HAAGA-HELIA
ammattikorkeakoulu

Passikuvausprosessin automatisointi

Martti Vaittinen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2014

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

23.11.2014

Tekijä
Martti Vaittinen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Passikuvausprosessin automatisointi

Sivu- ja liitesivumäärä
22

Opinnäytetyön otsikko englanniksi
Automating ID-picture taking process

Opinnäytetyön tavoitteena oli parantaa olemassa olevaa henkilökuvauksen liiketoimintaprosessia kokonaisuutena. Ensisijaisena tavoitteena oli luoda Microsoft Windows ympäristössä toimiva työpöytäohjelma passikuvien automaattiseen käsittelyyn poliisin lupahallinnon määräysten mukaisesti. Työssä kartoitettiin myös kuvausta liiketoimintaprosessina.

Teoriaosuudessa tarkastellaan passikuvauksen liiketoimintaprosessin muutosta julkisen vallan järjestelmämuutoksen myötä. C# -ohjelmointikielen sopivuutta projektiin tarkasteltiin GDI+ arkkitehtuurin osalta. Samoin pohdittiin laadun ja virheettömyyden merkitystä prosessissa.

Työssä pohdittiin automatisointiprosessin vaikutusta liiketoimintaan, sen jatkokehitys-mahdollisuuksia sekä ohjelman kaupallista hyödyntämistä. Ohjelman tilaaja on valokuvausalalla toimiva yritys.

Työvälineenä ohjelmointiosuudessa käytettiin Microsoft Visual Studio 2013 integroitua sovelluskehitysympäristöä.

Opinnäytetyön tuloksena syntyi ohjelma joka testattiin ja otettiin tuotantokäyttöön kahteen toimipisteeseen. Liiketoimintaprosessien osalta luotiin mallit sekä manuaalisesta kuvausprosessista että automatisoidusta kuvausratkaisusta.

Lisäksi pohdittiin ohjelman jatkokehitysmahdollisuuksia sekä arvioitiin omaa oppimista ja koko opinnäytetyöprosessia.

Asiasanat
C#, liiketoimintaprosessit, automatisointi, passikuva

HAAGA-HELIA
University of Applied Sciences

Abstract

23.11.2014

Author
Martti Vaittinen

Degree programme
Information Technology

Thesis title
Automating ID-picture taking process

Number of pages and appendix pages
22

The purpose of this thesis was to improve the existing process of taking ID-pictures. The primary objective was to create a Microsoft Windows-based program to facilitate ID-picture conversion into the new system by the Finnish National Police Board.

ID-picture taking as a business process was also analyzed. Quality management techniques were discussed.

Properties of Microsoft´s Windows GDI+ library and suitability of C# as a programming language to this project were analyzed. Quality management concepts were presented.

Tool used in the programming section was Microsoft Visual Studio 2013 integrated application development environment.

As a result, working software was created during this thesis process. Software was tested and utilized in two retail locations.

In addition, further development of the program was discussed and the entire thesis process evaluated.

Keywords
C#, business processes, automation, ID-pictures

Sisällys

Termit ja lyhenteet

BPR	Liiketoimintaprosessien uudistamismalli (Business Process Reengineering)
C#	Microsoftin kehittämä ohjelmointikieli joka pyrkii yhdistämään C++ ja Java-ohjelmointikielien parhaat puolet
. NET Framework	Ohjelmistokomponenttikirjasto
CLR	Virtuaalikone joka kääntää ohjelmakoodin suoritettavaan muotoon (Common Language Runtime)
MONO	Alusta- ja käyttöjärjestelmäriippumaton ohjelmistokehitysympäristö .NET-arkkitehtuurille
IDE	Ohjelmointiympäristö (Integrated Development Environment)
TQM	Kokonaisvaltainen laatujohtaminen (Total Quality Management)

1 Johdanto

Huhtikuusta 2013 alkaen on passiin ja henkilötodistukseen menevän kuvan voinut toimittaa poliisille sähköisesti. Poliisin lupahallinnon lupakuvapalvelun tarkoituksena on saattaa kaikki henkilökuvaa vaativat lupatoiminnot sähköisiksi. Myös eri maiden henkilökuvaa vaativat viisumihakemukset ovat yhä enemmän siirtymässä sähköisesti välitettäviksi.

Tavoitteena on tehostaa sekä poliisille menevien kuvien toimitusprosessia että myös muihin tarpeisiin otettavien kuvien käsittelyä. Tehostaminen tarkoittaa prosessin yksinkertaistamista siten, että lyhyelläkin koulutuksella digitaalinen kuvansiirto onnistuu ilman edistynyttä kuvankäsittelyosaamista ja että siihen kuluvaa yksikköaikaa onnistutaan lyhentämään.

Ensisijaisena tavoitteena on luoda Microsoft Windows ympäristössä toimiva työpöytäohjelma passikuvien automaattiseen käsittelyyn poliisin lupahallinnon määräysten mukaisesti. Työssä kartoitetaan myös kuvausta liiketoimintaprosessina.

Työn toimeksiantaja on pääkaupunkiseudulla toimiva valokuvausalan yritys, joka haluaa hyödyntää opinnäytetyön tuloksia kahdessa vähittäismyymälässään.

Työn teoriaosuudessa käsitellään henkilövalokuvausta liiketoimintaprosessina sekä työn toteuttamisen kannalta oleellisia osia .NET arkkitehtuurista ja C# -ohjelmointikielestä.

Työ rakentuu teoriaosuudesta sekä empiirisestä osuudesta jossa ohjelmoidaan Microsoft Windows ympäristössä toimiva työpöytäohjelma passikuvien automaattiseen käsittelyyn.

2 Liiketoimintaprosessit

Tunnistustarkoitukseen tulevan kuvan valmistusprosessi on monotoninen ja toistetaan aina lähes samanlaisena. Kuvaamisen ymmärtäminen liiketoimintaprosessina on tärkeää, koska merkittävin osa valokuvausalan vähittäiskauppojen myyntikatteesta tulee siitä. Liiketoimintaympäristö on vuosikymmenessä muuttunut täysin. Laitekaupan ja filminkehityksien hiipuessa voidaan nyt pitää alaa puhtaasti palveluliiketoimintana. Palveluliiketoiminnassa prosessien hallinta ja laatuajattelu ovat tärkeimmät toimintaa ohjaavat asiat.

Kuvausprosessi itsessään on siirtynyt viimeisten vuosien aikana täysin digitaaliseksi. Tulostus ja jälkikäsittely ovat kuitenkin edelleen manuaalisia ja työteliäitä prosesseja. Fyysisiä kuvia tarvitaan edelleen suurimmassa osassa käyttötarkoituksia.

Valokuva itsessään tulee säilymään biometrisena tunnisteena pitkään. Tulevaisuudessa fyysisiä tulosteita tarvitaan kuitenkin entistä harvemmin.

2.1 Laatu

Laatuajattelulla tarkoitetaan asiakkaan tarpeiden tyydyttämistä mahdollisimman tehokkaasti (Lecklin 2006, 18). Nyt tarkasteltavassa prosessissa korostuu laatukriteereistä virheettömyys. Passikuvan hylkäys aiheuttaa asiakkaalle aina suuremman haitan kuin mitä itse tuotteen arvo (hinta) edustaa. Hylkääminen tarkoittaa merkittävää ajanhukkaa ja ääritapauksessa vaikkapa matkan peruuntumista.

Prosessiajattelussa ensimmäinen vaihe on nykytilanteen kartoitus, jossa arvioidaan prosessin toimintaa ja tehokkuutta. Tätä seuraa analyysivaihe jossa ratkaistaan prosessissa olevia ongelmia ja valitaan käytettävät työkalut ja asetetaan mittarit. Kolmannessa vaiheessa prosessia muutetaan. (Lecklin 2006, 134-135).

Prosessin jatkuva kehittäminen edustaa laatutyötä. Kaikkien organisaatioiden tulee keskittyä tuotteidensa ja palvelujensa laatuun. Tutkimuksissa on todettu että korkea laaduntuottokyky heijastaa hyvää pääoman tuottoa sekä hyvää nettotulosta (Silén 1998, 5).

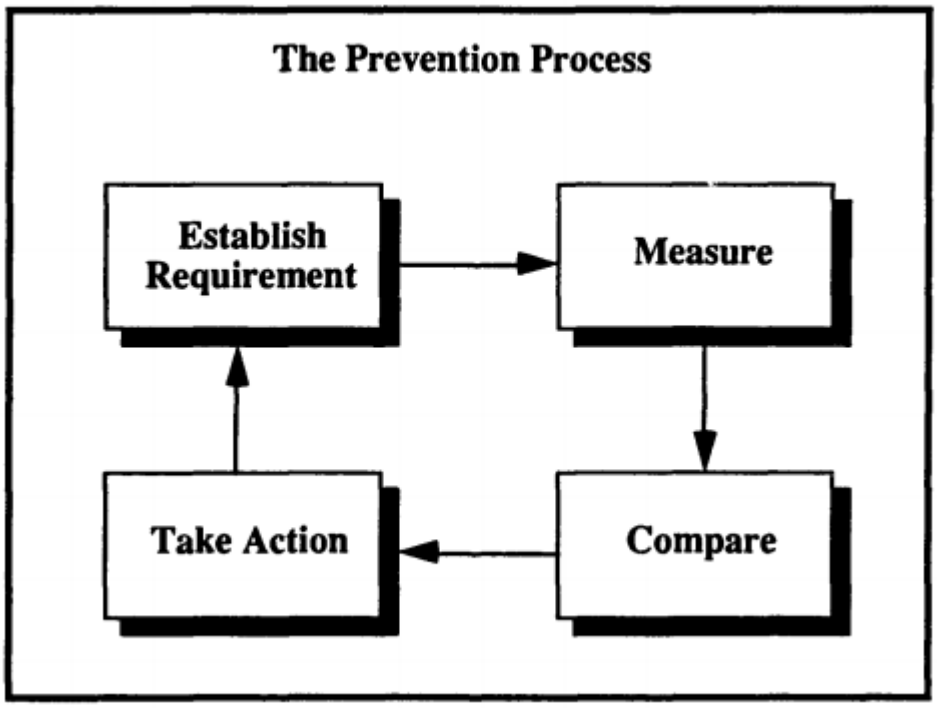
2.2 Laatutavoite

Projektin tavoitteena on nopeuttaa ja yksinkertaistaa passivalokuvauksen prosessia kokonaisvaltaisesti. Tavoiteasetteluun kuuluu parannusten tapahtuminen kaikilla osa-alueilla, niin nopeudessa, laadussa kuin kustannustehokkuudessakin.

Laadun mittaaminen tässä prosessissa on yksinkertaista. Virallinen mittari on kuvan hyväksyminen viranomaisten toimesta käyttötarkoitukseensa. Koska käytössä olevan manuaalisen prosessin virhetaso oli erittäin pieni, alle 1 hylkäys tuhatta kuvausta kohti, voitiin uudelle automatisoidulle prosessille asettaa tavoitteeksi vain virheiden nollataso.

Kokonaisvaltaisessa laatujohtamisessa (TQM) virheiden nollatoleranssin idean esitti yhdysvaltalainen Philip B. Crosby (Silén 1998, 43). Crosbyn mukaan (Suárez 1992, 4) laatua joko on tai ei ole. Mitään laadun väliportaita ei ole olemassa ja vastuu laadusta on ensisijaisesti organisaation ylimmällä johdolla.

Virheiden estämiseksi tulee ensin luoda vaatimusmääritys parannettavalle tuotteelle tai palvelulle, sitten kehittää sitä mitattavasti, vertailla kehittämisen tuloksia vaatimuksiin ja lopuksi suorittaa tarvittavat toimenpiteet. Tämä Crosbyn jatkuva virheidenestoprosessi on esitetty kaaviossa 1 (Suárez 1992, 6).



Kaavio 1. Virheiden välttäminen Crosbyn mukaan

2.3 Manuaalinen kuvausprosessi

Kaaviossa 2. on esitetty tunnistekuvauksen prosessi perinteisellä tavalla jossa lopputuloksena on fyysinen kuva ja tarvittaessa lähetyskelpoinen kuvatiedosto. Tämä prosessi perustuu toimeksiantajatahon toimintamalliin. Merkille pantavaa on työvaiheiden suuri määrä sekä prosessin edellyttämä kuvankäsittelytaito.

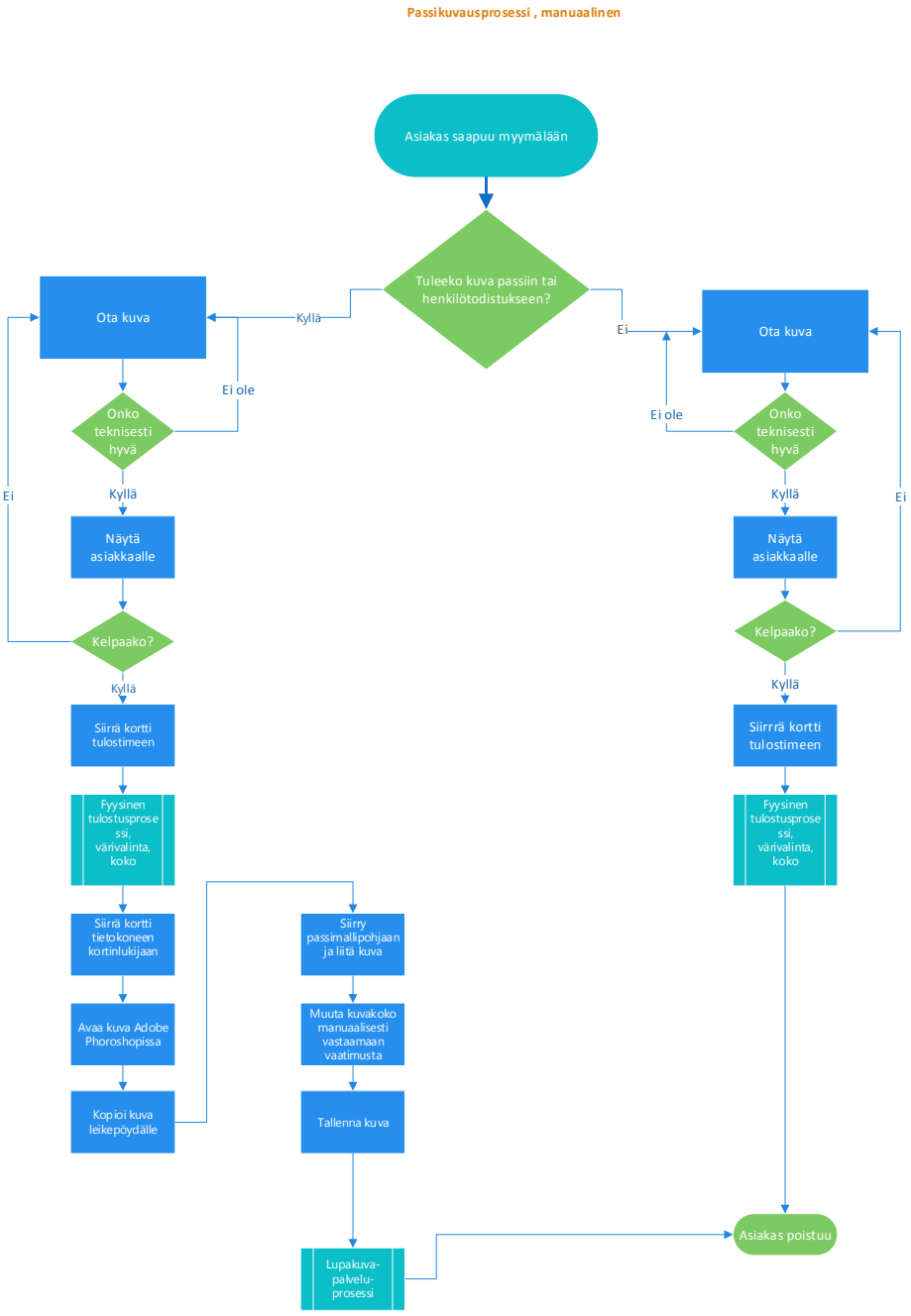
Liiketoimintaprosessiteorian kannalta suurin hyöty tässä tapauksessa saavutetaan vähentämällä ja yksinkertaistamalla toimenpiteiden määrää. Tilanne on hyvin samankaltainen kuin tavaroiden jakelussa eli logistiikkaketjussa. Logistiikassa korostuu kuitenkin läpimenoajan lyhentäminen BPR (Business Process Reengineering) termin mukaan.

BPR:n tavoitteita ovat:

- prosessien yksinkertaistaminen
- tuhlauksen poistaminen
- läpimenoaikojen lyhentäminen
- turhan työn eliminointi
- kustannusten pienentäminen
- automatisointi
- lisäarvon tuottaminen asiakkaalle

(Lähde: Logistiikan maailma, verkkosivusto)

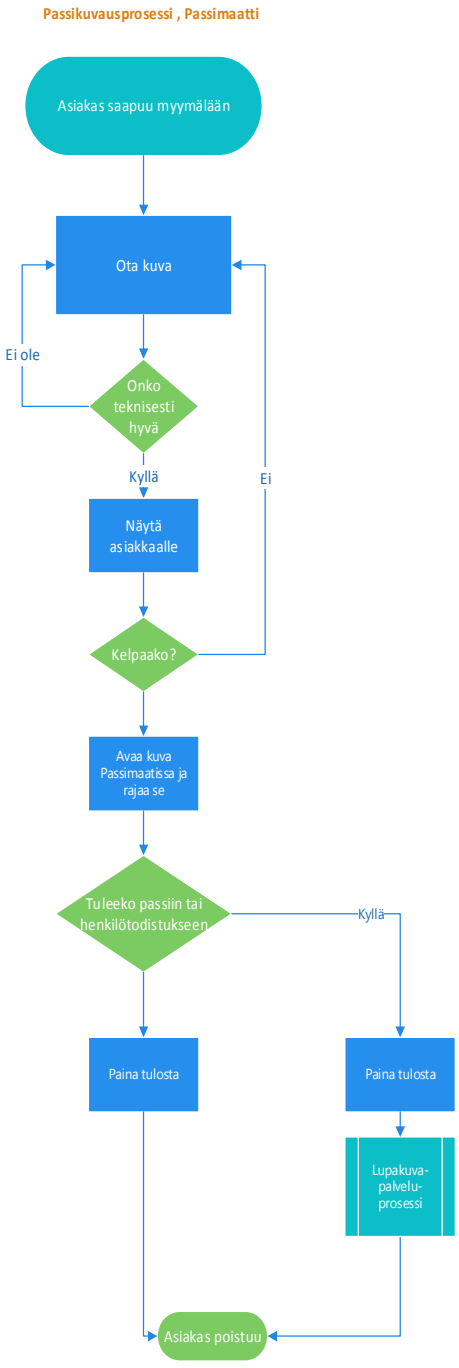
Tämän opinnäytetyön perustana liiketoimintaprosessin kehittämisen suhteen toimivat edellä käsitellyt laatu- ja tehokkuusmenetelmät.



Kaavio 2. Manuaalinen kuvausprosessi

2.4 Automatisoitu kuvausprosessi

Tämän opinnäytetyön kuluessa on muotoutunut kaavion 3 mukainen menettelytapa. Selkein ero vanhaan prosessin on prosessivaiheiden määrän pieneneminen. Myös yksittäisten toimintojen kohdalla toimenpiteiden määrä on pienentynyt.



Kaavio 3. Automatisoitu kuvausprosessi

2.5 Mitattavat hyödyt

Suurin hyöty uudesta järjestelmästä on prosessin läpivientiajan dramaattinen lyhentyminen. Tämä merkitsee mahdollisuutta palvella enemmän asiakkaita, asiakastyytyväisyyden kasvua ja lyhyempää perehdyttämisjaksoa uusille työntekijöille.

3 Kehitysympäristö ja työkalut

Kurssiohjelmaani kuului ohjelmointia sekä Javalla että C#:lla. Ohjelmointikieleksi valikoitui C# sen tarjoamien edistyneiden kuvankäsittelymahdollisuuksien vuoksi. Kehitystyö tapahtui Microsoft Visual Studio 2013 jossa on paras integrointi ohjelmointikieleen.

Vaihtoehtona ollut Mono Studio kehitysympäristö ei tuntunut aivan valmiilta eikä Visual Studiolla tehty ratkaisu toiminut siinä ollenkaan.

3.1 C#

Kehittäjänsä Microsoftin mukaan C# yhdistää C – kielen tehokkuuden ja Javan helppouden.

C# ohjelmointikieli sopi tehtävään mainiosti koska lähdekoodia oli runsaasti ja graafiset kirjastot kehittyneitä. Lisäksi oppimisen ja tulevaisuuden kannalta C# on hyvä valinta koska yritysmaailmassa sitä käytetään laajalti.

C# on vahvasti tyypitetty oliokieli. Vahva tyypitys edellyttää muuttujien olevan tyyppinsä mukaisia mikä edesauttaa virheiden havaitsemista. Olioperusta puolestaan jakaa ohjelman objekteihin ja metodeihin, mikä auttaa laajan kokonaisuuden hahmottamista.

C# on alustariippumaton mutta kirjoitettu toimimaan erityisesti .NET viitekehyksessä (Albahari & Albahari, 2010).

3.2 .NET Framework

Microsoftin kehittämä .NET Framework koostuu kahdesta osasta, luokkakirjastoista ja ajonaikaisesta ympäristöstä (CLR). Se tukee lukuisia eri ohjelmointikieliä joista suosituimmat ovat C# ja Visual Basic .NET. CLR tukee kaikkia ohjelmointikieliä joita voidaan kääntää Common Intermediate Language –koodiksi (CIL). CIL ei ole laitteisto- tai käyttöjärjestelmäkohtainen. Näin samaa suoritettavaa koodia voidaan ajaa eri ympäristöissä ilman muutoksia tai vain vähäisin muutoksin.

3.3 Mono

Mono on vapaan lähdekoodin ohjelmistoprojekti .NET yhteensopivan kehitysympäristön luomiseksi muillekin kuin Microsoft Windows käyttöjärjestelmille. Ensisijaisena tavoitteena on ajaa Microsoft .NET sovelluksia eri alustoilla kuten Android- ja Linux-käyttöjärjestelmissä. (http://en.wikipedia.org/wiki/Mono_(software)).

3.4 GDI+

Graphics Device Interface+ (GDI+) on Windows käyttöjärjestelmien komponentti joka koostuu ohjelmistorajapinnasta (API) ja käyttöjärjestelmän ydinosasta tulostamaan kuvia, grafiikkaa ja tekstiä sekä näytölle että tulostimelle. GDI+ korvaa aiemman Windows GDI kirjaston tarjoten monia tämän projektin edellyttämiä ominaisuuksia (http://www.techopedia.com/definition/24288/graphics-device-interface–gdi).

GDI+ toimii ohjelmien ja laitteiston välissä kuvantaen 2-ulotteisia kuvia, grafiikkaa ja tekstiä. Se tarjoaa täten Windows sovelluksille ominaisen WYSIWYG kokemuksen.

GDI+ kirjaston toiminnallisuus jakautuu viiteen osaan:

- kuvantaminen
- tulostaminen
- kaksiulotteinen vektorigrafiikka
- typografia
- suunnittelu

Tälle projektille tärkeitä asioita olivat kuvantaminen, tulostaminen ja vektorigrafiikka

3.4.1 GDI+ projektissa

Läpinäkyvyys eli alpha – kanava on GDI+ kirjastossa mukana värirakenteessa, mikä on tälle ohjelmalle tärkeä ominaisuus. Sen avulla visuaalinen hahmotus oli helppo toteuttaa. Kasvojen maskaaminen läpinäkyvän kerroksen ja apuviivojen avulla oli intuitiivinen tapa rajata kuva nopeasti ja tarkasti.

Tekstuuri- ja liukupensselit ovat myös tulleet GDI+ kirjastoon. Kirjasto tulee nyt myös suoraan eri tiedostomuotoja kuten .jpeg, .bmp, .tiff, .gif ja .png. Samoin sisäänrakennettuna ovat kuvien muokkauksen rutiinit kuten rajaus ja suurennossuhteen muutos. Muunnos rasteri- ja vektorimuotojen välillä käy myös helposti. Näitä kaikkia ominaisuuksia tarvittiin ohjelmassa.

sRGB-tuki on mukana värienhallinnassa. Kameralle otetut kuvat ovat natiivisti sRGB-muodossa, joten ohjelman suorittamissa konversioissa värimaailma säilyi läpi koko prosessin. Muita uusia tuettuja väriavaruuksia ovat mm. sRGB64 ja ICM.

3.4.2 Vektorigrafiikka

Vektorigrafiikalla voidaan piirtää yksinkertaisia muotoja, primitiivejä, tietyssä koordinaattisysteemissä. Näitä muotoja ovat mm. suorat, monikulmiot, ympyrät ja kaaret. Toisin kuin bittikarttakuvia vektorigrafiikalla tuotettuja kuvia voidaan skaalata rajattomasti laadun siitä kärsimättä.

3.4.3 Kuvantaminen

Image luokka tarjoaa metodit rasterikuvien (bittikartta) lataamiseen ja tallentamiseen. Image-tyyppinen objekti kapseloi bittikartan tai metatiedoston (vektorikuva) ja sen attribuutit. Image tyyppisiä objekteja voi luoda erilaisista tiedostotyypeistä kuten BMP, ICON, GIF, JPEG, Exif, PNG, TIFF, WMF ja EMF.(http://msdn.microsoft.com/en-us/library/windows/desktop/ms534462(v=vs.85).aspx)

3.4.4 Typografia

GDI+ tarjoaa luokat fonttien käyttöön. Luokkia ovat mm. Font ja FontFamily. GDI+:n avulla voidaan käyttää kaikkia järjestelmään asennettuja fontteja.

3.4.5 Tulostaminen

GDI+:n avulla on helppo käyttää Windows-käyttöjärjestelmän tulostustoiminnallisuutta. Näiden luokkien avulla hallitaan järjestelmään asennettujen tulostinten järjestystä, paperikokoja, tulostusresoluutiota ja muita toimintoja.

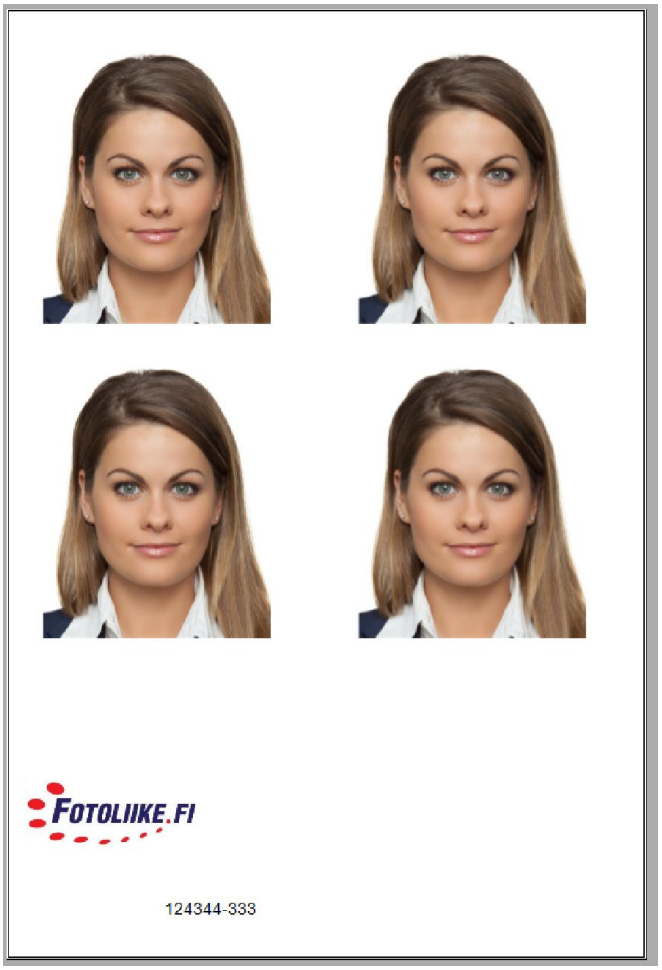
Oheisessa ohjelman koodiesimerkissä (listaus 1) muodostetaan tulostettava arkki 100x150mm neljälle passikuvalle, logolle ja sarjanumerolle:

```
private void myPrintDocument1_PrintPage(object sender, PrintPageEventArgs e)
        {
            //printteri 334dpi , pointit vs pikselit
            int xSiirtyma = 205;
            int ySiirtyma = 205;
            int koko = 650;
```

```
            Image logo = Image.FromFile(@"C:\\passik\logo.jpg");
            Image img= Image.FromFile(@"C:\\passik\print.jpg");

            img = FixedSize(img, koko, koko);
          //  img.RotateFlip(RotateFlipType.Rotate180FlipNone);


            int repeatTimes = 1; // yksi arkki , voi parametroida
            using (Graphics graphics = Graphics.FromImage(img))
            {
                Point newLocation = new Point(0, 10);
                Point logoLocation = new Point(10, 500);

                Point sarjaLocation = new Point(100, 580);


                for (int imageIndex = 0; imageIndex < repeatTimes; imageIndex++)
                {
                    e.Graphics.DrawImage(img, newLocation);
                    newLocation = new Point(newLocation.X + xSiirtyma, newLocation.Y);
                    e.Graphics.DrawImage(img, newLocation);
                    newLocation = new Point(newLocation.X-xSiirtyma, newLocation.Y+ySiirtyma);
                    e.Graphics.DrawImage(img, newLocation);
                    newLocation = new Point(newLocation.X+xSiirtyma , newLocation.Y);
                    e.Graphics.DrawImage(img, newLocation);
                    e.Graphics.DrawImage(logo, logoLocation);
                    e.Graphics.DrawString(Resources.Resource1.numero, new Font("Arial", 8), drawBru
sh, sarjaLocation);
                }

            }
```

Listaus 1. Tulostusrutiini

Printtausrutiinin tuloksena syntyvä kuva on esitetty kuvassa 4.

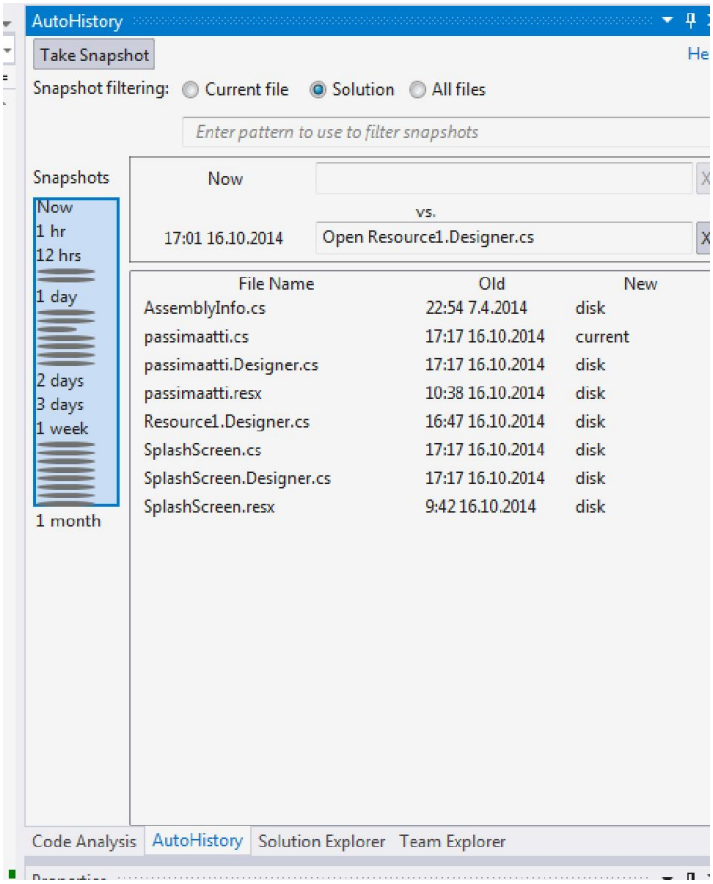


Kuva 1, tulostusarkki 4 kpl 40mmx50mm

3.5 Sovelluskehitysympäristö

C# ohjelmointia voi tehdä lähes kaikilla alustoilla, niin Windowsissa kuin muissakin käyttöjärjestelmissä. Koska kaikki perustyökalut on kehittänyt pisimmälle Microsoft, oli luonnollista valita Visual Studio IDE toteutukseen. Se sisältää visuaalisen suunnitteluympäristön, debuggerin virheiden löytämiseen, projektinhallintatyökalut ja julkaisutyökalut.

Visual Studioon voi liittää lisäosia tarpeen mukaan. Omassa ohjelmointiympäristössäni hyödyllinen lisäosa oli AutoHistory, jonka avulla voi palata aiempiin tiedostoversioihin helposti.



Kuva 2, AutoHistory lisäosa Visual Studio IDE

Visual Studio tarjoaa myös hyvät työkalut ryhmätyöskentelyyn.

4 Passimaatti-ohjelma

Ohjelman kehitystyössä käytettiin ketteriä menetelmiä. Oleellista tässä oli koko projektin ajan saada palautetta ohjelman eri versioiden toiminnasta käyttäjiltä. Ohjelma toteutettiin ns. suihkulähdemallin mukaan. Tässä mallissa ohjelmointi suoritetaan ensin ja dokumentointi vasta jälkeenpäin. Malli toki soveltuu vain pienten ohjelmien tekoon. Koska kyseessä oli yksittäinen ohjelma, jonka toiminnallisuus ei ollut kriittisen tärkeä liiketoiminnalle, sopi käytetty metodi ohjelmiston kehittämiseen hyvin.

Koska myös vanha järjestelmä oli koko ajan käytettävissä, voitiin kehitysversioita testata suoraan tuotannossa.

4.1 Vaatimusmäärittely

Vaatimusmäärittely luotavalle järjestelmälle oli varsin suoraviivainen prosessi. Ongelmat nykytilanteessa tunnistettiin ja koska osallisia tahoja oli vähän, voitiin määrittely suorittaa nopeasti.

4.1.1 Prosessi

Määrittelyn lähtökohtana oli jo tuotannossa olevan prosessin yksinkertaistaminen. Koko prossin määrityksessä todettiin järjestelmän tarvitsevan seuraavat komponentit:

- Kuvansiirtoratkaisu kamerasta suoraan taustajärjestelmään
- Ohjelma automatisoituun siirtotiedoston tuottoon
- Tulostusratkaisu joka kattaa kaikki tarvittavat fyysiset kuvakoot

4.1.2 Ohjelma

Poliisihallituksen ohjeissa määritellään digitaalisen kuvan formaatti ja toimitustapa (https://www.lupakuvienvastaanotto.fi/FaceImageHelp/Index.aspx). Kuva perustuu ICAO:n (kansainvälinen siviili-ilmailuliikennejärjestö) ohjeisiin.

Siirtotiedoston tietotekniset vaatimukset ovat ohjelman vaatimusmäärityksen pohja:

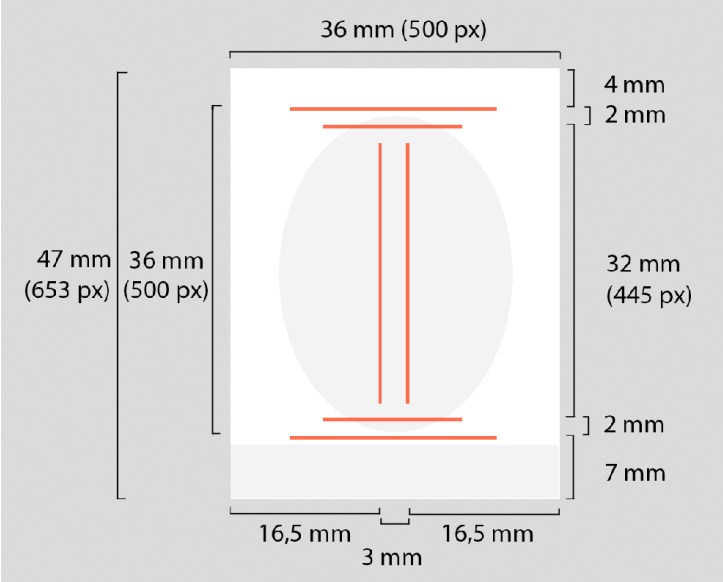
- Hakemuksen kuva saa olla enintään kuusi (6) kuukautta vanha.
- Valokuva voi olla mustavalkoinen tai värillinen.

- Kuvan mittojen on oltava tarkalleen 500 x 653 pikseliä. Yhdenkään pikselin poikkeamia ei hyväksytä.
- Kuva on tallennettava JPEG-muodossa (ei JPEG2000), tiedostopääte voi olla .jpg tai .jpeg.
- Kuvassa ei saa esiintyä liiallisesta pakkauksesta johtuvia JPEG-artefakteja (pakkaushäiriöitä).
- Kuvan suurin sallittu tallennuskoko on 250 kilotavua.
- Henkilön ulkonäköä muuttava kuvan manipulointi ja retusointi on kielletty.
- Digitaalinen ehostaminen ei ole sallittua.

(https://www.lupakuvienvastaanotto.fi/FaceImageHelp/HelpFiles/Kuvaohje.pdf).

Näin ollen ohjelman osalta vaatimusmääritys oli helppoa koska ohjeet ovat hyvin selkeät. Lupahallinnon kuvapalvelu suorittaa valokuvalle syöttövaiheessa koneellisen tarkistuksen ja mikäli em. vaatimukset eivät täyty, ei järjestelmä ota kuvaa vastaan. Tämä tarkistus koskee siirtotiedoston teknistä muotoa ja hahmontunnistukseen pohjautuvaa automaattista kasvojen arviointia. Lopullisen tarkistuksen tekee kuitenkin aina passivirkailija hakemusta jätettäessä.

Myös kasvojen asemoinnista on tarkat ohjeet. Kasvojen tulee olla tiedostossa kuvan 3 mukaisesti. Koska käytännössä ei kuvatessa ole mahdollista saada kasvoja täsmälleen oikean paikkaan, tuli ohjelmassa olla helppo tapa kohdistaa kasvot oikein. Samoin tuli suurennussuhteen muutoksen olla helppoa.



Kuva 3. Kasvojen asemointi

4.2 Suunnittelu

Suunnittelussa lähdettiin yksinkertaisuuden periaatteesta. Työvaiheita tuli olla mahdollisimman vähän. Suunnittelu- ja käyttöpalavereissa todettiin yksimielisesti että erityistä huomiota tuli kiinnittää asiakastyytyväisyyteen. Asiakkaan myyntitapahtuman kokemuksen tuli olla entistäkin parempi. Konkreettisiksi suunnittelutavoitteiksi asetettiin

- kuvaustapahtuman nopeus
- kiireettömyyden ilmapiiri
- otosten määrä
- esikatselu ja hyväksyntä
- digitaalisuuden mukanaan tuoma WOW – efekti
- koko prosessin nopeus

4.2.1 Laiteympäristö

Siirtoyhteys kamerasta tietokoneeseen voidaan toteuttaa monilla eri tavoilla niin langallisesti kuin langattomastikin.

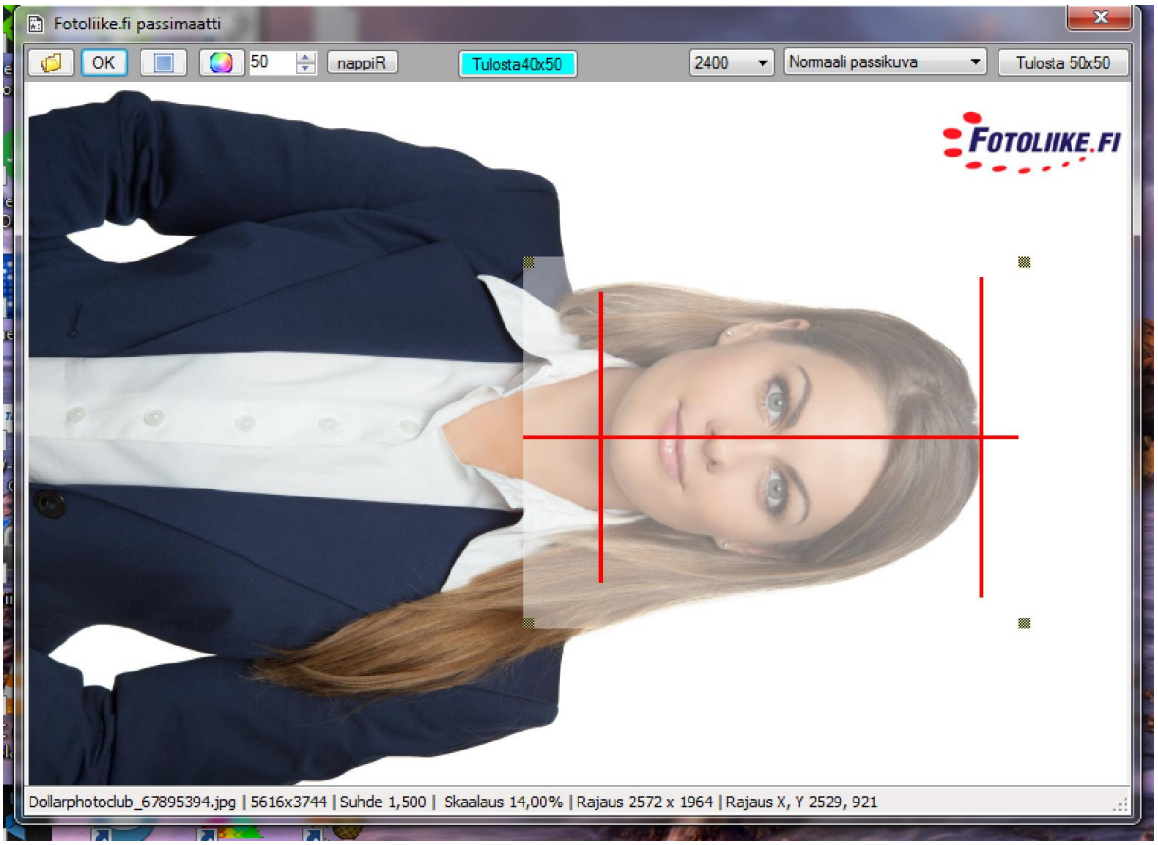
Koska langalliset siirtoratkaisut kamerasta taustajärjestelmään olivat kömpelöitä ja vikaantumisalttiita päädyttiin kuvansiirrossa käyttämään langatonta ratkaisua. Markkinoilla on muutamien valmistajien kamerakohtaisia ratkaisuja sekä yleiskäyttöisiä Wi-Fi muistikorttiin perustuvia ratkaisuja. (http://www.cnet.com/how-to/how-to-add-wi-fi-to-your-dslr-or-mirrorless-interchangeable-lens-camera/.)

Järjestelmässä päädyttiin käyttämään SDHC-formaatin mukaista EyeFi Mobi SDHC Wi-Fi-muistikorttia. Sen avulla kuva siirtyy heti kuvauksen jälkeen taustatietokoneelle haluttuun kansioon.

Koska jo aiemmin käytössä ollut kuvauskamera oli yhteensopiva valitun langattoman verkkokortin kanssa, ei kuvausrutiiniin tarvinnut puuttua.

4.2.2 Ohjelma

Sovellus ”Passimaatti” on tyypillinen Windows Forms -ohjelma. Se toimii työasemassa työpöydällä. Ohjelman tuli samalla tuottaa poliisihallituksen lupakuvajärjestelmän mukainen lähetettävä siirtotiedosto sekä tulostettavan kuvan edellyttämä tiedosto. Kuvassa 4 on esitetty ohjelman käyttöliittymä.



Kuva 4. Käyttöliittymä

Ohjelman käyttö on hyvin pelkistettyä. Ohjelmassa suoritettavat työvaiheet ovat :

1. kuvan avaus
2. kuvan rajaus haluttuun formaattiin
3. kohdistus
4. talletus
5. tulostus

Työvaiheiden jälkeen tuloksena on fyysinen tuloste ja poliisihallituksen ohjeiden mukainen lähetysvalmis tiedosto.

4.2.3 Tulostusratkaisu

Poliisihallinnon lupakuvaohjeissa on määritelty fyysisen kuvan spesifikaatiot. Kuva tulee käytännössä tulostaa joko valokuvaprosessilla tai laminoivalla sublimaatiotulostimella arkistointikelpoisuuden takaamiseksi.

Ensimmäiseksi tulostimeksi valittiin jo olemassa oleva Noritsu IP-64 sublimaatiotulostin joka pystyy tulostamaan 10x15 kuva-arkin noin kymmenessä sekunnissa. Aiemmassa manuaalisessa järjestelmässä tulostus kesti yli minuutin minkä vuoksi tulostimia oli kaksi kappaletta. Tämä työvaihe oli ruuhka-aikoina monesti prosessin pullonkaula. Tulostuksen ohjelmoinnissa käytetyn GDI+ ohjelmakirjaston tulisi tarjota identtinen tulostuskoko ja laatu kaikille Windowsissa tuetuille tulostimille.

4.3 Testaus

Vanha kuvausjärjestelmä toimi koko ajan uuden rinnalla joten testit voitiin suorittaa tuotantoympäristössä. Testausta suoritettiin kahdessa myymälässä heterogeenillä laitteilla. Myymälässä 1 työasemana oli Intel i3 prosessorilla ja 4 gigatavun keskusmuistilla varustettu Windows 7 työasema. Näyttö oli resoluutioltaan 1920x1080.

Tulostin Noritsu IP-64 oli kytkettynä sekä suoraan työasemaan että myös tulostuspalvelimeen jolloin sitä käytettiin verkon kautta. Verkotuksella ei ollut merkitystä suorituskykyyn, tulostusajassa ei ollut eroja.

Myymälässä 2 työasemana toimi Intel Celeron-tasoisella prosessorilla ja 4 gigatavun keskusmuistilla varustettu 17-tuumainen Windows 7 kannettava. Näyttö oli resoluutioltaan 1600x900.
Tulostimena toimi jo olemassa oleva sublimaatioprintteri.

Molemmissa testiympäristöissä havaittiin sama GDI virhetilanne joka toistui satunnaisesti.

Kuva 5. GDI+ kirjaston virheilmoitus

Muistinkäyttöä ja koodia analysoidessa havaittiin virheen olevan muistin käsittelyssä. .NET- arkkitehtuurissa muistin roskanpoiston ”Garbage collection” tulisi tapahtua automaattisesti. Näin havaittiinkin tapahtuvan mutta viiveellä. Ohjelman objektit eivät vapautuneet muistista koodin mukaan. Ohjelman käyttämä muistimäärä kuvan latauduttua vaihteli 50 Mt ja 400Mt välillä. Pelkästään odottamalla muutamia kymmeniä sekunteja putosi muistin käyttö alhaisemmaksi.

Ohessa koodilistaus 2 jonka tulisi poistaa bittikartat ja oliot muistista kun kuvatiedostot on tallennettu.

```
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                try
                {
                  //Bitmap iso2 = null;
                    bmp = (Bitmap)CropImage(pictureBox1.Image, ScaledCropRect);
                   iso2 = (Bitmap)CropImage(pictureBox1.Image, IsoScaledCropRect);


System.Drawing.Image bmp2 = FixedSize(bmp, 653, 500);
System.Drawing.Image iso3 = FixedSize(bmp, 800, 600);


                    bmp2.RotateFlip(RotateFlipType.Rotate270FlipNone);
                    iso2.RotateFlip(RotateFlipType.Rotate270FlipNone);
                    bp2 = (Bitmap)bmp2;
                    iso = (Bitmap)iso2;
                    iso.SetResolution(334.0F, 334.0F);
                    bp2.SetResolution(300.0F, 300.0F);

                    saveJpeg(saveFileDialog1.FileName, bp2, 85);
                    saveJpeg(@"c:\passik\print.jpg", iso, 85);
                    bp2.Dispose();
                    iso.Dispose();

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, "Ok painettu, virhe");
                }
            }

            if (bmp != null)
                bmp.Dispose();
```

Listaus 2. Muistin vapauttaminen

Tähän ongelmaan löytyi myös viitteitä kirjallisuudesta

- (http://stackoverflow.com/questions/17130382/understanding-garbage-collection-in-net).

- http://blogs.msdn.com/b/tess/archive/2009/02/03/net-memory-leak-to-dispose-or-not-to-dispose-that-s-the-1-gb-question.aspx

Ongelma poistui käytettäessä pienempää resoluutiota kuvatessa.

Käyttökokemus molemmissa testausympäristöissä oli yhtenevä. Käyttäjien mielestä suurin ero syntyi näytön resoluutiosta. Suuremman resoluution näytössä käyttökokemus oli jouhevampi koska siirtyminen ikkunoiden välillä oli helpompaa.

4.4 Käyttöönotto ja ylläpito

Käyttöönotto sujui ensimmäisessä myymälässä odotettua paremmin. Uusi järjestelmä otettiin vanhan rinnalle käyttökokemuksien kartuttamiseksi. Käyttäjiä oli neljä ja ohjelman omaksuminen helppoa. Siirtymäkautta helpotti lupakuvapalvelusta tuleva palaute. Sen avulla verkkopalvelussa nähdään lähetettyjen kuvien laatuarviointi.

Laatuarviointi on kaksivaiheinen. Kuvaa lähetettäessä suorittaa järjestelmä kuvatiedostolle automaattisen laatuarvioinnin. Automaattinen laadunarviointi arvioi kuvan tummuusastetta, valaisun tasaisuutta sekä kuvan terävyyttä. Valokuvan laadun arvioi lopullisesti virkailija kun itse hakemus tulee käsiteltäväksi.

Siirtymäaika uuteen systeemiin oli yksi päivä. Käyttöönottoa seuraavasta päivästä alkaen kaikki kuvat tehtiin uudella järjestelmällä.

Toisessa myymälässä käyttöönottokausi oli pidempi, useita viikkoja, jona aikana uutta ja vanhaa järjestelmää käytettiin rinnakkain. Ongelmia aiheutui myymälän sisäisestä langattomasta verkosta, jonka toiminta oli takkuista. Teknisten ongelmien ratkettua toimi järjestelmä hyvin.

Ylläpitoa varten toimeksiantajalle toimitetaan ohjelman koko dokumentointi. Ylläpitosuunnitelma laaditaan jos ohjelmalle löytyy riittävää kaupallista kysyntää.

5 Jatkokehitys

Tällä hetkellä ohjelma tukee kaikkia suomalaisia kuvaformaatteja sekä useimpien maiden viisumi- ja lupakuvamuotoja. Kattavuus on nyt yli 99% kaikista tulosteista. Erikoisemmat tulosteet tehdään edelleen vanhalla järjestelmällä.

Harvemmin tarvittavia erikoiskokoja tulee lisätä paperitulostusformaatteihin. Lisäksi tulee lisätä digitaalisten viisumikuvien tiedostoformaatteja.

Kaupallista hyödyntämistä tutkitaan tekemällä ohjelmasta markkina-analyysi valokuvausalan toimijoiden keskuudessa. Ohjelman laajempi jakelu edellyttää selkeää ohjeistusta niin laiteympäristön, asennuksen kuin käytönkin osalta. Tämänkaltaista panostusta ei kovin rajatulle asiakaskunnalle kannata tehdä.

6 Yhteenveto

Projektin tuloksena syntyi Microsoft Windows käyttöjärjestelmässä toimiva työasemaohjelma, jolla henkilöstä otettu kasvokuva saadaan muutamalla työvaiheella lähetysvalmiiksi tiedostoksi sekä halutun kaltaiseksi paperikuvaksi.

Koska vaaraa liiketoiminnalle ei ollut, vastoin kaikkia ohjelmistokehityksen perusperiaatteita, oli erityisen hyödyllistä käyttää ohjelman eri kehitysversioita suoraan tuotannossa. Ensimmäinen prototyyppi syntyi muutamassa viikossa. Vaikkakin sen toiminnallisuus rajoittui tuottamaan vain oikean kokoisen kuvatiedoston, oli tämä oleellista projektin lopputulosta silmällä pitäen.

Rajanveto ketterän ja ”Cowboy”-tyyppisen koodauksen välillä on pieni. Halu saavuttaa tietty toiminnallisuus nopeasti ja vaivattomasti menee helposti dokumentoinnin ja kunnollisen kommentoinnin ohi. Tätä opinnäytetyöprosessia voi pitää erikoistapauksena joka ei sovi kuin harvoihin tapauksiin. Tässä tapauksessa tekijä hallitsi koko tehtäväkentän, niin liiketoimintaprosessin kuin tietoteknisenkin osan. Ongelman ratkaisuun oli myös looginen polku ja näköpiirissä selkeä ratkaisu.

Ohjelmisto ja uusi liiketoimintaprosessi on otettu käyttöön ongelmitta kahdessa vähittäismyymälässä. Käyttöönoton helppous ja käyttäjien lyhyessä ajassa saavuttama nopea systeemin hallinta oli yllättävää.

Oppimisen kannalta tämä työ on ollut palkitseva. Ohjelmistokehityksen aikataulu piti hyvin paikkansa, sen sijaan dokumentointiin kului odotettua enemmän aikaa.

Tämä opinnäytetyö pakotti tekijänsä systemaattisesti miettimään myös työn eri vaiheita ja kuvausta prosessina. Tämä puolestaan on antanut ideoita jatkokehitykseen niin passikuvausprosessin edelleen jalostamiseen kuin kokonaan uusien liiketoimintamahdollisuuksien luomiseen.

Lähteet

.NET Garbage collection, Luettavissa http://stackoverflow.com/questions/17130382/understanding-garbage-collection-in-net. Luettu 21.10.2014

Albahari, J., Albahari, B. 2011. C# in a nutshell. Fourth edition. O'Reilly Media Inc. Sebastopol.

Carey, M. Taming the Wild West: the Differences between Agile and Cowboy Coding. Blogi. Luettavissa: http://careytechblog.blogspot.fi/2013/08/taming-wild-west-differences-between.html. Luettu: 10.8.2014.

GDI+ Interface. Luettavissa http://www.techopedia.com/definition/24288/graphics-device-interface--gdi. Luettu 28.8.2014.

Image class. Luettavissa http://msdn.microsoft.com/en-us/library/windows/desktop/ms534462(v=vs.85).aspx. Luettu 29.10.2014.

Logistiikan maailma. Verkkosivusto. Luettavissa http://www.logistiikanmaailma.fi/wiki/Toimitusketjun_kehitt%C3%A4minen. Luettu 22.10.2014.

Lecklin, O. 2006. Laatu yrityksen menestystekijänä. 5. painos. Talentum. Helsinki.

Mehra, P. What's new in GDI+ for GDI Programmers?. Luettavissa http://www.c-sharpcorner.com/Blogs/1620/what%E2%80%99s-new-in-gdi-for-gdi-programmers.aspx. Luettu 2.9.2014.

Mono. Luettavissa http://en.wikipedia.org/wiki/Mono_(software). Luettu 12.8.2014.

Poliisihallitus. Lupakuvapalvelu. Luettavissa https://www.lupakuvienvastaan-otto.fi/FaceImageHelp/Index.aspx. Luettu: 21.3.2014.

Silén, T. 1998. Laatujohtaminen. WSOY. Porvoo.

Suarez J. 1992. Three Experts on Quality Management. Luettavissa http://dtic.mil/dtic/tr/fulltext/u2/a256399.pdf. US Department of the Navy. Arlington

WiFi. Luettavissa http://www.cnet.com/how-to/how-to-add-wi-fi-to-your-dslr-or-mirrorless-interchangeable-lens-camera/. Luettu 12.10.2014.