetty mitä ongelmia ja vikatilanteita piste saattaisi kohdata.

Kesän aikana kävi selväksi, että koulutus oli ollut riittämätöntä. Työntekijät jotka koulutettiin käyttämään järjestelmää, olivat aikaisempina vuosina käyttäneet myös vanhaa järjestelmää. Tämän vuoksi he eivät täysin ymmärtäneet, että uudessa järjestelmässä ei enää tarvinnut tehdä asioita joita vielä vanhassa järjestelmässä oli tarvinnut tehdä. Tästä

syystä työntekijät ja koulutettavat koulutettiin uudelleen. Uudella koulutuskerralla painotettiin vanhan ja uuden järjestelmän eroavaisuuksia. Työntekijöille jaettiin myös ohjeet työpisteellä työskentelyyn.

4.7 Jatkokehitys

Ohjelmassa ja muussa järjestelmässä on vielä varaa jatkokehitykselle. Ohjelmaan tehtiin tasaisin väliajoin pieniä korjauksia ja parannuksia, mutta isommat ideat ja ehdotukset vaativat enemmän tutkimusta eivätkä vielä valmistuneet tätä opinnäytetyötä kirjoittaessa.

Tällä hetkellä kuvia voidaan myydä ainoastaan paperisessa muodossa. Asiakkaan ostettua paperinen kuva, voidaan kuva hävittää päivän lopuksi tietokoneelta sitä säästelemättä. Kuvien myyminen esimerkiksi digitaalisessa muodossa aiheuttaisi omanlaisia haasteitaan. Kuvan toimittaminen asiakkaalle sähköpostin- tai multimediaviestin-välityksellä vaatisi jonkinlaisen varmistuksen, että asiakas todella sai kuvan itselleen. Asiakas on saattanut antaa väärän sähköpostiosoitteen / puhelinnumeron tai myyjä on kirjoittanut tiedot väärin ylös. Tämän vuoksi myydyt kuvat tarvitsisi tallentaa koneelle mahdollisia uudelleenlähetyksiä varten. Kuvien myyminen myös hidastuisi, koska asiakkaalta tarvitsisi kysyä sähköpostiosoite tai puhelinnumero pelkän kuvanumeron sijaan.

Tämän opinnäytetyön ulkopuolelle jätettiin kamera, sensori ja salama. Laitteet jätettiin tästä opinnäytetyöstä pois, koska olisivat vaatineet liikaa aikaa toteuttaa uudelleen. Ulkopuolelle jätetyt laitteet olivatkin kuvausjärjestelmän suurin ongelma. Jatkokehityksenä laitteet tullaan uusimaan tai tekemään muutoksia nykyiseen kokoonpanoon.

5 YHTEENVETO

Työn tavoitteena oli suunnitella ja toteuttaa uusi kuvausjärjestelmä Tampereen Särkänniemi Oy:n vanhan järjestelmän tilalle. Uuden järjestelmän tavoitteeksi asetettiin kuvausjärjestelmän riippumattomuus käyttöjärjestelmästä ja ympärillä olevista laitteista. Järjestelmän tulisi olla myös mahdollisimman käyttäjäystävällinen.

Tärkeimpänä tavoitteena oli olla riippumaton käyttöjärjestelmästä. Ohjelmaa tulisi pystyä ajamaan millä tahansa käyttöjärjestelmällä. Tavoitteeseen päästiin käyttämällä hyväksi Qt-luokkakirjastoja, jotka mahdollistavat käyttöjärjestelmäriippumattomuuden. Valmis ohjelma toimii Linux, Windows ja Mac OS X käyttöjärjestelmissä.

Seuraava tavoite oli olla riippumaton muista laitteista. Tämä käytännössä tarkoittaa, että ohjelman pitäisi toimia vaikka kamera, tulostimet, USB-kytkin ja näytöt vaihdettaisiin järjestelmästä. Tavoite mahdollistettiin rakentamalla kuvausohjelmaan Asetukset-käyttöliittymä, jonka kautta käyttäjä voi tehdä muutoksia ohjelman toimintaan.

Kuvausohjelma pyrittiin pitämään mahdollisimman käyttäjäystävällisenä. Uudessa kuvausohjelmassa vähennettiin käyttäjältä vaadittavaa vuorovaikutusta vanhaan kuvausohjelmaan verrattuna. Uudessa kuvausohjelmassa automatisoitiin asioita, joita oli vielä vanhassa joutunut tekemään käsin.

LÄHTEET

About SQLite. 2012. About SQLite. Luettu 04.08.2012. http://www.sqlite.org/about.html

Digia to acquire Qt from Nokia. 2012. Digia to acquire Qt from Nokia. Luettu 10.08.2012. http://digia.com/en/Home/Company/News/Digia-to-acquire-Qt-from-Nokia/

Harorld, E. & Means, S. 2004. 3. painos. XML in a Nutshell

Libgphoto2. 2012. Libgphoto2 Supported cameras. Luettu 05.08.2012. http://gphoto.org/proj/libgphoto2/support.php

Qt Creator. 2012. Qt Creator IDE and tools. Luettu 09.08.2012. http://qt.nokia.com/products/developer-tools/

Qt Development Frameworks. 2012. Wikipedia: Qt Development Frameworks. Luettu 10.08.2012. http://en.wikipedia.org/wiki/Qt_Development_Frameworks

Qt Framework. 2012. Wikipedia: Qt (framework). Luettu 10.08.2012. http://en.wikipedia.org/wiki/Qt_(framework)

Qt Kehitysympäristö. 2012 Wikipedia: Qt (kehitysympäristö). Luettu 10.08.2012. http://fi.wikipedia.org/wiki/Qt_(kehitysymp%C3%A4rist%C3%B6)

Revision control. 2012. Wikipedia: Revision control. Luettu 16.08.2012 http://en.wikipedia.org/wiki/Revision_control

Signals and Slots. 2012. Qt Reference Documentation: Signals & Slots. Luettu 04.08.2012 http://doc.qt.nokia.com/4.7-snapshot/signalsandslots.html

USB. 2012. Universal Serial Bus. Luettu 16.08.2012 http://en.wikipedia.org/wiki/Universal_Serial_Bus

W3C XML. 2012. XML Essentials. Luettu 13.08.2012. http://www.w3.org/standards/xml/core

W3C XSLT. 2012. Luettu 13.08.2012 XSL Transformations (XSLT). http://www.w3.org/TR/xslt/

XML. 2012. Wikipedia: XML. Luettu 13.08.2012 http://en.wikipedia.org/wiki/XML