Haaga-Helia
ammattikorkeakoulu Oy

Vaatimusmäärittelyn merkitys tietojärjestelmän kehittämisessä

Päivi Lehtimäki

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2017

Haaga-Helia
University of Applied Sciences Ltd.

Tiivistelmä

9.2.2017

Tekijä(t)
Päivi Lehtimäki

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Vaatimusmäärittelyn merkitys tietojärjestelmän kehittämisessä

Sivu- ja liitesivumäärä
51+21

Opinnäytetyön otsikko englanniksi
Requirements engineering's meaning for information system's development

Opinnäytetyö käsittelee vaatimusmäärittelyn merkitystä tietojärjestelmän kehittämisessä. Työssä tarkastellaan Yritys X:n ylläpitovaiheessa olevan tietojärjestelmän pienkehitystyöprojektia, joka toteutettiin vuonna 2014. Opinnäytetyön kirjoittaja oli kehitysprojektissa yhtenä projektiryhmän jäsenenä määrittelemässä asiakasvaatimuksia, sekä testaamassa määriteltyjä vaatimuksia järjestelmän pääkäyttäjän näkökulmasta. Opinnäytetyö toimii opinnäytetyön kirjoittajalle oman oppimisen prosessina.

Työn tavoitteena oli tutkia vaatimusmäärittelyn prosessia, kuinka se tulisi toteuttaa ja mitä tekniikoita vaatimusten dokumentoimisessa tulisi käyttää. Teoriatietojen perusteella analysoitiin Yritys X:n pienkehitystyöprojektin vaatimusmäärittelyprosessia, sekä voisiko kohdeorganisaatio kehittää vaatimusmäärittelyään jollain tavalla.

Käytännössä vaatimusten dokumentoiminen havaittiin tässä kehitysprojektissa puutteelliseksi. Asiakasvaatimuksiin perustuvaa toiminnallisten vaatimusten määrittelydokumenttia ei luotu, eikä vaatimuksia dokumentoitu yleisten alalla käytettävien tekniikoiden, kuten esimerkiksi UML-kuvausten avulla. Lisäksi projektiin liittyvä dokumentaatio oli osittain vain sovelluksen kehittäjien hallussa. Jatkossa vaatimusten dokumentoimiseen tulisi panostaa ja kaikki projektin dokumentaatio tulisi olla myös tietohallinnon ja liiketoiminnan saatavilla, eikä vain kehittäjien, jotta niitä voitaisiin hyödyntää tulevissa projekteissa.

Asiasanat
Vaatimusmäärittely, ohjelmistokehitys, tietojärjestelmä, tietohallinto

Sisällys

1 Johdanto

Useat tutkimukset osoittavat, että laadukkaasti toteutettu vaatimusmäärittely ja vaatimustenhallinta ovat ohjelmistokehityksen yksi peruspilari, jotka toimivat tärkeänä yhdistävänä tekijänä ohjelmistotuotantoon kuuluvien muiden aktiviteettien välillä, ja vaikuttavat merkittävästi myös siihen, että kehitysprojektin aikataulu ja budjetti eivät ylity. (Haikala & Mikkonen 2011, 61; Haikala & Märijärvi 2006, 94; Kotonya & Sommerville 2004, 9; Pohl & Rupp 2011, 2.)

Kuinka siis vaatimusten määrittely voidaan toteuttaa laadukkaasti, jotta tietojärjestelmästä saadaan asiakkaan vaatimuksia vastaava ja kehitysprojekti pysyy aikataulussaan ja budjetissaan? Alan kirjallisuuden mukaan vaatimusmäärittelijöiden täytyy ymmärtää eri sidosryhmien tarpeet ja *todellinen* ongelma, joka kehitettävällä tuotteella täytyy ratkaista. Lisäksi vaatimusmäärittelijöiden tulee ymmärtää vaatimusten liittyminen ohjelmiston ympäristöön, sekä muihin prosesseihin ja järjestelmäympäristöihin. Vaatimukset täytyy osata myös kuvata yksiselitteisesti ja ristiriidattomasti, ja vaatimuksiin tulevia muutoksia täytyy pystyä hallitsemaan, jotta vaatimusmäärittelyssä voidaan onnistua. (Haikala & Mikkonen 2011, 64; Pohl & Rupp 2011, 11-18; Robertson & Robertson 2013, xxii.)

Opinnäytetyöni käsittelee vaatimusmäärittelyn merkitystä tietojärjestelmän kehittämisessä. Työn teoriaosuuden tavoitteena on tutkia alan kirjallisuuden avulla vaatimusmäärittelyn prosessia, kuinka se tulisi toteuttaa ja mitä asioita sen toteuttamisessa tulee ottaa huomioon, jotta erityisesti toiminnallisten vaatimusten määrittelydokumentista saadaan kehitettävän järjestelmän suunnittelua, toteuttamista ja testausta tukeva dokumentti. Työn alussa esitellään vaatimuksen käsite, vaatimusluokat ja vaatimusmäärittelijöiden roolit. Sen jälkeen käsitellään vaatimusmäärittelyn prosessi. Opinnäytetyön pääpaino on vaatimusmäärittelyn prosessissa, jonka osa-alueista tarkastellaan hieman laajemmin yhtä osa-aluetta, vaatimusten dokumentointia. Dokumentoinnin tarkemmalla tutkimisella halutaan saada käsitys siitä, mikä merkitys dokumentoinnilla on vaatimusmäärittelyssä ja sitä kautta ohjelmiston kehittämisessä, sekä minkälaisilla menetelmillä vaatimukset tulisi tuottaa. Lopuksi käydään lyhyesti läpi vaatimusten validointivaihetta ja vaatimustenhallintaa. Työssä esitellään lyhyesti myös yksi tietojärjestelmän kehittämisen elinkaarimalli, vesiputousmalli,

jota käytettiin työn empiriaosuuden toteuttamisessa. Opinnäytetyö toimii opinnäytetyön kirjoittajalle oman oppimisen prosessina.

Tämän opinnäytetyön empiriaosuudessa kuvataan Yritys X:n ylläpitovaiheessa olevan tietojärjestelmän pienkehitystyö, joka toteutettiin vuonna 2014. Opinnäytetyön kirjoittaja oli kyseisessä kehitysprojektissa yhtenä projektiryhmän jäsenenä määrittelemässä käyttäjien vaatimuksia, sekä testaamassa määriteltyjä vaatimuksia järjestelmän pääkäyttäjän näkökulmasta, jonka vuoksi sain työnantajaltani luvan käyttää kehitysprojektia opinnäytetyöni empiriaosuuden perustana. Pienkehitystyössä käytettiin ohjelmistoprosessimallina pääosin iteroivaa vesiputousmallia, mutta osittain myös prototyyppilähestymistapaa, jolla tarkoitetaan ohjelmistotuotannossa vaillinaisen prototyypin rakentamista ohjelman joidenkin ominaisuuksien tutkimista varten (Haikala & Mikkonen 2011, 38). Ohjelmistoprosessimallin vaiheista käydään läpi esitutkimus, määrittely ja testaus. Kyseisistä vaiheista kuvataan, kuinka vaatimusten määrittelyn tuottaminen toteutui käytännössä esitutkimus- ja määrittelyvaiheessa. Lisäksi tarkastellaan, kuinka hyvin vaatimusten määrittely tuki testausvaihetta tämän kehitysprojektin kohdalla. Tarkoituksena on peilata teorian ja käytännön eroavaisuuksia ja tarkastella, kuinka hyvin teorian opit toteutuvat käytännön työssä.

Empiirisen osuuden aineistona käytetään vuonna 2014 toteutetun Yritys X:n tietojärjestelmän pienkehitystyöprojektin aikana tuotettuja dokumentteja, jotka liittyvät Yritys X:n kuvatyönkulkuprojektiin. Lisäksi empiirisen osuuden kuvaamisessa hyödynnetään opinnäytetyön kirjoittajan kokemusnäkökulmaa kyseisen pienkehitystyöprojektin osalta.

Tämän opinnäytetyön tavoitteena on löytää vastaus seuraaviin kysymyksiin: Miten vaatimusmäärittelyprosessi tulisi toteuttaa? Mitä tekniikoita vaatimusten dokumentoimisessa tulisi käyttää? Kuinka vaatimusmäärittelyprosessi ja vaatimusmäärittelydokumentin tuottaminen toteutui käytännössä pienkehitystyöprojektin kohdalla kohdeorganisaatiossa? Voisiko kohdeorganisaatio kehittää vaatimusmäärittelyään jollain tavalla?

Opinnäytetyö keskittyy tarkastelemaan vaatimusmäärittelyä tietojärjestelmän ylläpitovaiheessa. Työn ulkopuolelle jää pienkehitystyön kohteena olevan tietojärjestelmän suunnittelun ja toteutuksen tekniset kuvaukset, sekä valitun Media-arkistojärjestelmän ja vertailussa mukana olleen toisen järjestelmän, FotoWaren, teknologian tarkempi kuvaus. Toteutetusta pienkehitystyöstä ei kuvata ohjelmistoprosessimallin suunnittelu-, toteutus-, käyttöönotto- ja ylläpitovaiheita.

2 Vaatimusmäärittely, määrittelyn prosessi ja vaatimustenhallinta

Robertsonin & Robertsonin (2013, 1-2) mukaan vaatimusten määrittelyn päätavoite on keskittää huomio liiketoiminnan ongelman ymmärtämiseen ja tarjota sopivin ratkaisu liiketoiminnan ongelmaan. Jos tarkoituksena on rakentaa sovellus, sen täytyy tarjota optimaalista arvoa sen omistajalle. Tämä hyöty tulee yleensä esille siten, että sovellus tarjoaa kyvykkyyttä, joka ei ollut aiemmin mahdollista, tai sovellus muuttaa jonkin liiketoimintaprosessin nopeammaksi tai paljon kätevämmäksi. Jotta arvo on optimaalista, kehitettävän tuotteen täytyy tarjota hyöty, joka on tasapainossa tuotteen kustannusten kanssa.

Kehittäjät ovat kehittäneet kaikista hyödyllisimmät tuotteet silloin, kun he ovat ymmärtäneet täsmällisesti, mitä kehitettävällä tuotteella on tarkoitus saada aikaan tuotteen käyttäjille ja millä tavalla se on tarkoitus saavuttaa (Robertson & Robertson 2013, 3).

Valitettavasti vaatimuksia ei kuitenkaan aina ole ymmärretty oikein, koska tutkimusten mukaan 60 % ohjelmistoprojektien epäonnistumisista saa alkunsa vaatimusten määrittelyvaiheen aikana tapahtuneista virheistä. Jos vaatimusten määrittelyä ei tehdä prosessin alussa kunnolla, vaatimusten heikko laatu vaikuttaa tuotteen koko kehityksen elinkaareen, jolloin tuotteen hinta saattaa moninkertaistua. (Robertson & Robertson 2013, 4; Haikala & Mikkonen 2011, 61.)

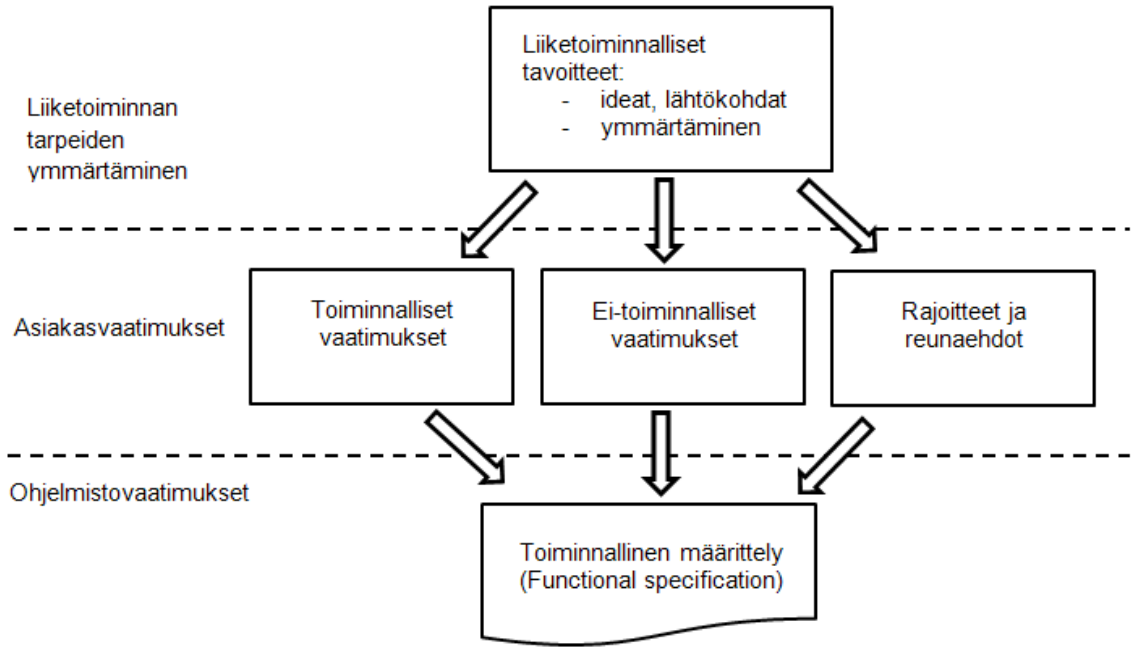
2.1 Vaatimuksen käsite ja vaatimusluokkia

Vaatimusten määritellään olevan ominaisuuksia tai rajoitteita, joita toteutettavalla tietojärjestelmällä tulee olla tai mitä tietojärjestelmällä pystyy tekemään (Haikala & Mikkonen 2011, 61; Robertson & Robertson 2013, 1). Vaatimukset määrittelevät eri sidosryhmien tarpeet järjestelmän suhteen, mutta eivät ota kantaa siihen, minkälainen tekninen toteutus nämä tarpeet käytännössä täyttää (Pohjonen 2002, 28; Robertson & Robertson 2013, 26).

Määrittelyvaiheen ydintehtävä on kyetä kuvaamaan kuinka toteutettava tietojärjestelmä tukee käyttäjien toimintaa, sekä muuttaa asiakasvaatimukset täsmällisiksi ohjelmistovaatimuksiksi, jotka määrittelevät toteutettavan järjestelmän (Haikala & Märijärvi 2006, 38). Määrittelyvaiheen tuloksena syntyy yleensä muodollinen dokumentti, jolle ei ole olemassa yhtä standardia nimeä, vaan siitä käytetään eri organisaatioissa erilaisia nimityksiä, kuten esimerkiksi vaatimusdokumentti, vaatimusmäärittelydokumentti, järjestelmän vaatimusten spesifikaatio tai toiminnallinen määrittelydokumentti (Kotonya & Sommerville 2004, 15; Pohjonen 2002, 30).

Toiminnallisessa määrittelydokumentissa, jota käytetään kommunikoinnin välineenä eri sidosryhmien välillä koko kehittämisprosessin aikana, kuvataan ohjelmiston asiakasvaatimukset, jotka luokitellaan yleensä kolmeen luokkaan (kuva 1):

- toiminnalliset vaatimukset,
- ei-toiminnalliset vaatimukset (laatuvaatimukset),
- rajoitteet/reunaehdot (Haikala & Märijärvi 2006, 63-64; Pohjonen 2002, 28; Pohl & Rupp 2011, 8-10; Robertson & Robertson 2013, 10-11).



Kuva 1. Asiakas- ja ohjelmistovaatimukset (Haikala & Mikkonen 2011, 62).

Toiminnalliset vaatimukset määrittävät, mitä ohjelmiston täytyy tehdä. Nämä vaatimukset ovat niin sanottuja liiketoiminnan vaatimuksia, joka tarkoittaa sitä, että ne määrittävät ohjelmiston tarpeet tukeakseen liiketoimintaa. (Robertson & Robertson 2013, 10, 26, 225.) Toiminnallisiin vaatimuksiin luokitellaan muun muassa käyttöliittymä ja liittymät muihin järjestelmiin, miten eri sidosryhmät ovat yhteydessä järjestelmään ja miten he työskentelevät järjestelmän kanssa. (Haikala & Märijärvi 2006, 39, 63-64; Pohjonen 2002, 28.) Esimerkiksi kuvapankin asiakkaiden täytyy pystyä lataamaan kuvia omalle työasemalleen weblinkin kautta ilman erillistä kirjautumista järjestelmään.

Ei-toiminnalliset vaatimukset ovat laatuvaatimuksia, joita kehitettävällä ohjelmistolla täytyy olla ja ne määrittävät, kuinka hyvin ohjelmisto tekee siltä vaadittavat asiat. Ei-

toiminnallisten vaatimusten tehtävä on tukea toiminnallisia vaatimuksia. (Robertson & Robertson 2013, 10, 245-246, 249). Laatuvaatimuksiin kuuluvat esimerkiksi järjestelmän käytettävyys, suoritusteho, vasteaika, tietoturva ja siirrettävyys. (Haikala & Märijärvi 2006, 39, 63-64; Pohl & Rupp 2011, 8-10). Esimerkiksi sisällönhallintajärjestelmän tulee toimia sekä Windows 10, että Macintosh 10 –ympäristöissä tai kuvien haku saa kestää maksimissaan 3 sekuntia/haku.

Rajoitteet/reunaehdot voidaan nähdä globaaleina vaatimuksina. Ne auttavat määrittämään, mikä vaatimusten osajoukko voidaan sisällyttää lopulliseen ohjelmistoon. (Robertson & Robertson 2013, 11, 59). Rajoitteet/reunaehdot ovat ei-toiminnallisten vaatimusten erikoistapauksia, joilla määritellään rajoitteita ja reunaehtoja järjestelmälle asetetuille toiminnallisille vaatimuksille (Pohjonen 2002, 28). Reunaehtoihin ja rajoituksiin luetaan tyypillisesti laitteistolle ja ohjelmistolle asetetut rajoitteet, hinta, toteutusaikataulu, käytettävät työkalut, muistitila, joka järjestelmällä on käytettävissä, ohjelmointikieli, jolla järjestelmä tulee toteuttaa, sekä lait ja standardit (Haikala & Mikkonen 2011, 61; Haikala & Märijärvi 2006, 39, 63-64; Pohl & Rupp 2011, 8-10).

Asiakasvaatimuksille on olemassa tarkkoja kriteereitä, millaisia niiden tulee olla, jotta ne voidaan luokitella hyvin määritellyiksi vaatimuksiksi. Hyvän asiakasvaatimuksen tulee olla yksiselitteinen, tarkka, ymmärrettävä, testattava, jäljitettävä taaksepäin ja eteenpäin. (Haikalan & Mikkosen 2011, 64). Vaatimus on yksiselitteinen, jos vaatimuksen tarkoitus on sama sekä sidosryhmälle, jolta vaatimus on peräisin, että kehittäjälle, joka rakentaa vaatimuksen (Robertson & Robertson 2013, 21). Riittävän tarkalla vaatimuksella voidaan varmistaa, että vaatimuksen täyttyminen on mitattavissa. Testattavalla vaatimuksella pystytään mittaamaan, että vaatimus on täytetty. Taaksepäin jäljitettävyydellä voidaan selvittää vaatimuksen alkuperä. Eteenpäin jäljitettävyydellä voidaan jäljittää vaatimuksen tekninen toteutus ja vaatimukseen liittyvät testitapaukset. (Haikalan & Mikkosen 2011, 64.)

2.2 Vaatimusmäärittelijöiden roolit

Vaatimusmäärittelyssä käytetään toteutustapana yleensä projektia, johon osallistuu linjaorganisaatiosta eri tyyppisiä asiantuntijoita, jotka koostuvat esimerkiksi projektipäälliköstä, tietojärjestelmän omistajasta, prosessien omistajista, yksiköiden asiantuntijoista eli käyttäjistä, tietohallinnon suunnittelijoista, kehittäjistä ja tarvittaessa ulkopuolisista asiantuntijoista. (JUHTA – Julkisen hallinnon tietohallinnon neuvottelukunta 2012.)

Projektipäällikkö vastaa kokonaisuudesta, johon kuuluu vaatimusmäärittely, projektin ositus, resursointi sekä viestintä ja yhteydenpito eri sidosryhmiin. Tietojärjestelmän omistajan vastuulle kuuluu vaatimusmäärittelytyön ohjaus ja valvonta sekä määrittelykuvauksien hyväksyminen. Yksiköiden asiantuntijat eli käyttäjät vastaavat järjestelmän toiminnallisista ja ei-toiminnallisista kuvauksista joko suullisesti ja/tai kirjallisesti. Tietohallinnon asiantuntija vastaa puolestaan siitä, että toiminnallisten vaatimusten kuvaukset tuotetaan tietyn standardin mukaisesti esimerkiksi UML-mallinnuksen (engl. Unified Modeling Language) eli graafisen mallinnuskielen avulla. Mikäli projektissa käytetään ulkopuolisia vaatimusmäärittelytyön asiantuntijoita, he vastaavat esimerkiksi vaatimusmäärittelyn suunnittelusta ja menetelmien koulutuksesta sekä ohjauksesta. (JUHTA – Julkisen hallinnon tietohallinnon neuvottelukunta 2012).

Tietojärjestelmän kehittäjän näkökulmasta tärkeimmän ryhmän järjestelmän kehittämiseen osallistuvista ihmisistä muodostavat käyttäjät, jotka ovat kohdealueen asiantuntijoita ja jotka tulevat käyttämään rakennettavaa tietojärjestelmää päivittäisissä työtehtävissään. Pohjonen huomauttaa, että käyttäjien ja kehittäjien välinen kommunikonti on usein ongelmallista, johtuen molempien tahojen erilaisista odotuksista ja tavoitteista ja vaikeudesta löytää yhteinen kieli. Kommunikaatio-ongelmaa voi yrittää ratkaista esimerkiksi sillä tavoin, että tutustuttaa käyttäjät tulevan järjestelmän ominaisuuksiin mahdollisimman varhaisessa vaiheessa ja kommunikaation tulisi olla riittävän monipuolista ja avointa. (Pohjonen 2002, 47, 49-51.)

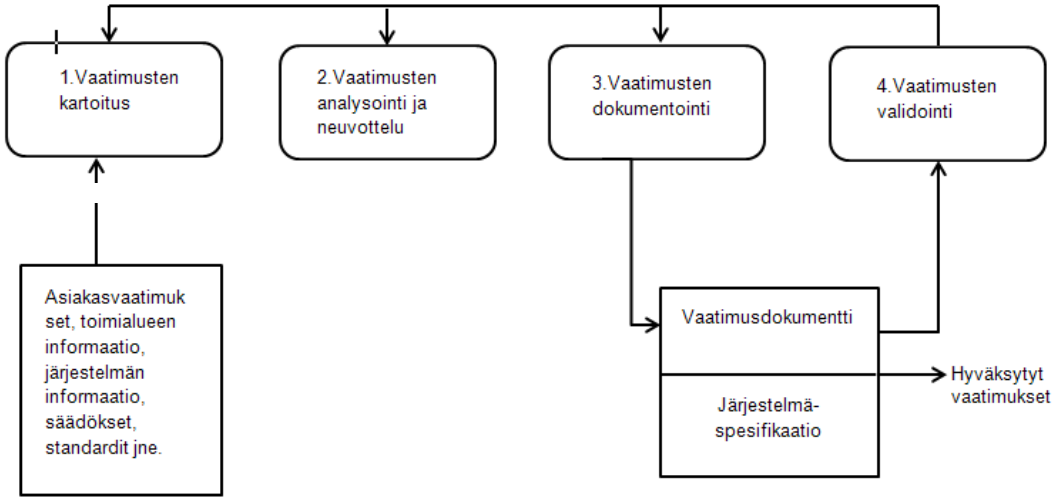
Vaatimusmäärittelyn prosessi

Vaatimusmäärittelyprosessista on olemassa useita erilaisia tapoja, joissa prosessiin kuuluvien aktiviteettien eli toimintojen nimeämiset vaihtelevat ja niiden käsittelyn järjestys vaihtelee. Kotonya & Sommerville (2004, 4-6) toteavatkin, että vaatimusmäärittelyn prosessiin ei ole olemassa yhtä ideaalia tapaa, vaan jokaisen organisaation tarpeisiin täytyy räätälöidä oma vaatimusten käsittelyprosessinsa.

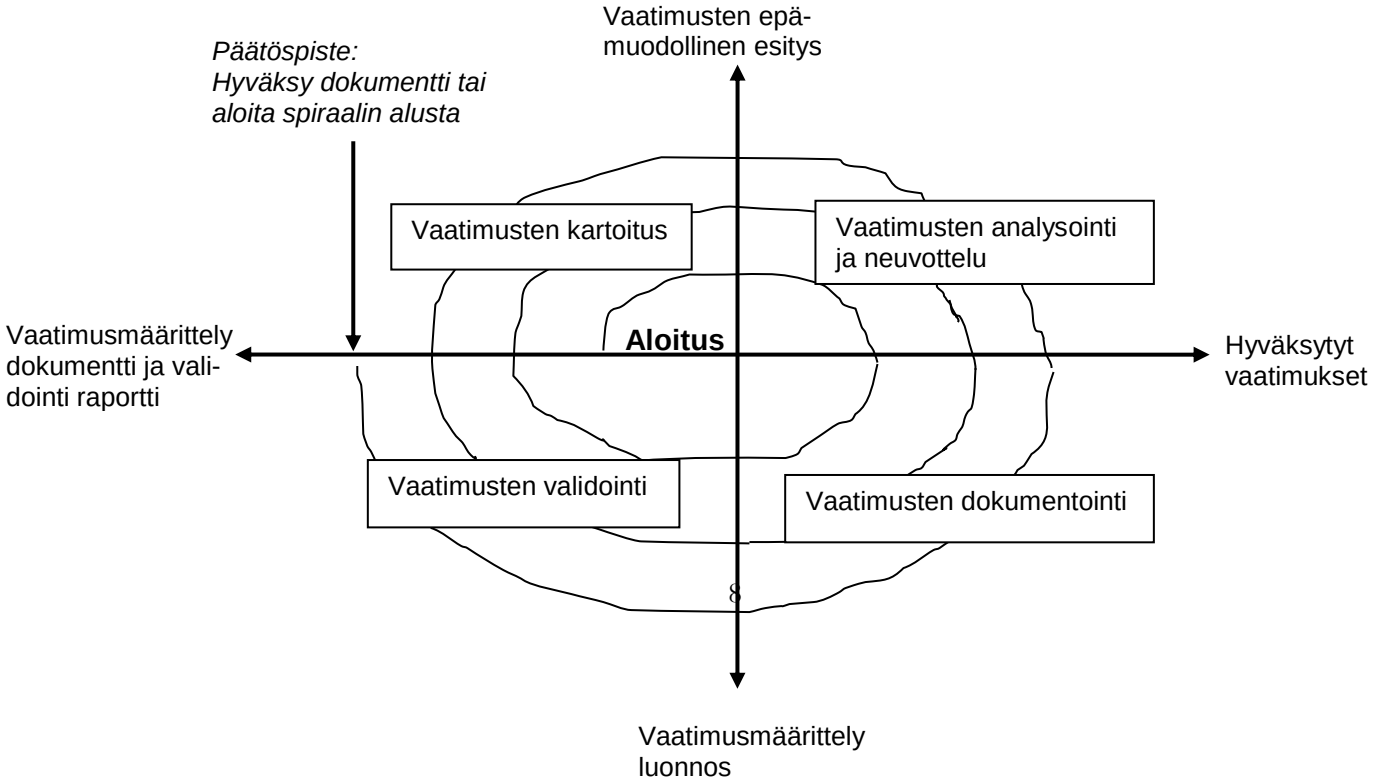
Kotonya & Sommerville (2004, 32-33) jakavat vaatimusmäärittelyprosessin aktiviteettien vaiheet vaatimusten kartoittamiseen, analysointiin ja neuvotteluun, dokumentointiin ja validointiin (kuva 3). Vaihtoehtoinen tapa esittää vaatimusmäärittelyprosessin aktiviteettien vaiheet, on spiraalimalli (kuva 4), joka korostaa prosessin inkrementaalisuutta. Se tekee aktiviteettien toistumisen selkeämmäksi. Kuva osoittaa, että vaatimusmäärittelyprosessin eri toimintoja toistetaan kunnes tehdään päätös, että vaatimusmäärittelydokumentti hyväksytään. Jos vaatimusmäärittelydokumentin luonnoksesta löytyy ongelmia, vaatimusten

kartoitus-, analysointi ja neuvottelu-, dokumentointi- ja validointispiraali aloitetaan uudelleen. Tämä jatkuu, kunnes on tuotettu hyväksytty vaatimusmäärittelydokumentti tai kunnes ulkoiset tekijät, kuten aikataulupaineet tai resurssien puute pakottavat lopettamaan vaatimustenmäärittelyprosessin.

Kuva 3. Vaatimusmäärittelyprosessin toimintamalli (Kotonya & Sommerville 2004, 32).



Kuva 4. Vaatimusmäärittelyprosessin spiraalimalli (Kotonya & Sommerville 2004, 35).



Tässä opinnäytetyössä vaatimusmäärittelyprosessi kuvataan Kotonyan & Sommervillen toimintamallin mukaisesti, jonka avulla tutkitaan, kuinka vaatimusmäärittelyprosessi tulisi toteuttaa ja mitä asioita sen toteuttamisessa tulisi ottaa huomioon, jotta erityisesti toiminnallisten vaatimusten määrittelydokumentista saataisiin tietojärjestelmän kehittämistä tukeva dokumentti. Prosessin ensimmäinen vaihe alkaa vaatimusten kartoittamisesta.

2.3 Vaatimusten kartoittaminen

Robertson & Robertson (2013, 16-17) suosittelevat, että vaatimusmäärittelyprojekti käynnistetään aloituskokouksella, johon kootaan pääsidosryhmät, eli ihmiset, jotka ovat välttämättömiä projektin menestymiselle, jotta saavutetaan yhteisymmärrys projektin tavoitteista ja liiketoiminnan syystä projektin tekemiselle. Projektin käynnistyskokous määrittelee liiketoiminnan ongelman laajuuden ja varmistaa samanaikaisesti sidosryhmiltä kehitettävän alueen. Kokous vahvistaa myös mitkä toiminnallisuudet täytyy sisällyttää vaatimusten kartoittamiseen ja mitkä toiminnallisuudet on erityisesti suljettu pois. Kun kokoukseen osallistujat ovat saavuttaneet yhteisymmärryksen liiketoiminta-alueen laajuuden tutkimisessa, ryhmä tunnistaa sidosryhmät, joilla on tietoa, joka liittyy tuotteeseen tai joilla on vaatimuksia toteutettavan tuotteen suhteen. Nämä ihmiset täytyy tunnistaa, jotta vaatimusmäärittelijä voi aloittaa työskentelyn heidän kanssaan löytääkseen kaikki tarvittavat vaatimukset.

Kun projektin käynnistys on tehty, vaatimusten määrittelijät aloittavat asiakasvaatimusten kartoittamisen eli keräämisen. Se on liiketoiminnan työn tutkimista, jonka kautta saadaan ymmärrys siitä mitä liiketoiminta tekee. Ensimmäinen asia vaatimusten kartoittamisessa on tutkia ja saada ymmärrys siitä, millä tavalla työ tehdään tällä hetkellä. Heti kun on saavutettu ymmärrys työn perusolemuksesta, siitä riisutaan pois tämän hetkinen teknologia, jotta saadaan puhdas kuva todellisesta liiketoiminnasta. Kartoittamisvaiheen olennaisin tehtävä on kehittää uudelle ohjelmistolle vaatimukset yhdessä sidosryhmien kanssa. (Robertson & Robertson 2013, 17-18, 87, 90.)

Vaatimusten kartoittaminen on vaativa ja aikaa vievä työvaihe. Vaatimukset, jotka kerätään projektin alussa, ovat lähes poikkeuksetta puutteellisia ja usein huonosti määriteltyjä. Lisäksi vaatimukset voivat olla keskenään ristiriitaisia ja monitulkintaisia. (Pohjonen 2002, 28.) Asiakasvaatimusten kartoittamista hankaloittaa lisäksi se, että vaatimusten kartoittamiseen ei ole olemassa yhtä ainoaa menetelmää, jonka avulla saataisiin riittävän yksiselitteinen, oikea ja kattava lopputulos. Sen vuoksi vaatimusten kartoittamiseen sovelletaan

samanaikaisesti useita erilaisia menetelmiä. (Haikala & Märijärvi 2006, 94-95.) Sama tekniikka ei sovellu jokaiseen projektiin, joten vaatimusmäärittelijän on valittava projektikohtaisesti sopivin menetelmä vaatimusten löytämiseen (Robertson & Robertson 2013, 91).

Pohjosen (2002, 28) mukaan perinteisin tapa asiakasvaatimusten keräämiseksi on järjestelmän eri sidosryhmien edustajien haastattelut, jotka tarjoavat myös hyvän tilaisuuden kerätä tietoa järjestelmän tulevista käyttäjistä sekä järjestelmän ympäristöön liittyvistä yleisistä käytänteistä. Muita tapoja vaatimusten keräämiseen ovat esimerkiksi aivoriihet, ideointipalaverit, workshopit, markkinatutkimukset, kilpailijoiden tuotteiden tutkiminen ja prototyyppien käyttäminen. Näiden lisäksi asiakasvaatimuksia saadaan kerättyä tarkastelemalla ja analysoimalla organisaation jo olemassa olevia tietojärjestelmiä ja toimintatapoja.

Robertson & Robertson (2013, 19, 92) listaavat asiakasvaatimusten kartoittamiseen pääosin samoja menetelmiä kuin Pohjonenkin, mutta he lisäävät listaan vielä muutamia erilaisia menetelmiä: Post-it muistilaput, oppipoikamenetelmän, liiketoiminnan käyttötapaukset ja skenaariot.

Post-it muistilappujen käyttäminen toiminnallisuuden mallintamisessa on nopea mallinnustekniikka, jossa jokaista muistilappua voidaan käyttää kuvaamaan aktiviteettia ja muistilaput voidaan nopeasti järjestää uudelleen näyttämään erilaisia tapoja, joilla työ on tehty tai voitaisiin tehdä. Oppipoikamenetelmässä vaatimusmäärittelijä istuu työntekijän vieressä, kun työntekijä kuvailee työtä, jota hän parhaillaan tekee ja kertoo tavoitteitaan työlle, mitä hän toivoo, että sillä voisi tehdä. Samalla vaatimusmäärittelijä tekee havaintoja, kysyy kysymyksiä ja mahdollisesti kirjaa samalla alustavaa vaatimusmallia työstä. Liiketoiminnan käyttötapauksia Robertsonit pitävät vaatimusten kartoittamisen kulmakivenä. Jos vaatimukset löytyvät *toiminnasta*, he suosittelevat niiden kuvaamista liiketoiminnan käyttötapausten (engl. business use case, BUC) avulla, yksi käyttötapaus kerrallaan. He kehottavat tämän metodin käyttämistä vaatimusten kartoittamisen yhtenä keräämistapana, riippumatta käytettävän projektin tyypistä. Skenaarion, eli tarinan avulla tutkitaan liiketoiminnan käyttötapausta ja määritellään siinä tehtävä työ askel askeleelta. (Robertson & Robertson 2013, 19, 92, 98-99, 130.)

Vaatimusten analysoinnista puhutaan erillisenä toimintona, joka seuraa vaatimusten kartoittamista. Käytännössä vaatimusten kartoittaminen ja analysointi ovat kuitenkin läheisesti toisiinsa lomittuvia prosesseja ja niitä tehdään osittain yhtä aikaa. (Kotonya & Sommerville 2004, 57.)

2.4 Vaatimusten analysointi ja neuvottelu

Vaatimusten analysointivaihe tutkii asiakasvaatimuksia, jotka saadaan sidosryhmiltä. Vaatimukset ovat analysointivaiheessa usein epätäydellisiä ja ne on ilmaistu epämuodollisesti. Analysoinnin tavoitteena onkin varmistaa, että vaatimukset kohtaavat sidosryhmien tarpeet, eikä vaatimusten kuvausten tarvitse olla yksityiskohtaisesti oikein vielä tässä vaiheessa. Vaatimusten analysointivaihe etsii vastausta kysymykseen: ”Olemmeko kirjanneet oikeat vaatimukset?” (Kotonya & Sommerville 2004, 88).

Analysointivaiheessa vaatimuksia tutkitaan yksityiskohtaisesti ja tarvittaessa vaatimuksia tarkennetaan sekä selvitetään niiden keskinäisiä suhteita ja prioriteettia, eli tärkeyttä. Vaatimukset voivat olla ristiriitaisia joko keskenään tai ominaisuuksien kanssa, jotka on toteutettu järjestelmän aikaisemmassa versiossa. Vaatimusten yhteensovittamisessa joudutaan tekemään kompromisseja sillä periaatteella, että taataan ohjelmiston säilyvyys sekä ulkoisilta että sisäisiltä ominaisuuksiltaan yhdenmukaisena. (Haikala & Märijärvi 2006, 96).

Vaatimusten prioriteetin arvioinnilla helpotetaan päätöstä, kun mietitään mitkä ominaisuudet ohjelmiston seuraavaan versioon otetaan mukaan ja mitkä ominaisuudet voidaan jättää toteutettavaksi myöhemmässä vaiheessa. Vaatimuksille voidaan asettaa esimerkiksi seuraavia prioriteetteja: välttämätön (kriittinen ominaisuus, ilman tätä ominaisuutta ei tulla toimeen), toivottu *(*tärkeä toiminto, mutta tullaan toimeen ilmankin*)* ja valinnainen (voidaan toteuttaa jossain myöhemmässä versiossa). Vaatimusten priorisointia täytyy tarkastella loppujen lopuksi liiketoiminnan kannalta ja tutkia muun muassa vaatimusten aiheuttamia kustannuksia odotettuihin tuottoihin verrattuna, sekä vaatimusten toteuttamiseen liittyviä aikatauluriskejä. (Haikala & Märijärvi 2006, 96).

Kun vaatimuksia analysoidaan, siitä seuraa, että alustaviin vaatimuksiin tulee muutoksia, niitä yhdistellään ja saatetaan jopa löytää aivan uusia vaatimuksia. Kun analysoidut vaatimukset ryhmitellään ja numeroidaan, niihin on mahdollista viitata muista dokumenteista. Myös vaatimuksen muutosherkkyyttä, eli vaatimuksen muutoksen todennäköisyyttä, on mahdollista yrittää arvioida analysointivaiheessa. Muutosherkkyyksien analysointi auttaa projektin riskien arvioinnissa ja sopivan vaihejakomallin valitsemisessa projektia varten. Silloin, kun vaatimukset ovat riittävän stabiileja, perinteisen vesiputousmallin käyttäminen on mahdollista. Vastaavasti jos muutosherkkyys on suuri, ovat esimerkiksi prototyyppilähestymistapa ja inkrementaalinen kehittäminen, jossa tuotekehityssyklit ovat lyhyitä,

soveltuvampia vaihtoehtoja projektia varten. Muutosherkkyyden vaikutus ulottuu myös suunnitteluvaiheen ohjelmistoarkkitehtuurin valintaan. (Haikala & Märijärvi 2006, 96-97).

Neuvottelujen tuloksena eri sidosryhmät päättävät, mitkä vaatimukset järjestelmään hyväksytään. Tämä prosessi on välttämätön koska konfliktit ovat lähes väistämättömiä eri sidosryhmien välillä, informaatio saattaa olla epätäydellistä ja kaikkien vaatimusten toteuttaminen saattaa ylittää kehitettävälle järjestelmälle varatun budjetin. Yleensä vaatimuksissa on myös joustavuutta, joten neuvottelu on tarpeen, jotta päätetään ne vaatimukset, jotka on tärkeää toteuttaa järjestelmään. (Kotonya & Sommerville 2004, 32-33; Haikala & Mikkonen 2011, 66.)

Vaatimusten analyysi- ja neuvotteluvaiheet ovat kalliita ja aikaa vieviä prosesseja, koska asiantuntijoiden täytyy lukea dokumentit huolellisesti ja pohtia vaatimusten käyttökelpoisuutta, tarpeellisuutta ja johdonmukaisuutta, sekä sitä kuinka esitetyt vaatimukset voidaan toteuttaa. Vaatimusten analysoinnissa voidaan käyttää erilaisia tarkistuslistoja, joissa on lista kysymyksistä, joita analysoija voi käyttää arvioidakseen jokaisen vaatimuksen erikseen. Apuna voidaan käyttää myös esimerkiksi vuorovaikutusmatriiseja, joissa verrataan vaatimuksia niiden ristiriitaisuuden, päällekkäisyyden ja itsenäisyyden perusteella. (Kotonya & Sommerville 2004, 77-80.) Sen jälkeen kun vaatimuksia on analysoitu, testattu ja neuvottelujen tuloksena on päätetty mitkä vaatimukset hyväksytään, seuraa hyväksyttyjen vaatimusten dokumentointi.

2.5 Vaatimusten dokumentointi

Pohjosen (2002, 30) sekä Pohlin & Ruppin (2011, 33) mukaan vaatimusten dokumentoinnilla on tärkeä merkitys, koska järjestelmän kehittäminen perustuu vahvasti määritettyihin vaatimuksiin. Kaikki vaatimukset vaikuttavat analyysiin, suunnitteluun, toteutukseen ja testausvaiheisiin suoraan ja epäsuorasti. Lisäksi vaatimusdokumentin laatu vaikuttaa merkittävällä tavalla siihen, kuinka hyvin projekti edistyy ja kuinka onnistunut projekti lopulta on.

Robertsonin & Robertsonin (2013, 5, 32-33) näkemys on, että joissakin tapauksissa on tehokkaampaa kommunikoida vaatimukset suullisesti ja toisissa tapauksissa taas on väärjäämätön tarve kirjoittaa vaatimusdokumentti. Monissa tapauksissa kuitenkin vaatimusten kirjoittaminen auttaa merkittävästi sekä vaatimusmäärittelijää, että sidosryhmiä ymmärtämään vaatimukset oikein. Vaatimusdokumentin tuottamiseen vaikuttaa merkittävästi myös

projektin koko. Jos kyseessä on pieni projekti, dokumentin tuottaminen on epämuodollisempaa kuin keskisuurissa ja suurissa projekteissa, tai jos työtä ollaan ulkoistamassa.

Haikala & Märijärvi (2006, 66) kuitenkin huomauttavat, että hyvät vaatimusmäärittelydokumentitkaan eivät takaa sitä, että projektin lopputulos on onnistunut. Todellinen onnistuminen mitataan asiakkaan saamalla hyödyllä suhteessa käytettyyn investointiin.

Vaatimusmäärittelydokumentin sisällön esittämiseen on olemassa useita erilaisia tapoja, johon vaikuttavat esimerkiksi kehitettävän järjestelmän tyyppi, vaatimuksiin sisällytettyjen yksityiskohtien määrä, organisaation käytännöt, vaatimusmäärityksprosessin budjetti ja aikataulu. Jotta voidaan varmistaa, että dokumenttiin on sisällytetty kaikki olennainen tieto, organisaation täytyy määritellä itse omat standardit vaatimusdokumentin sisällölle ja muokata itselleen sopiva dokumenttimalli. (Kotonya & Sommerville 2004, 15.) Robertsonin & Robertsonin (2013, 393-471) suositteleman vaatimusmäärittelydokumentin mallipohjaan on sisällytetty seuraavia asioita:

1. **Projektin tarkoitus**. Perustele miksi järjestelmä halutaan kehittää. Kuvaa liiketoiminnan ongelma ja kuinka kehitettävän järjestelmän on tarkoitus ratkaista ongelma.
2. **Sidosryhmäkuvaukset**. Listaa ihmiset, jotka liittyvät jollain tavalla järjestelmän ympäristöön. Kuvaa ja kategorisoi järjestelmän tulevat käyttäjät.
3. **Järjestelmän rajoitteet**. Kuvaa tämän hetkisen järjestelmän ympäristön toteutus. Yhteissovellukset, valmisohjelmistot, kuvaus kohdealueesta, jossa sovellusta käytetään, aikataulurajoitukset, budjettirajoitukset, yhtiön rajoitukset.
4. **Terminologia**. Määrittele kaikki termit, sisältäen kirjainlyhenteet, joita sidosryhmät käyttävät projektiin liittyen.
5. **Relevantit faktat ja oletukset**. Kirjaa liiketoiminnan säännöt, jotka vaikuttavat vaatimuksiin, sekä kehittäjien oletukset, jotka liittyvät kehitettävään järjestelmään.
6. **Toiminnan kohteeksi rajattu alue**. Kuvaa tämän hetkiset liiketoimintaprosessit. Tee sidosryhmäkaavio, jossa tutkittava kohde on rajattu ympäristöstään. Esitä taulukkomuotoinen kuvaus jokaisesta ulkoisesta sidosryhmästä ja kerro sidosryhmien merkitys kohdealueelle, kirjaa tapahtumat yksilöivällä tunnisteella, sekä kirjaa tietovirrat, jotka virtaavat kohdealueen ja sidosryhmien välillä. Kuvaa kohdealueen työtoimintaa liiketoiminnan käyttötapauksina (BUC) sekä kirjoittamalla niistä myös skenaariot. Kuvauksissa voit visioida myös tulevaa toimintaa, niin että saadaan jo kuva tulevan järjestelmän käyttötilanteista.
7. **Käsitemallit**. Kirjaa säilytettävien tietojen vaatimukset.

8. **Kehitettävän tuotteen kohteeksi rajattu alue**. Kuvaa kehitettävän tuotteen käyttötapauskaavio.
9. **Toiminnalliset vaatimukset.**
10. **Ei-toiminnalliset vaatimukset.**
11. **Projektin muut asiat.** Avoimet asiat, uudet ongelmat, riskit, kustannukset.

Asiakasvaatimusten dokumentointitekniikasta ja dokumenttimalleista ei anneta myöskään vain yhtä suositusta, vaan jokainen organisaatio voi valita itselleen sopivimman tavan dokumentoida vaatimukset. Vaatimusten dokumentoimiseen on olemassa valmiita mallipohjia, joissa on opastettu mitä tietoa vaatimuksista tulee kirjata ja millä tavalla. Robertson & Robertson (2013, 29-31) esittävät niin sanotun Snow Card –mallin, jonka mukaan vaatimukset voidaan dokumentoida. Snow Card –mallissa yksilöllisillä vaatimuksilla on attribuutit eli määritteet, jossa jokainen määrite tukee jollakin tavalla vaatimuksen ymmärtämistä ja vaatimuksen tarkkuutta ja siten ohjelmiston kehityksen virheettömyyttä. Mallissa asiakasvaatimuksista dokumentoitavia asioita ovat: vaatimuksen yksilöllinen tunnistetieto, vaatimustyyppi, liiketoiminnan tapahtuman/BUC:in/PUC:in yksilöllinen tunnistetieto, vaatimuksen kuvaus, vaatimuksen perustelu, vaatimuksen esittäjä, hyväksymiskriteeri, vaatimuksen tärkeys asiakkaalle, prioriteetti, konfliktit, mahdolliset muut vaatimusta tukevat materiaalit, vaatimuksen versio/luontipäivä/muutokset. Malli on esitetty tarkemmin liitteessä 1.

Dokumentoinnin avulla on tarkoitus helpottaa kommunikointia sidosryhmien kanssa ja samalla lisätä vaatimusmäärittelydokumentin laatua. Asiakasvaatimusten dokumentointiin voidaan käyttää esimerkiksi luonnolliseen kieleen perustuvia menetelmiä tai kaavioihin perustuvia menetelmiä, tai näiden molempien yhdistelmiä. (Pohl & Rupp 2011, 33.)

Dokumentointi luonnollisella kielellä

Luonnolliseen kieleen perustuvat menetelmät ovat yleisimmin käytössä vaatimusten dokumentoinnissa ja niiden etu on siinä, että kaikkien sidosryhmien on helppo ymmärtää sanallista kuvausta, joka on tuotettu jollakin tutulla kielellä, kuten esimerkiksi suomenkielellä tai englanninkielellä. (Kotonya & Sommerville 2004, 19; Pohl & Rupp 2011, 34–35).

Pohl & Rupp (2011, 57) huomauttavat, että luonnollisella kielellä tuotettujen vaatimusten haittapuolena voidaan pitää sitä, että kyseessä olevaa tapaa ei ole formalisoitu, joten kirjoitetut vaatimukset ovat usein monitulkintaisia. Sen vuoksi luonnollisen kielen käyttäminen johtaa usein väärinymmärryksiin.

Kotonya & Sommerville (2004, 19) listaavat kolme yleistä ongelmaa, joita luonnollisella kielellä kirjoitettujen vaatimusten yhteydessä esiintyy ja joihin vaatimusmäärittelijöiden tulisi kiinnittää huomiota vaatimuksia määritellessään:

1. Vaatimukset kirjoitetaan käyttäen vaikeaselkoisia ehdollisia lauseita, jotka hämmentävät lukijaa: jos A silloin B, jos kuitenkin C silloin…
2. Terminologiaa käytetään huolimattomasti ja epäjohdonmukaisesti.
3. Vaatimusten kirjoittajat olettavat, että lukijalla on erityistä tietoa toimialasta tai järjestelmästä ja he jättävät vaatimusmäärittelystä oleellista tietoa pois.

Pohl & Rupp (2011, 50) esittävät puolestaan seuraavan listan, joka heidän näkemyksensä mukaan sisältää kaikista oleellisimmat ongelmat vaatimusmäärittelijöiden työn kannalta, joihin heidän tulisi kiinnittää huomiota vaatimuksia kirjoittaessaan: *nominalisaatio, substantiivit ilman viittausta, universaali kvantisointi, puutteellisesti määritellyt ehdot* ja *epätarkasti määritellyt prosessiverbit.*

Nominalisaatiossa prosessi muutetaan yksittäiseksi tapahtumaksi, jossa esimerkiksi tekijä tai muut tarkentavat tiedot eivät tule esille. Prosessisana *siirtää* kääntyy nominalisaatiossa sanaksi siirto, jossa verbi on korvattu nominilla. Muita tyypillisiä esimerkkejä nominalisaatiosta ovat termit syöte (input), varaus (booking) ja hyväksyminen (acceptance). (Pohl & Rupp 2011, 50.)

Esimerkki lauseesta, jossa on käytetty nominalisaatiota: ”Tietojärjestelmän kaatuessa täytyy tehdä uudelleenkäynnistys”. Termit *tietojärjestelmän kaatuessa* ja *uudelleenkäynnistys* kuvaavat molemmat prosessia, joka pitäisi analysoida täsmällisemmin. Kenen esimerkiksi täytyy tehdä tietojärjestelmän uudelleenkäynnistys ja mikä tietojärjestelmä on kyseessä? (Pohl & Rupp 2011, 50.)

Substantiivit ilman viittausta ovat epätäydellisesti määriteltyjä, joista puuttuu lähdeviite. Termejä, jotka sisältävät epätäydellisesti määriteltyjä substantiiveja ovat esimerkiksi käyttäjä, ohjain, järjestelmä, viesti, tiedot, toiminto. (Pohl & Rupp 2011, 51.)

Esimerkki lauseesta, jossa substantiivia on käytetty ilman viittausta: ”Tiedot näytetään käyttäjälle pääteohjelmalla.” Lauseesta ei selviä mistä tiedoista, käyttäjästä ja pääteohjelmasta tarkalleen ottaen on kysymys. Esimerkki lauseesta, jossa substantiiviin on liitetty viittaus: ”Järjestelmä näyttää laskutuksen tiedot rekisteröityneelle käyttäjälle pääteohjelmalla, johon hän on kirjautunut.” (Pohl & Rupp 2011, 51.)

Universaali kvantisointi täsmentää määriä tai taajuuksia. Vaarana on, että asioita yleistetään liikaa. Universaalia kvantisointia voidaan helposti identifioida seuraavien sanojen avulla: ei koskaan, aina, ei, ei kukaan/ei mikään, jokainen, kaikki, muutama, ei mitään. (Pohl & Rupp 2011, 51-52.)

Esimerkki lauseesta, jossa on käytetty universaalia kvantisointia ja asioita on siten yleistetty liikaa: ”Järjestelmä näyttää kaikki tietojoukot jokaisessa alavalikossa.” Lauseesta täytyy selvittää, tarkoitetaanko todellakin kaikkia tietojoukkoja ja kaikkia alavalikoita. (Pohl & Rupp 2011, 52.)

Puutteellisesti määritellyt ehdot on toinen mittari potentiaalisen tiedon katoamisesta. Vaatimukset, jotka sisältävät ehtoja määrittävät käyttäytymisen, joka täytyy esiintyä, kun ehto on saavutettu. (osa, joka usein puuttuu). Puutteellisesti määriteltyjä ehtoja voi tarkastella seuraavien sanojen avulla: jos…sitten, siinä tapauksessa että, -ko, -kö, riippuen jostakin. (Pohl & Rupp 2011, 52.)

Esimerkki lauseesta, jossa vaatimuksen ehdot on määritelty puutteellisesti: ”Ravintolan järjestelmän pitää tarjota kaikki vaihtoehdot rekisteröityneelle asiakkaalle, joka on yli 20-vuotias.” Lauseesta jää epäselväksi, mitkä vaihtoehdot järjestelmän täytyy tarjota asiakkaalle, joka on 20-vuotias tai nuorempi. Ensimmäinen esimerkki lauseesta, jossa vaatimuksen ehdot on määritelty riittävän tarkasti: ”Ravintolan järjestelmän pitää tarjota kaikki alkoholittomat vaihtoehdot kaikille rekisteröityneille asiakkaille, jotka ovat nuorempia kuin 21-vuotiaat.” Toinen esimerkki lauseesta, jossa vaatimuksen ehdot on määritelty riittävän tarkasti: ”Ravintolan järjestelmän pitää tarjota kaikki vaihtoehdot, mukaan lukien kaikki alkoholipitoiset vaihtoehdot, kaikille yli 20-vuotiaille käyttäjille.” (Pohl & Rupp 2011, 52.)

Epätarkasti määritellyt prosessiverbit. Jotkut prosessiverbit vaativat enemmän kuin yhden substantiivin, jotta ne on määritelty riittävän tarkasti. Esimerkiksi prosessiverbi *siirtää* (transmit) vaatii vähintään kolme täydennystä, jotta sitä voidaan tarkastella riittävän tarkasti: *mitä* siirretään, *mistä* siirretään, ja *minne* siirretään. (Pohl & Rupp 2011, 53.)

Lisäksi passiivimuodon sijaan kannattaa käyttää mieluummin aktiivimuotoa, joka on ymmärrettävämpi. Esimerkki lauseesta, jossa on käytetty passiivimuotoa: ”Käyttäjän kirjautuessa sisään kirjautumistiedot kirjataan.” Lauseesta jää epäselväksi kuka kirjautumistiedot kirjaa, sekä missä ja kuinka se tehdään. Sama lause esitettynä aktiivimuodossa: ”Järjes-

telmän täytyy sallia, että käyttäjä syöttää käyttäjätunnuksensa ja salasanansa käyttäen pääteohjelman näppäimistöä.” (Pohl & Rupp 2011, 53.)

Pohl & Rupp (2011, 54–57) suosittelevat vaatimusten kuvaamiseen luonnollisella kielellä seuraavaa viisivaiheista mallia, jota he pitävät hyvänä menetelmänä, jos halutaan saavuttaa laadukkaasti määritellyt vaatimukset: määrittele laillinen velvollisuus, määrittele vaatimuksen ydin, luonnehdi järjestelmän toiminta, lisää objektit ja määrittele loogiset ja väliaikaiset ehdot.

1. **vaihe: Määrittele laillinen velvollisuus.** Vaatimuksen laillinen velvollisuusaste voidaan määritellä kiireellisen vaatimuksen ja tulevaisuudessa olevan vaatimuksen välillä, käyttämällä vaatimusten kuvaamisessa modaalisia verbejä, kuten *pitää, pitäisi* ja *tulee*. (Pohl & Rupp 2011, 54.)

2. **vaihe: Vaatimuksen ydin.** Jokaisen vaatimuksen ydin on *toiminnallisuus*, jota vaatimus määrittelee ja tarkentaa (esimerkiksi tulostaa, tallentaa, liittää tai laskea). Toiminnallisuus viittaa prosesseihin, jotka ovat aktiviteetteja ja ne voidaan kuvata käyttämällä vain verbejä. Prosessi-sanat täytyy määritellä niin selkeästi kuin mahdollista ja niitä tulee käyttää niin johdonmukaisesti kuin mahdollista. (Pohl & Rupp 2011, 54.)

3. **vaihe: Luonnehdi järjestelmän toiminta**. Toiminnallisia vaatimuksia varten järjestelmän toiminnot voidaan luokitella kolmeen vaatimusmalliin:

1. vaatimusmalli: *Järjestelmän autonominen toiminto:* järjestelmä tekee toiminnon itsenäisesti.
2. vaatimusmalli:*Käyttäjän vuorovaikutus:* Järjestelmä tarjoaa prosessin käyttäjälle palveluna.
3. vaatimusmalli:*Käyttöliittymän vaatimus*: Järjestelmä tekee toiminnon, joka riippuu kolmannesta osapuolesta (esimerkiksi toisesta järjestelmästä). Järjestelmä on passiivinen ja odottaa ulkoista tapahtumaa. (Pohl & Rupp 2011, 54.)

Mikä tahansa järjestelmässä määritelty toiminto, dokumentoidaan käyttäen yhtä edellä mainitusta kolmesta vaatimusmallista. Sen jälkeen, kun kaikki kolme vaatimusmallia on määritelty, vaatimuksen rakenne on luotu (kuva 5). Esimerkkien hakasulkeisiin kirjoitetut sanat täytyy korvata vaatimuksen mukaisesti. (Pohl & Rupp 2011, 55.)

Kolmannen vaiheen 1. vaatimusmallia, *Järjestelmän autonominen toiminto*, käytetään kun vaatimukset luodaan. Se kuvaa järjestelmän toimintoja, jotka esitetään itsenäisesti, jolloin käyttäjä ei ole vuorovaikutuksessa toiminnon kanssa. (Pohl & Rupp 2011, 55.) Malliin määritellään seuraavat vaatimukset:

JÄRJESTELMÄN PITÄÄ/PITÄISI/TULEE/ <**prosessi verbi**>

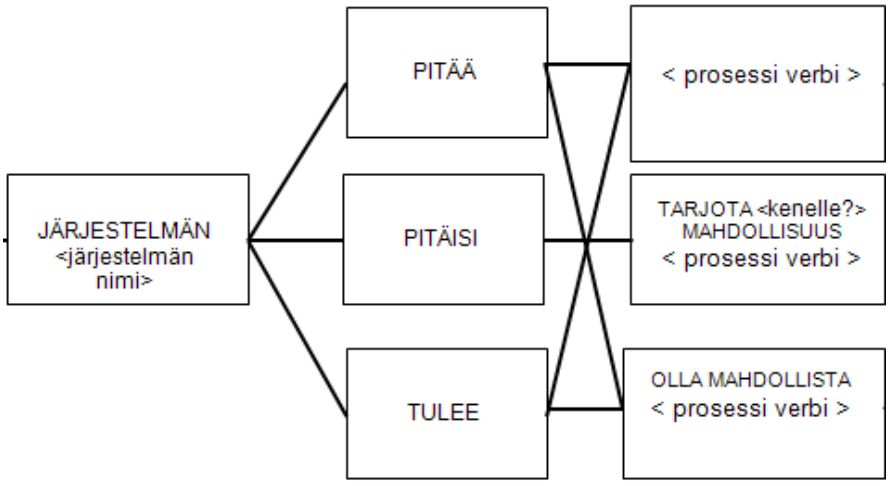
<Prosessi verbi> kuvaa toiminnallisuutta, joka on kuvattu 2. vaiheessa. Esimerkiksi *tulostaa* on prosessiverbi tulostustoiminnallisuudelle, joka esiintyy järjestelmässä. Jos järjestelmä tarjoaa toiminallisuuden käyttäjälle tai järjestelmä on suoraan vuorovaikutuksessa käyttäjän kanssa, vaatimukset luodaan käyttäen kolmannen vaiheen 2. vaatimusmallia (Pohl & Rupp 2011, 50). *Käyttäjän vuorovaikutus*:

JÄRJESTELMÄN PITÄÄ/PITÄISI/TULEE/**tarjota**<**kenelle?**>mahdollisuus<prosessi verbi>

Käyttäjä, joka on vuorovaikutuksessa järjestelmän kanssa liitetään järjestelmään vaatimuksen kautta <kenelle?>. (Pohl & Rupp 2011, 55.)

Jos järjestelmä suorittaa toiminnon, joka on riippuvainen toisista järjestelmistä, käytetään kolmannen vaiheen 3. vaatimusmallia: *Käyttöliittymän vaatimus*. Milloin tahansa viesti tai tieto toisista järjestelmistä saapuu, järjestelmän täytyy reagoida suorittamalla määritelty toiminto. (Pohl & Rupp 2011, 55-56.) Malliin määritellään seuraava vaatimus:

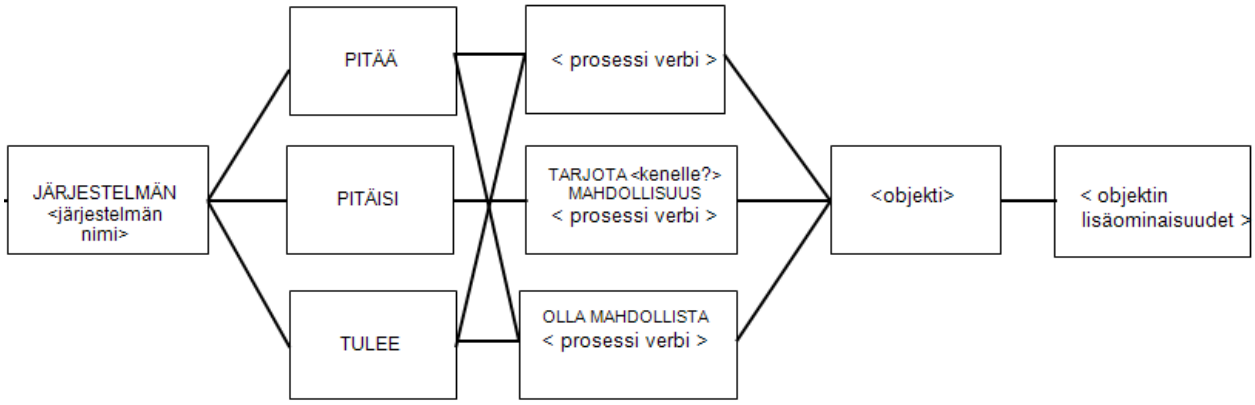
JÄRJESTELMÄN PITÄÄ/PITÄISI/TULEE/**olla mahdollista**<prosessi verbi>



Kuva 5. Vaatimuksen rakenne on luotu 1.-3. vaiheen mukaisesti (Pohl &Rupp 2011, 55).

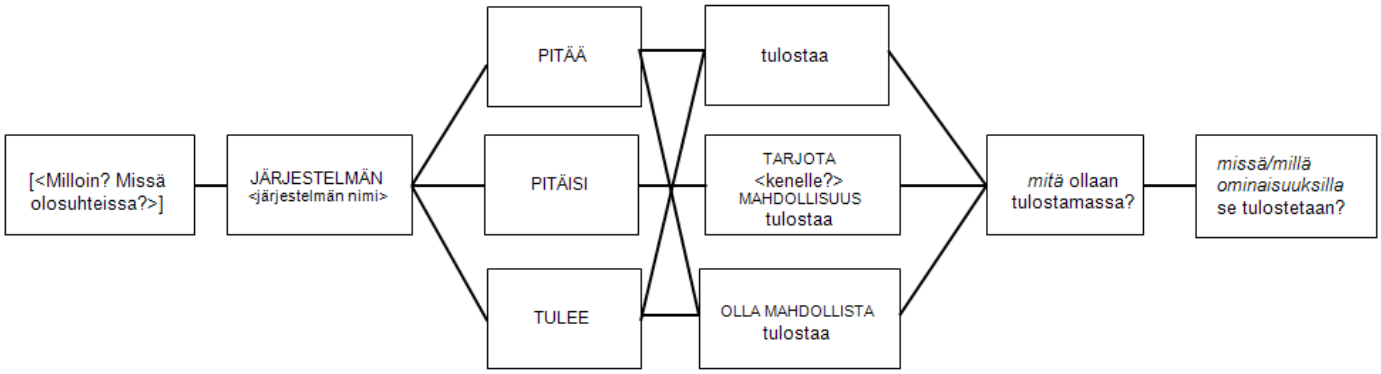
4. **vaihe: Lisää objektit.** Jotkut prosessiverbit vaativat yhden tai usemman lisäobjektin, jotta niitä voidaan tarkastella täydellisinä. Tässä vaiheessa, potentiaaliset puuttuvat objektit ja objektien täydennykset, adverbiaalit, tunnistetaan ja lisätään vaatimuksiin (kuva 6). Esimerkiksi prosessiverbille *tulostaa*, lisätään tieto *mitä* ollaan tulostamassa ja *missä* se tulostetaan: (Pohl & Rupp 2011, 56.)

JÄRJESTELMÄN PITÄÄ/PITÄISI/TULEE/olla mahdollista<tulostaa><**objekti**>*mitä*<***objektin lisäominaisuudet***> *missä/millä ominaisuuksilla* se tulostetaan



Kuva 6. Vaatimuksen rakenteeseen on lisätty objektit (Pohl & Rupp 2011, 56).

5. **vaihe: Määrittele loogiset ja väliaikaiset ehdot.** Tyypillisesti vaatimuksiin ei dokumentoida jatkuvia toiminnallisuuksia, vaan sellaiset vaatimukset, jotka esitetään tai tarjotaan vain loogisissa tai väliaikaisissa ehdoissa. Väliaikaiselle ehdolle valitaan niin pian kuin –konjuktio ja loogisille ehdoille valitaan jos -konjuktio. Sen sijaan kun -konjuktiota tulee välttää, koska se ei ilmaise selkeästi, onko kyseessä väliaikainen vai looginen ehto. Vaatimuksen alkuun lisätään sivulause (kuva 7), joka kuvaa laatuvaatimukset, eli ehdot, jotka vaatimuksen tulee täyttää. (Pohl & Rupp 2011, 56.)



Kuva 7. Luonnollisella kielellä toteutettu valmis vaatimusmalli, johon on lisätty myös ehdot. Esimerkissä prosessiverbin tilalla on käytetty tulostaa-verbiä. (Pohl & Rupp 2011, 57).

Edellä kuvatun mallin avulla on mahdollista kirjoittaa vaatimusten määrittely siten, että se olisi mahdollisimman ymmärrettävä ja helposti luettava. Pohl & Rupp (2011, 57) toteavat kuitenkin, että tätä vaatimusmallia ei kannata ottaa väkisin käyttöön, vaan on suositeltavaa tarjota vaatimusmäärittelijöille ensin koulutusta metodin käyttämiseen ja käyttää sitä sen jälkeen yhtenä vaatimusmäärittelyprosessia tukevana työkaluna.

Dokumentointi kaaviotekniikkaan perustuvilla menetelmillä

Kaaviotekniikkaa, erityisesti UML-notaatiota, käytetään usein luonnollisen kielen lisäksi vaatimusten dokumentoinnissa. Notaatio on graafinen kuvauskieli, jota UML-kaaviotekniikassa käytetään. UML-kaavioita on kahta päätyyppiä: rakennekaaviot ja käyttäytymiskaaviot. Käyttäytymiskaavioihin kuuluvat käyttötapauskaaviot (engl. use case diagram) ovat notaationsa osalta UML:n yksinkertaisin kaaviotekniikka. Kaaviotekniikan etuna pidetään sitä, että vaatimukset voidaan esittää tarkasti ja niiden avulla voidaan myös tehostaa kommunikointia projektin eri sidosryhmien välillä. (Fowler & Scott 2002, 5-7; Haikala & Mikkonen 2011, 73, 77.)

UML:n yksinkertaisimman kaaviotekniikan eli käyttötapausten avulla pystytään kuvaamaan havainnollisesti asiakasvaatimuksia ja niistä voidaan myös johtaa ohjelmistovaatimukset. Käyttötapaukset (engl. use case) voidaan kuvata esimerkiksi tekstuaalisina kuvauksina lomakepohjien avulla, kuvina tai kaavioina, joissa kuvataan mitä käyttäjä pystyy tekemään järjestelmällä. (Haikala & Mikkonen 2011, 82-83; Haikala & Märijärvi 2006, 95.)

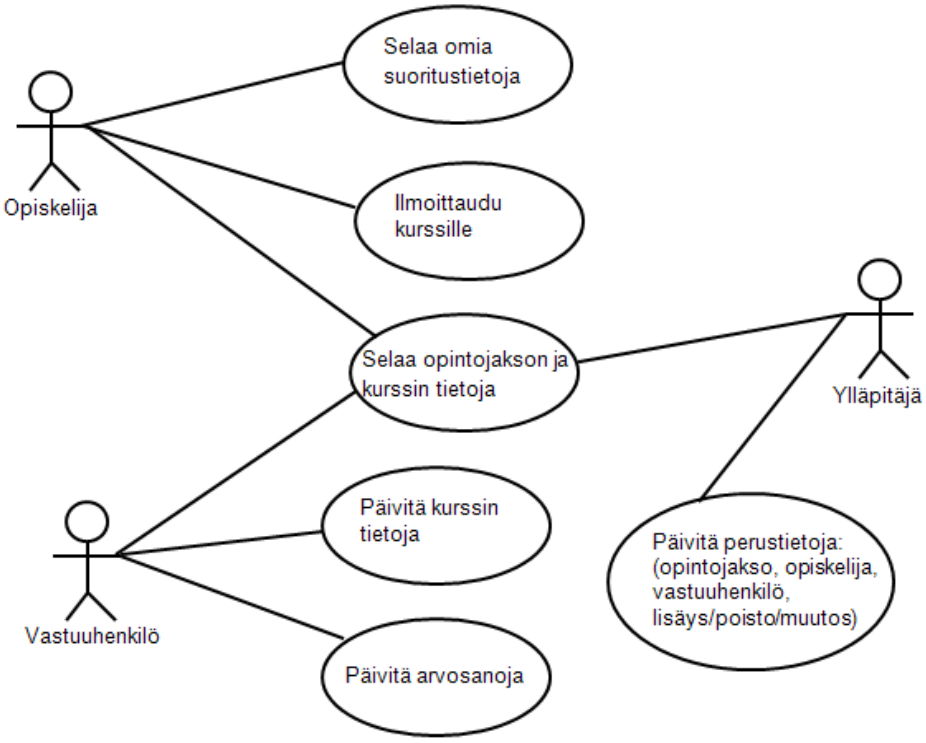
Seuraavaksi tarkastellaan lähemmin käyttötapauskaaviota, käyttötapauksen kuvaamista yksinkertaisimmalla tavalla, eli käyttötapauksen tekstuaalista kuvaamista lomakepohjan

avulla sekä käyttötapauksen kuvausta riippuvuuskaaviona. Ensimmäisenä tarkastellaan käyttötapauskaaviota.

Käyttötapauskaavio

Kuvan 8 käyttötapauskaavio sisältää UML-kaaviotekniikan kaikki tärkeimmät elementit: käyttötapaukset, aktorit ja niitä yhdistävät viivat. Kuvassa olevat soikiot kuvaavat käyttötapauksia. Aktorit edustavat järjestelmän käyttäjiä (opiskelija, vastuuhenkilö ja ylläpitäjä). Viiva aktorin ja käyttötapauksen välillä kertoo, että aktori osallistuu jotenkin kyseiseen käyttötapaukseen. (Haikala & Mikkonen 2011, 77-78.)

Opiskelija selailee tarjolla olevia kursseja ja omia suoritustietojaan. Löydettyään kiinnostavan kurssin hän yrittää ilmoittautua kurssille. Jos edellytykset ovat olemassa (esitietokurssit suoritettu) ilmoittautuminen onnistuu. Kurssilla on vastuuhenkilö, joka vastaa kurssin tietojen ajan tasalla pitämisestä, ja lisäksi koko järjestelmällä on ylläpitäjä, joka ylläpitää järjestelmän perustietoja. (Haikala & Mikkonen 2011, 78.)



Kuva 8. Esimerkki käyttötapauskaaviosta, joka kuvaa opiskelijan kurssille ilmoittautumiseen liittyviä käyttötapauksia (Haikala & Mikkonen 2011, 77).

Käyttötapauksen tekstuaalinen kuvaaminen lomakepohjan avulla

UML ei normita käyttötapauksen kuvausta millään tavalla. Eräs tyypillinen tekstuaalinen tapa kuvata käyttötapaus, on esitetty taulukossa 1. Käyttötapauksen tärkein osa on kuvaus, joka kertoo, mitä käyttötapauksessa pitäisi tapahtua. Ensin kuvataan käyttötapauksen peruskulku, eli miten se etenee normaalisti, ja vaihtoehtoiset kulut tai poikkeustilanteet merkitään tarinaan esimerkiksi hakasulkeisiin: [Esitietovaatimukset eivät täyty]. Poikkeusten käsittely voidaan selittää erikseen joko käyttötapauksessa tai siihen liittyvässä muussa tekstissä. UML ei tue ei-toiminnallisten vaatimusten dokumentointia, joten ne kannattaa kirjata esimerkiksi käyttötapauksen yhteyteen (muut vaatimukset). (Haikala & Mikkonen 2011, 79.)

Taulukko 1. *Ilmoittaudu kurssille* -käyttötapauksen tekstuaalinen kuvaus lomakepohjan avulla (Haikala & Mikkonen 2011, 80).

Nimi: Ilmoittaudu kurssille
Versiohistoria: versio 1.0/ijh
Osallistujat: Opiskelija
Esiehdot: Opiskelija, kurssi ja opintojakso on syötetty järjestelmään, kurssille ilmoittautuminen on avattu, opiskelija on kirjautuneena järjestelmään (KT:t Päivitä perustietoja, Päivitä kurssin tietoja)
Kuvaus: Opiskelija seuraa WWW-linkkiä, joka johtaa kurssin sivulle. Hän valitsee vaihtoehdon ilmoittaudu. Järjestelmä tarkastaa, että opiskelijalla on tarvittavat esitiedot [Poikkeus: esitietovaatimukset eivät täyty]. Järjestelmä varmistaa vielä OK/CANCEL-kyselyllä, että opiskelija todella haluaa ilmoittautua. Tämän jälkeen järjestelmä lisää opiskelijan kurssin osallistujaksi.
Poikkeukset: Esitietovaatimukset eivät täyty: opiskelijalle annetaan luettelo puuttuvista esitietokursseista.
Lopputulos: Opiskelija on rekisteröity kurssin osallistujaksi.
Muut vaatimukset: Päivittäin käsitellään kiireisimpänäkin tuntina enintään noin 50 varausta. Vastausajan on aina oltava alle 5 sekuntia.

Käyttötapausten kiistattomana etuna voidaan pitää sitä, että myös asiakas pystyy ymmärtämään käyttötapauksia, koska niiden kuvauskieli ja terminologia on yksinkertainen ja selkeä. Käyttötapausten lähtökohtana ovat aina käyttäjän tarpeet ja lisäarvon tuottaminen käyttäjälle, siten käyttötapauksessa onkin perusteellinen kuvaus siitä, kuinka käyttäjän kannalta tärkeä tarve saadaan toteutettua. Kuvausosuudessa kerrotaan miten alkutilan-

teesta (esiehdot) päästään ”suorittamalla” kuvaus selkeään lopputilanteeseen (lopputulos). Haikalan & Mikkosen mukaan (2011, 80) käyttötapausta kirjoitettaessa kannattaa kiinnittää huomiota seuraaviin asioihin:

1) Käyttötapauskaavio määrittää järjestelmän toiminnallisuuden korkealla abstraktiotasolla, eikä siihen yleensä kannata sisällyttää esimerkiksi käyttöliittymän yksityiskohtia.
2) Kirjoita yksinkertaisia, selkeitä ja lyhyitä lauseita, joilla vaiheistat käyttötapauksen tarinan.
3) Kaikkia käyttötilanteita ja asiakasvaatimuksia ei voi tai ei kannata esittää käyttötapauksina.
4) Huomioi, että sovellusalueella saattaa olla omia kuvauskäytäntöjä, eikä kuvaus ole aina tekstuaalinen.
5) Käyttötapauksesta tulee kuvata selkeästi, mikä laukaisee toiminnan käynnistymisen (engl. triggers) ja mikä on lopputilanne (engl. outcome), eli mihin päädytään.
6) Käyttötapaukseen sisältyy yleensä useampia kuin yksi ohjelmaan toteutettava toiminto.
7) Käyttötapauksen kuvaus kannattaa pitää tiiviinä, eikä sen pitäisi ylittää yhden A4-arkin pituutta. (Haikala & Mikkonen 2011, 80-81.)

Käyttötapauksen kuvaaminen yksinkertaisimmalla kaaviolla (kuva 8) toimii hyvänä apuvälineenä ohjaten keskustelua, kun kaavion ensimmäistä versiota piirretään fläppitaululle ja projektin eri osapuolet yrittävät rajata järjestelmää sen ympäristössään ja saada yhteistä näkemystä järjestelmästä. Kaavion ei ole tarkoitus esittää yksikäsitteistä ja täydellistä määrittelyä järjestelmästä, vaan tarjota kommunikointia helpottavat visuaaliset puitteet. (Haikala & Mikkonen 2011, 81).

Käyttötapauksen kuvaus riippuvuuskaaviona

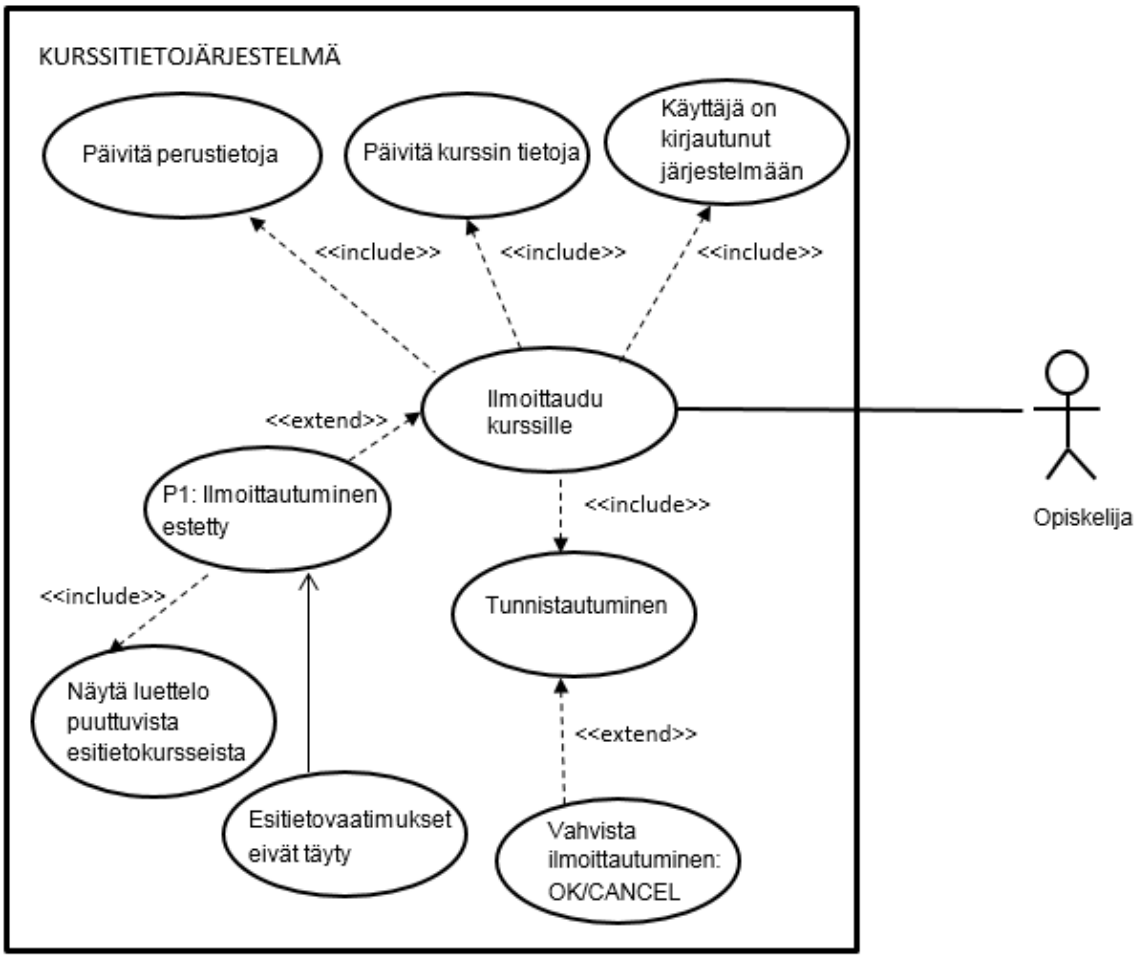
Riippuvuuskaaviota käytetään silloin, kun halutaan tarkentaa käyttötapausten välisiä suhteita. Kuvassa 9 on tyypillisiä käyttötapauskaaviossa käytettävissä olevia merkintöjä, kun käyttötapaus kuvataan riippuvuuskaaviona: aktorit, käyttötapausten väliset suhteet, käyttötapauksen perussisältöä laajentavat osat, yleistyssuhteet ja esiehdot. Järjestelmän käyttötapaukset voidaan piirtää laatikon sisälle, ja laatikkoon voidaan merkitä järjestelmän nimi. Kuvassa järjestelmän nimi on kurssitietojärjestelmä. (Haikala & Mikkonen 2011, 78.)

Kaikki aktorit eivät ole ihmisiä. Tällöin voidaan aktorin symbolina käyttää laatikkoa, jossa on stereotyyppi <<actor>> ja aktorin nimi. (Stereotyyppi-actoria ei löydy kuvasta). Käyttötapausten välisiä suhteita ilmaistaan termeillä sisällyttää (engl. include), laajentaa (engl. extend) tai yleistys (engl. generalization). Käyttötapaus voi sisällyttää itseensä toisen käyttötapauksen. Kuvassa esimerkiksi *Tunnistautuminen*-käyttötapaus sisältää *Vahvista ilmoittautuminen* –käyttötapauksen. (Haikala & Mikkonen 2011, 78; Helsingin yliopiston www-sivut 2009.)

Käyttötapauksen perussisältöä laajentavat osat voidaan liittää yleiseen käyttötapaukseen esimerkiksi sellaisissa tapauksissa, että käyttötapauksella on vaihtoehtoisia tai valinnaisia käyttötapauksia, tai vastaavasti virhe- ja poikkeustilanteissa. Kuvassa *Ilmoittaudu kurssille* - käyttötapaus voi esimerkiksi suorittaa poikkeustilanteessa *Ilmoittautuminen estetty* -käyttötapauksen, mikäli käyttäjän esitietovaatimukset eivät täyty. (Haikala & Mikkonen 2011, 78; Helsingin yliopiston www-sivut 2009.)

Yleistyssuhde on joko käyttäjien tai käyttötapausten välinen tai se voi olla erikoistapaus, joka tarkentaa yleisempää tapausta tai lisää siihen ominaisuuksia. (Haikala & Mikkonen 2011, 78; Helsingin yliopiston www-sivut 2009). Myöskään yleistyssuhdetta ei löydy kuvasta.

Kuvaan on merkitty *Ilmoittaudu kurssille* -käyttötapauksen esiehdot omina käyttötapauksinaan: Käyttäjä on kirjautunut järjestelmään, Päivitä kurssin tietoja ja Päivitä perustietoja. Näiden esiehtoihin liittyvien käyttötapausten kirjaamista suositellaan kuitenkin vain tekstuaaliseen kuvaukseen kyseisen käyttötapauksen yhteyteen. (Haikala & Mikkonen 2011, 78.)



Kuva 9. *Ilmoittaudu kurssille* -käyttötapaus riippuvuuskaaviona. Riippuvuuskaavio on mukailtu edellä kuvatun taulukon (taulukko 1) *Ilmoittaudu kurssille* -käyttötapauksen tekstuaalisesta kuvauksesta (Haikala & Mikkonen 2011, 80).

Sen jälkeen kun vaatimukset on kuvattu luonnollisen kielen lisäksi UML-notaatiota apuna käyttäen, on vuorossa vaatimusmäärittelyprosessin viimeinen vaihe. Vaihe on nimeltään vaatimusten validointi eli kelpoistaminen (Haikala & Mikkonen 2011, 67; Kotonya & Sommerville 2004, 87).

2.6 Vaatimusten validointi

Vaatimusten validointivaiheella pyritään osoittamaan, että lopullinen vaatimusmäärittelydokumentin luonnos on johdonmukainen ja tarkka, siinä on yrityksen standardin mukainen rakenne, vaatimukset ovat ymmärrettäviä, ristiriidattomia ja virheettömiä. Vasta sen jälkeen vaatimusdokumentti julkaistaan ja sitä käytetään järjestelmän kehittämisen pohjana. Vaatimusten validointivaihe etsii pääasiallisesti vastausta kysymykseen: ”Onko vaatimukset kirjattu oikein?” (Kotonya & Sommerville 2004, 88-89.)

Validointivaihe tehdään yleensä yhdessä sidosryhmien kanssa, esimerkiksi katselmoimalla vaatimusmäärittelydokumentti. Katselmointi voidaan suorittaa myös esikatselmoimalla dokumentti yhden henkilön toimesta, joka tarkastaa dokumentin selvät ongelmakohdat, ennen varsinaista katselmointia sidosryhmien kanssa. Siten säästetään katselmointiprosessiin kuluvaa aikaa ja kustannuksia. Katselmointiin voidaan käyttää erilaisia tarkistuslistoja, jotka helpottavat virheiden, puutteiden ja ristiriitojen löytymistä. Validointia voidaan tehdä myös prosessin aikaisemmissa vaiheissa prototyypin avulla, jos prototyyppi on kehitetty vaatimusten kartoittamisvaiheessa. Lisäksi vaatimusdokumentin erilaiset järjestelmän mallit, kuten esimerkiksi käsitemallit, luokkamallit, sidosryhmäkaaviot ja prosessikuvaukset tarkastetaan, että ne ovat ristiriidattomia, yhdenmukaisia ja kuvaavat sidosryhmien vaatimuksia. Tämän vaiheen tuotoksena syntyy yleensä lista vaatimuksissa ilmenneistä ongelmista ja yhteisesti sovitut toimenpiteet näiden ongelmien ratkaisemiseksi. (Haikala & Mikkonen 2011, 67; Haikala & Märijärvi 2006, 98; Kotonya & Sommerville 2004, 87-96, 100-104.)

2.7 Vaatimustenhallinta

Vaatimustenhallinta on keskeinen prosessi, jolla hallitaan suurta määrää tietoa ja varmistetaan, että tieto jaetaan oikeille ihmisille oikeaan aikaan. Vaatimustenhallinnalla tarkoitetaan pääasiallisesti sitä, että sen avulla hallitaan jo määriteltyihin vaatimuksiin tulevia muutoksia, vaatimusten välisiä suhteita ja vaatimusmäärittelydokumentin ja muiden dokumenttien välisiä riippuvuuksia. Vaatimustenhallintaa tehdään koko kehittämisprojektin ajan muiden kehittämistehtävien kanssa rinnakkain. Siten mitkä tahansa muutokset jo hyväksyttyihin vaatimuksiin voidaan katsoa olevan osa vaatimustenhallinnan prosessia, jossa vaatimusten muuttumista hallitaan. (Kotonya & Sommerville 2004, 113-115.)

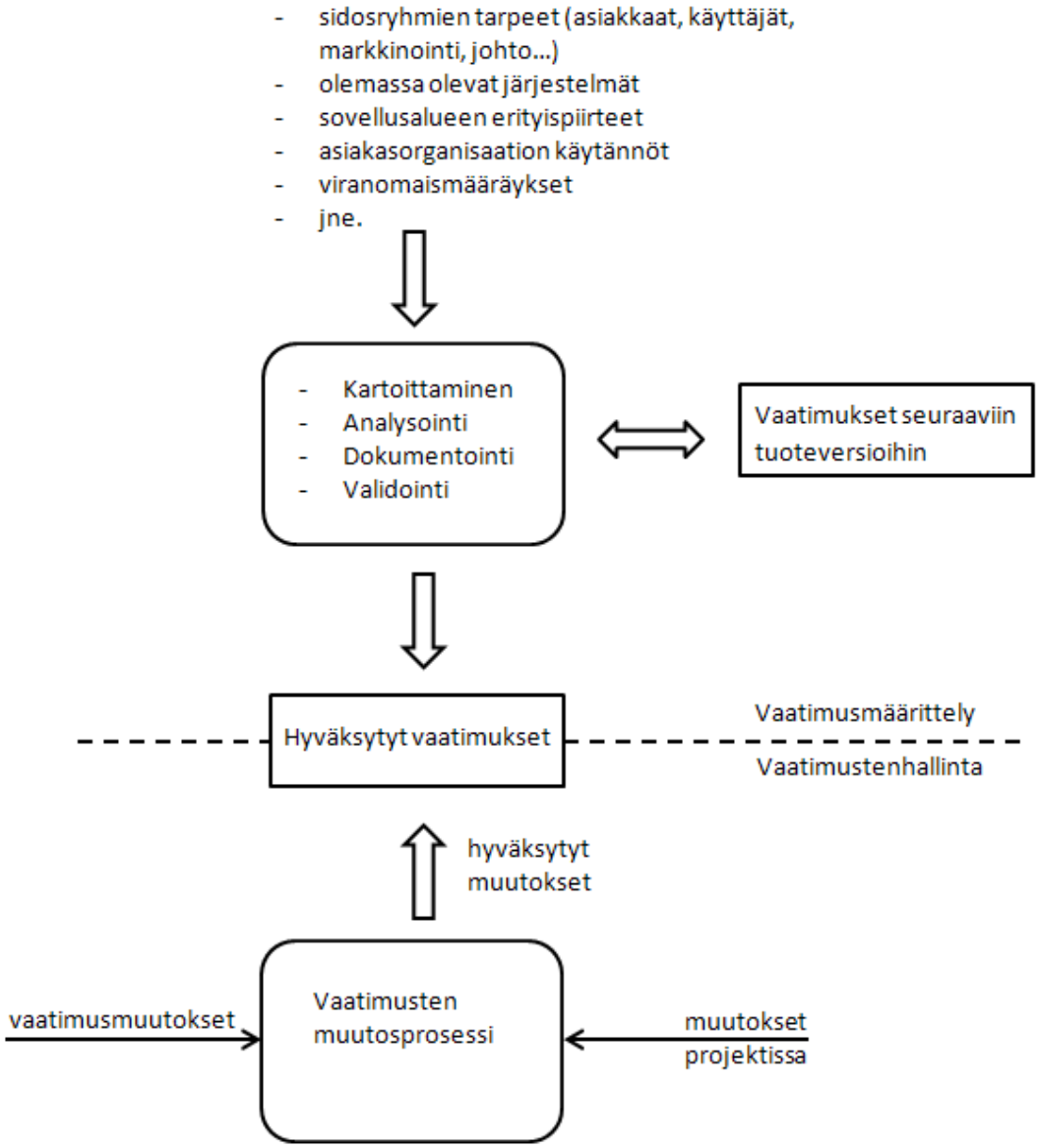
Ongelmat vaatimustenhallinnassa tarkoittavat usein sitä, että järjestelmä, jonka vaatimukset eivät tyydytä asiakasta, toteutetaan tai järjestelmän kehityksen aikataulu saattaa pitkittyä ja uudelleen tehtävä suunnittelu ja toteutus, jotka sisältävät vaatimuksiin tulevat muutokset, aiheuttavat korkeita kustannuksia. Lisäksi matalalla prioriteetilla olevat, vähemmän tarpeelliset, muutokset saatetaan toteuttaa järjestelmään ennen korkealla prioriteetilla olevia muutoksia, jos vaatimustenhallinnassa on ongelmia. (Kotonya & Sommerville 2004, 114-115.)

Vaatimuksia ei voi hallita tehokkaasti ilman vaatimusten jäljitettävyyttä. Vaatimusta voidaan sanoa jäljitettäväksi, jos voidaan osoittaa keneltä vaatimus on peräisin, miksi vaati-

mus on esitetty, mihin muihin vaatimuksiin kyseessä oleva vaatimus liittyy, kuinka tämä vaatimus liittyy muuhun informaatioon, kuten järjestelmän suunnitteluun, toteutukseen ja dokumentointiin. Keskeinen vaatimustenhallinnan esiehto on, että jokaisella vaatimuksella täytyy olla jonkinlainen yksilöllinen tunniste. Vaatimusten jäljittämistä ja hallintaa varten voidaan luoda esimerkiksi tietokanta, johon vaatimukset tallennetaan. (Kotonya & Sommerville 2004, 114-115, 117.)

Vaatimuksiin tulee muutoksia esimerkiksi siitä syystä, että projektin aikana ohjelmiston toimintaympäristössä voi tapahtua erilaisia muutoksia, projektin alussa kaikkia asiakkaiden vaatimuksia ei välttämättä ymmärretä oikein, asiakkaiden ymmärrys heidän todellisista vaatimuksista projektin aikana lisääntyy, jonkin asiakasvaatimuksen täyttämiseksi määritelty ominaisuus havaitaan virheelliseksi, asiakasvaatimusten priorisoinnissa tapahtuu muutos, asiakasvaatimuksen toteuttaminen osoittautuu liian kalliiksi tai aikataulupaineista johtuen jokin ominaisuus jätetään toteuttamatta. (Haikala & Märijärvi, 2006, 94-99; Kotonya & Sommerville 2004, 116.)

Haikalan & Mikkosen (2011, 67) mukaan vaatimusten muutosten hallinta on olennaisin prosessi vaatimustenhallinnassa, koska tämä on työtä, joka laskutetaan asiakasprojekteissa erikseen. Vaatimusten muutoksia varten projektissa täytyy olla sovittuna selkeät menettelyt. Täytyy esimerkiksi sopia, miten muutokset hyväksytään asiakasvaatimuksiin. Muutosprosessin vaiheisiin kuuluu muun muassa muutospyynnön tekeminen, analysointi, testaus ja hyväksyminen. Muutosprosessissa vaatimuksen käsittely etenee saman käytännön mukaan kuin vaatimusmäärittelyssäkin. Muutospyynnön hyväksyntä tapahtuu esimerkiksi projektin ohjausryhmässä. Kuvassa 10 on kuvattu vaatimustenhallintaan liittyvät asiat katkoviivan alapuolella.



Kuva 10. Vaatimustenkäsittelyprosessi Leffingwellin mukaan [2007]. (Haikala & Mikkonen 2011, 66).

Pienissä projekteissa muutostenhallintaprosessi hoidetaan usein erittäin kevyesti. Muodollisempaan muutostenhallintaan siirrytään vasta ylläpitovaiheessa, kun projekti on päättynyt. (Haikala & Mikkonen 2011, 67.)

Muutosten suurin ongelma on se, että tehdyt muutokset vaikuttavat yleensä muihinkin vaiheisiin, joten muutokset aiheuttavat lisätyötä sitä enemmän, mitä myöhäisemmässä vaiheessa ne joudutaan tekemään. Sen vuoksi on selvitettävä minne muutokset vaikuttavat ja millä tavalla ne vaikuttavat, koska ohjelmistoprojektin suurimpia riskejä ovat muuttuvat, virheelliset ja puutteelliset asiakasvaatimukset. (Haikala & Märijärvi 2006, 94; Kotonya & Sommerville 2004, 113-115).

3 Tietojärjestelmän kehittäminen ja vaatimusten määrittely: Case Yritys X

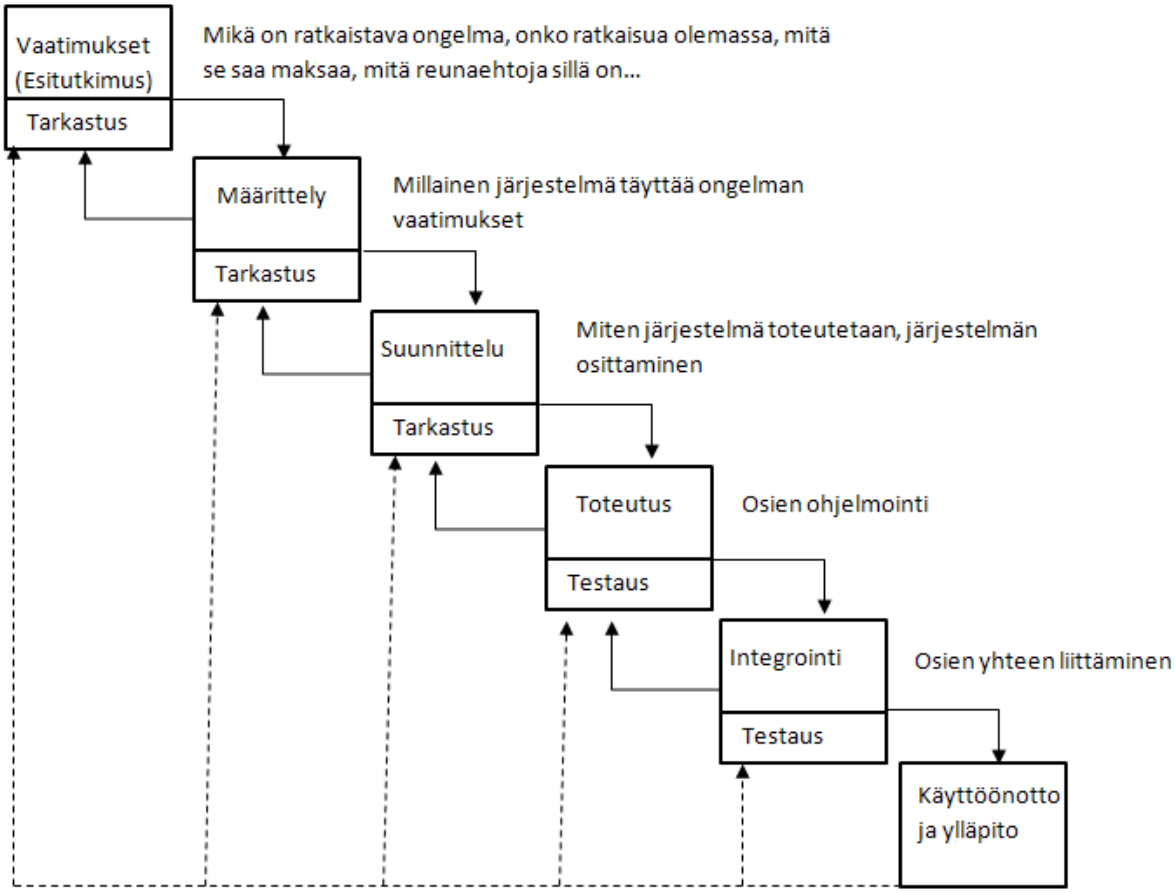
Tietojärjestelmän kehittäminen on systemaattista toimintaa, joka saa yleensä alkunsa tarpeesta kehittää uusi järjestelmä tai tarpeesta ylläpitää vanhaa järjestelmää. Tietojärjestelmän kehittäminen muodostuu joukosta ajallisesti toisiaan seuraavia vaiheita, joissa suoritetaan kehittämiseen liittyviä tehtäviä. Näitä tietojärjestelmän kehitykseen liittyviä peräkkäisiä vaiheita nimitetään tietojärjestelmän elinkaareksi. Vaihejaon tärkein tehtävä on määrittää järjestelmän kehittämiseen vaadittavat tehtävät, niiden ajoitus ja riippuvuudet toisistaan. (Pohjonen 2002, 26.)

Tietojärjestelmän elinkaaresta eli ohjelmistoprosessista on tehty malleja, jotka kuvaavat tärkeimmät ohjelmistoprosessin vaiheet ja niiden suoritusjärjestyksen. Ohjelmistoprosessimalleja on useita erilaisia, joissa prosessin vaiheiden suoritusjärjestys ja vaiheiden määrä, sekä vaiheiden nimeämiskäytäntö vaihtelee. Tunnetuin elinkaarimalli on vesiputousmalli (engl. waterfall model) (Royce 1970). Muita malleja ovat esimerkiksi prototyyppilähestymistapa (Floyd 1984), inkrementaalinen malli (Pressman1997), spiraalimalli (Boehm 1988), iteratiiviset mallit ja ketterät mallit (Beck1999; Poppendieck ja Poppendieck 2006; Takeuchi ja Nonaka 1986). (Haikala & Mikkonen 2011, 37-56.) Mikäli tietojärjestelmän kehittämiseen ei käytetä mitään edellä mainittuja valmiita malleja, puhutaan niin sanotusta ad hoc –menetelmästä, jossa ongelman ratkaisuksi kelpaa mikä tahansa toimiva tapa. (Haikala & Märijärvi 2006, 27).

Vesiputousmalli

Tässä työssä tarkastellaan tietojärjestelmän kehittämisen vaiheita iteroivan vesiputousmallin mukaisesti (kuva 11), koska työn empiriaosuudessa kuvattu kehitystyö toteutettiin kyseisen mallin mukaisesti.

Vesiputousmalli on yksi tunnetuimpia ohjelmistokehitysmalleja. Winston W. Royce esittti vuonna 1970 artikkelissaan *”Managing the Development of Large Software Systems”* vesiputousmallin ensimmäisen virallisen määritelmän. Royce pitää iterointia taaksepäin vesiputousmallin tärkeimpänä ominaisuutena (Haikala & Mikkonen 2011, 37).



Kuva 11. Esimerkki vesiputousmallista (Haikala & Märijärvi 2006, 36).

Haikalan & Märijärven (2006, 37) mukaan iteroivan vesiputousmallin ohjelmiston kehitysprosessi alkaa esitutkimusvaiheesta (kuva 11), jonka päämääränä on selvittää syitä siihen, miksi uusi järjestelmä tulisi rakentaa, mitä vaatimuksia järjestelmälle on asetettu, mitkä ovat järjestelmän viiteryhmät ja mitä eri vaihtoehtoja toteuttamisen ratkaisemiseksi on olemassa.

Esitutkimusta seuraa määrittelyvaihe, jossa kartoitetaan järjestelmälle vaadittavat asiakasvaatimukset ja määritellään toteutettava järjestelmä. Vaiheen tuloksena syntyy toiminnallinen määrittely –dokumentti, joka sisältää seuraavan vaiheen eli suunnitteluvaiheen vaatimukset. (Haikala & Märijärvi 2006, 38-39).

Suunnitteluvaihe jaetaan usein kahteen tasoon: arkkitehtuurisuunnitteluun ja moduulisuunnitteluun. Moduulisuunnitteluvaiheessa suunnitellaan jokaisen moduulin sisäinen rakenne. Moduuli on kokonaisuus, jonka avulla koteloidaan kaikki järjestelmän osakokonaisuuteen liittyvät toiminnot. Lisäksi moduulilla on määritelty rajapinta, jonka kautta ta-

pahtuu moduulin kommunikointi ympäristön kanssa. Suunnittelun tarkoituksena on muuntaa asiakkaan tarpeiden mukaan tehty määrittely tekniseksi määrittelyksi, jossa kuvataan miten järjestelmä toteutetaan. Suunnittelun tavoitteena on myös saada järjestelmästä mahdollisimman selkeä, ymmärrettävä, tehokas, luotettava, ylläpidettävä ja helposti siirrettävä. Näiden tavoitteiden kannalta avainasemassa on ohjelmiston onnistunut osittaminen moduuleiksi. (Haikala & Märijärvi 2006, 81-82).

Suunnittelua seuraa varsinainen toteutusvaihe, jossa ohjelmoidaan järjestelmään tulevat eri osat. Toteutuksen jälkeen on vuorossa integrointivaihe, jonka jälkeen kehitysprosessi päättyy ohjelmiston käyttöönotto- ja ylläpitovaiheeseen. Jokaiseen vaiheeseen liittyy tarkastuksia, katselmuksia tai testausta, joiden avulla pyritään varmistamaan toteutettavan järjestelmän laatu jo mahdollisimman varhaisessa vaiheessa. Testauksen tavoitteena on löytää ohjelmistosta virheitä. Sen vuoksi testauksen kannalta spesifikaatio eli määrittelydokumentti on aivan oleellinen, koska testauksen yhteydessä virhe on yhtä kuin poikkeama spesifikaatiosta. Siten lopputuloksen oikeellisuutta ei voida todeta, jos spesifikaatiota ei ole käytettävissä. Toiminnallinen määrittelydokumentti ja tekninen määrittelydokumentti ovat yleisimpiä testauksessa käytettäviä spesifikaatioita. (Haikala & Märijärvi 2006, 37-41, 78, 287; Pohjonen 2002, 27, 32.)

Vesiputousmallia kohtaan on esitetty alan kirjallisuudessa paljon kritiikkiä. Erityisesti on kritisoitu lineaarista vesiputousmallia, jossa tietojärjestelmän kehittämisen vaiheet seuraavat suoraviivaisesti toisiaan, koska käytännössä järjestelmän kehittäminen on paljon monimutkaisempaa, kuin lineaarisessa vesiputousmallissa on annettu ymmärtää. Mallista on kritisoitu myös sitä, että kehittämisen vaiheissa taaksepäin peruuttaminen on hankalaa ja kallista, koska peruuttaminen edellyttää yleensä sitä, että kaikki edeltävät vaiheet joudutaan uusimaan. Malli kuvaa myös huonosti ohjelmistoprosessille tyypillisen iteratiivisuuden, jossa toteutettavat vaatimukset jaetaan tuotekehityssykleihin, ja jokaisen syklin lopussa demonstroidaan siihen asti toteutettuja vaatimuksia. Lisäksi mallin ongelmana pidetään sitä, että riskien hallinta on monimutkaista ja kehitystyön tuloksia pystytään esittämään asiakkaalle vasta prosessin loppuvaiheessa. Siitä huolimatta, että vesiputousmalli on vanhanaikainen ja sen käytössä on selviä ongelmia, se on tunnetuin prosessimalli ja sitä käytetään edelleen yleisesti. Erityisesti, jos sallitaan iterointi ja vaiheiden käynnistäminen jo ennen edellisen vaiheen päättymistä, saadaan vesiputousmallista moneen tilanteeseen sopiva toimintamalli, kuten kuvassa 11 on esitetty. (Haikala & Mikkonen 2011, 37, 40-43; Kotonya & Sommerville 2004, 33-34; Pohjonen 2002, 40.)

Työn empiriaosuudessa kuvataan Yritys X:n ylläpitovaiheessa olevan tietojärjestelmän pienkehitystyö, joka toteutettiin vuonna 2014. Pienkehitystyövaiheessa käytettiin ohjelmistoprosessimallina pääosin Winston W. Roycen esittämää iteroivaa vesiputousmallia ja osittain käytettiin myös prototyyppilähestymistapaa.

Tässä työssä prosessimallin vaiheista käydään läpi esitutkimus ja vaatimusten määrittely sekä testaus. Esitutkimusvaiheessa ja määrittelyvaiheessa kuvataan, kuinka toiminnallisten vaatimusten määrittelyprosessi ja siitä syntyvän vaatimusmäärittelydokumentin tuottaminen toteutui käytännössä tämän kehitysprojektin kohdalla. Testausvaiheessa käydään läpi kehitysprojektissa tuotettujen vaatimusten testausta.

Opinnäytetyön kirjoittaja oli kyseisessä kehitysprojektissa yhtenä projektiryhmän jäsenenä määrittelemässä järjestelmän toiminnallisia vaatimuksia, sekä testaamassa määriteltyjä vaatimuksia järjestelmän pääkäyttäjän näkökulmasta. Tarkoituksena on peilata teorian ja käytännön eroavaisuuksia ja tarkastella, kuinka hyvin teorian opit toteutuivat käytännön työssä. Empiirisen osuuden aineistona käytetään vuonna 2014 toteutetun Yritys X:n tietojärjestelmän pienkehitystyöprojektin aikana tuotettuja dokumentteja ja lisäksi kehitystyön vaiheiden kuvaamisessa hyödynnetään opinnäytetyön kirjoittajan kokemusnäkökulmaa kyseisen pienkehitystyöprojektin osalta.

Lähtökohta tietojärjestelmän kehittämiselle tuli liiketoiminnan tavoitteesta saada yhtenäistettyä yhtiön kuvatyönkulkuprosessit sen jälkeen, kun yrityksen kuvatoimisto, Yritys Y, fuusioitiin Yritys X:ään. Kuvatyönkulkuprojektin tavoite sovellusten osalta oli arvioida Yritys Y:ssä käytössä olleen digitaalisen sisällönhallintajärjestelmän, FotoWaren tuoteperheen, käyttökelpoisuutta verrattuna Yritys X:n käytössä olevan digitaalisen Mediaarkistojärjestelmän, käyttökelpoisuuteen ja valita vertailun jälkeen toinen kyseessä olevista sovelluksista, johon tulevat muutokset toteutettaisiin.

Yritys X:lle oli tarkoitus luoda uusi sisäinen kuvapankki vanhan julkisen kuvapankin tilalle kuvien pitkäaikaista säilytystä varten ja samalla oli tarkoitus yhtenäistää kuvatyönkulkuprosessiin liittyviä toimintoja, kuten kuvien hankintaprosessia, digitaalisen sisällön hallintaa ja kuvien arkistointiprosessia. Kaikki nämä edellä mainitut ominaisuudet ja toiminnallisuudet tuli olla samassa sovelluksessa.

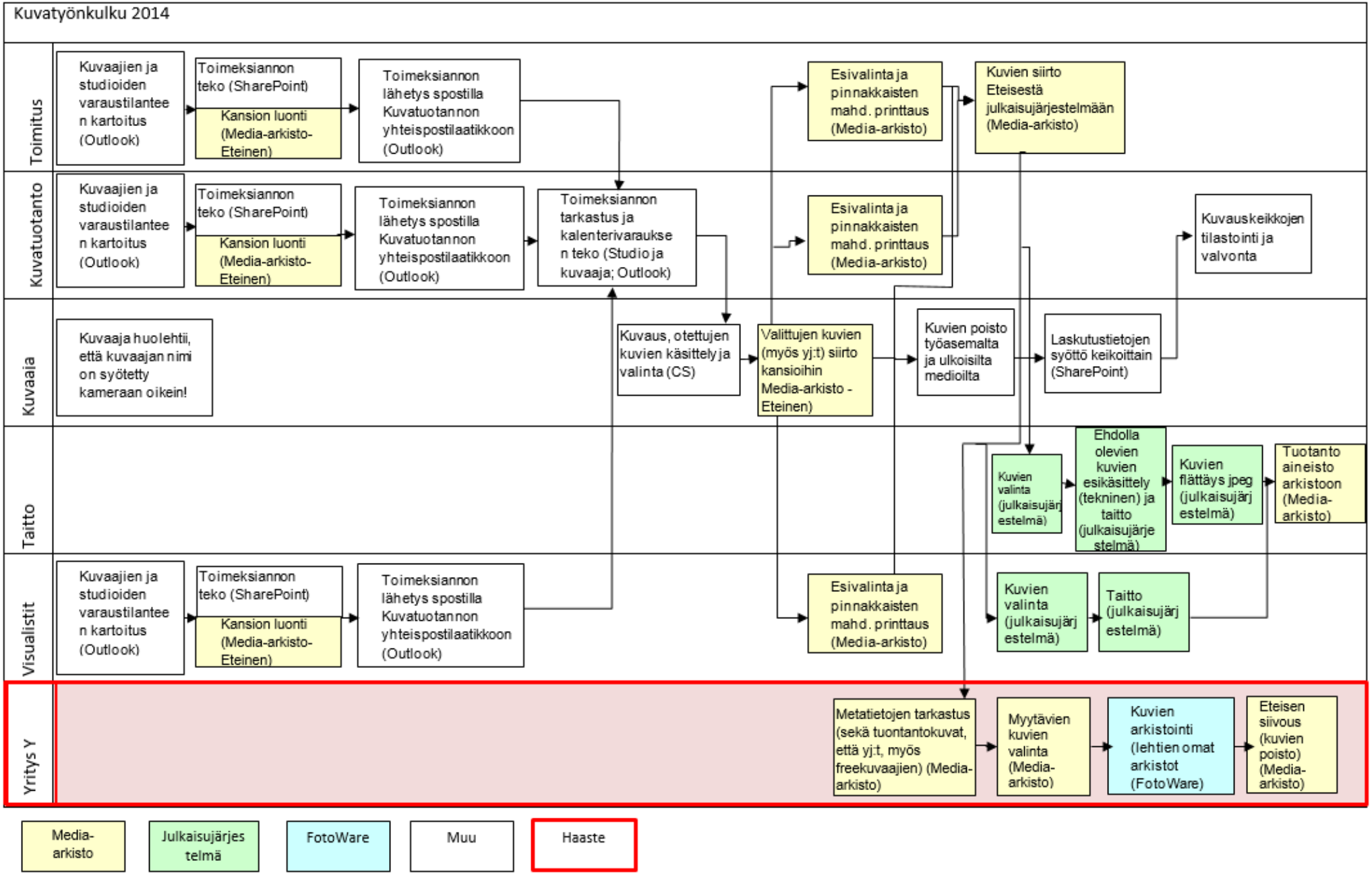
Kehittämishankkeen pienuuden vuoksi hanke toteutettiin esitutkimuksen jälkeen yhtenä projektina, jota projektipäällikkö ohjasi jatkuvasti projektin aikana. Projektin läpivienti koos-

tui projektille ominaisista neljästä eri vaiheesta: projektin suunnittelusta, käynnistämisestä, toteutuksesta ja päättämisestä.

Projektiorganisaatio muodostui projektiryhmästä ja ohjausryhmästä. Projektiryhmään kuuluivat sekä projektipäällikö että projektin jäsenet. Projektin jäsenet koostuivat sovellusalueen asiantuntijasta, sovelluksen pääkäyttäjästä ja kehittäjästä, joka oli samalla sovellusasiantuntija ja sovellustoimittaja. Projektin vastuuhenkilönä toimi projektipäällikkö. Projektin korkein päättävä elin oli johtoryhmä, joka koostui projektin asiakasorganisaation edustajista.

Projektin alussa projektipäällikkö laati projektisuunnitelman, jossa kuvattiin miten projektille asetetut tavoitteet oli tarkoitus saavuttaa. Projektin suunnitteluvaiheessa tutkittiin eri ratkaisujen ajalliset ja taloudelliset tulokset ja sen perusteella valittiin paras toteutustapa. Suunnittelun yhteydessä kartoitettiin myös potentiaalisia ongelmia. Johtoryhmän hyväksymä projektisuunnitelma oli projektiryhmän toimeksianto, joka kuvasi myös valtuudet.

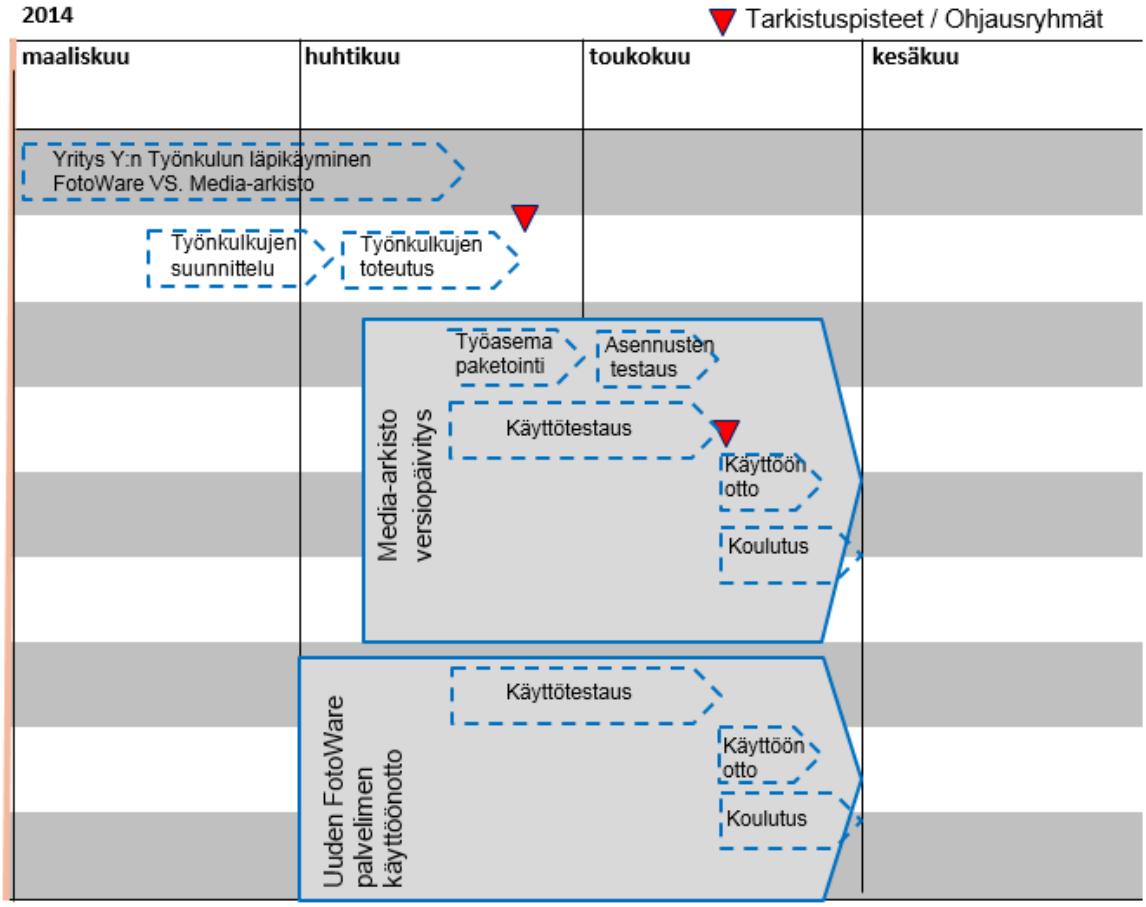
Projektin ensimmäisessä vaiheessa, eli suunnitteluvaiheessa, käytiin läpi kuvatyönkulun liiketoimintaprosessikaavio (kuva 12), joka oli kuvattu uimaratakaaviona. Kaaviosta käy ilmi työnkulkujen vaiheet, kuka tekee, mitä tekee ja missä vaiheessa tekee kulloisenkin prosessissa vaaditun tehtävän. Liiketoimintaprosessikaavion avulla tehtiin kohdealueen rajaus, joka on esitetty kuvassa punaisella rajauksella. Kaavion avulla selvitettiin myös kuvatyönkulkuprosessin nykytila sekä samalla suunniteltiin prosessin tavoitetilaa ja pohdittiin tulevia muutoksia ja kehitystarpeita.



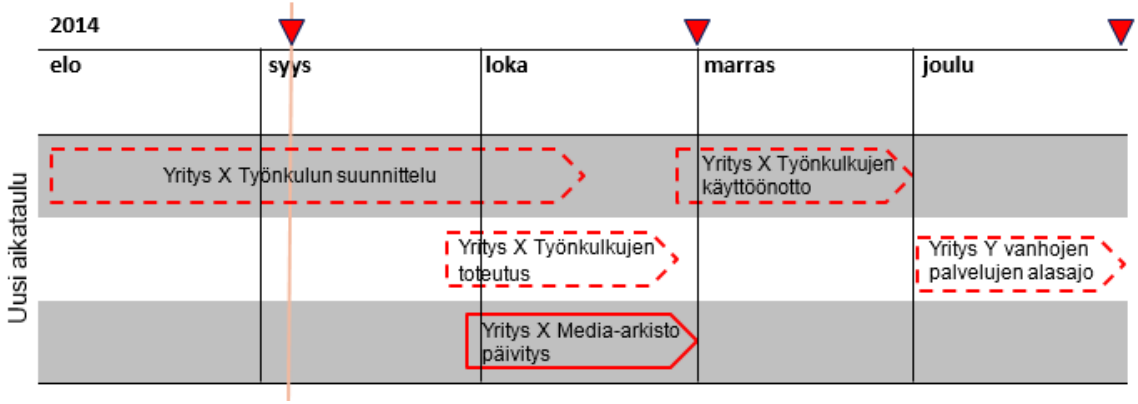
Kuva 12. Kuvatyönkulun nykytilan prosessikaavio.

Projektin toimituksen aikatauluksi suunniteltiin ensivaiheessa kolme kuukautta (kuva 13). Projekti käynnistyi maaliskuussa 2014 Yritys Y:n työnkulkujen läpikäynnillä, jonka oli arvioitu kestävän huhtikuun alkupuolelle. Seuraavaksi oli tarkoituksena käynnistyä työnkulkujen toteutus, jonka oli määrä jatkua yhtä aikaa työnkulkujen läpikäynnin kanssa. Huhtikuun puolivälistä oli arvioitu alkavan työasemapaketointi ja samalla oli arvioitu alkavaksi uuden Media-arkiston -versiopäivityksen käyttötestaus, jonka piti kestää toukokuun puoliväliin. Työasema-asennusten testauksen oli määrä alkaa toukokuun alussa ja kestoksi arvioitiin pari viikkoa. Käyttöönoton ja koulutuksen suunniteltiin alkavan toukokuun puolivälissä ja jatkuvan toukokuun loppuun. Alkuvaiheen suunnitelma oli, että kesäkuun alkuun mennessä projektin pitäisi olla valmis.

Projektin toimituksen aikataulu muuttui kuitenkin suunnitellusta, koska samaan aikaan tämän projektin kanssa kaikki tietohallinnon resurssit olivat kiinni useamman viikon ajan tuotantoympäristön päivitysprojektissa. Tästä johtuen testiympäristön luonti ja testien aloittaminen myöhästyivät kuukaudella. Media-arkiston pilottijärjestelmä, jolla teimme testaukset, pystytettiin toukokuussa 2014 ja ensimmäiset testit päästiin tekemään vasta kesäkuun 2014 loppupuolella, juuri ennen kesälomien alkua. Aikataulumuutos heijastui samalla koko projektin kokonaisaikatauluun ja siten projektipäällikkö joutui laatimaan projektille uuden aikataulun (kuva 14), jossa työ suunniteltiin toteutettavaksi kokonaisuudessaan vuoden 2014 loppuun mennessä.



Kuva 13. Projektin 1. vaiheistus ja aikataulu.



Kuva 14. Projektin 2. vaiheistus ja aikataulu.

Esitutkimus

Ylätason vaatimus tietojärjestelmän kehittämiselle perustui Yritys X:n liiketoimintastrategiaan. Tietojärjestelmän kehittämisprojekti käynnistettiin projektiryhmän aloituskokouksella, jossa oli mukana eri sidosryhmien edustajia. Projektipäällikkö esitteli projektin ja sen ympärillä tehtävät asiat, sekä asiakokonaisuudet. Tietojärjestelmältä vaadittavia tarpeita ja

vaatimuksia oli tarkoitus lähteä kartoittamaan asiakasvaatimusten pohjalta ja tutkia mitä eri vaihtoehtoja toteuttamisen ratkaisemiseksi on olemassa.

Esitutkimusvaiheessa tarkastelimme ja analysoimme ensiksi olemassa olevia tietojärjestelmiä ja toimintatapoja, mutta vielä ei otettu mitään kantaa siihen, mikä järjestelmä palvelut toteuttaisi. Kävimme tarkoin läpi sovellukset, joita Yritys Y:n, arkistointiyksikkö oli aiemmin käyttänyt omissa työnkuluissaan. Pääsovellukset, joilla Yritys Y:n arkistointityö oli tehty, olivat FotoWaren tuoteperheen sovellukset ja toinen sovellus, jota oli käytetty arkistointityössä vain kuvien siirtoon, oli Media-arkiston –sovellus. FotoWaren järjestelmästä oli sillä hetkellä sekä tuotannossa oleva järjestelmä, että kehitysympäristössä oleva uusi järjestelmä. Tuotannossa oleva FotoWaren kuvapankkijärjestelmä oli määrä ajaa alas vuoden 2014 loppuun mennessä, joten vertailussa mukana oleva järjestelmä FotoWaren osalta oli kehitysympäristössä oleva uusi kuvapankkijärjestelmä. Toinen vertailtava järjestelmä oli Media-arkisto, digitaalisen sisällön hallintajärjestelmä, joka oli sillä hetkellä tuotannossa Yritys X:ssä. Projektiryhmän oli tarkoitus arvioida kummankin sovelluksen käyttökelpoisuutta kuvatyönkulkuun liittyvissä toiminnoissa, kuten kuvien hankintaprosessissa, digitaalisen sisällön hallinnassa, sekä kuvien arkistointiprosessissa. Samalla oli tarkoitus kartoittaa, mitä yhteisiä toimintoja sovelluksilla on, millä tavalla ne linkittyvät toisiinsa ja kuinka niiden olemassa olevia toimintoja olisi mahdollista yhdistää. Vertailujen lopputuloksena näistä kahdesta järjestelmästä oli tarkoitus valita toinen, johon tulevat muutokset toteutettaisiin.

Vertailu tapahtui siten, että molempien järjestelmien toimittajat esittelivät erillisissä workshopeissa edustamansa järjestelmän ominaisuuksia ja opinnäytetyön kirjoittaja esitteli tuotannossa olevan kuvapankkijärjestelmän palvelinympäristön ja arkistointisovelluksen tärkeimmät toiminnallisuudet arkistointiprosessin näkökulmasta. Esitysten ja workshopissa käytyjen keskustelujen pohjalta projektiryhmä pohti, kumpaanko järjestelmään tulevat muutokset toteutettaisiin. Projektiryhmä arvioi, voitiinko Media-arkistosovelluksella korvata kaikki ne toiminnot, joita kuvatyönkulkuprosessissa vaadittiin ja jotka oli aiemmin toteutettu FotoWaren tuoteperheen sovelluksilla. Media-arkistojärjestelmä nousi vertailun perusteella ainoaksi järkeväksi ratkaisuksi muutosten toteuttamisympäristönä, koska kyseinen järjestelmä toimi sillä hetkellä muutenkin Yritys X:n digitaalisen sisällön hallintajärjestelmänä. Järjestelmien ylläpidon kannalta oli kustannustehokasta ja järkevää yhdistää kaikki toiminnot yhteen järjestelmään ja lähteä kehittämään olemassa olevaa Media-arkistojärjestelmää siihen suuntaan, kuin kuvatyönkulkuprosessi vaati.

Seuraavaksi kävimme tarkemmin läpi järjestelmän toiminnallisia vaatimuksia, eli mitä muutosten toteuttamisympäristöksi valitun Media-arkistojärjestelmän odotettiin tekevän käyttäjän näkökulmasta.

Vaatimusmäärittelyprosessin vaiheet kuvataan Kotonyan & Sommervillen prosessin mukaisesti, johon kuuluu vaatimusten kartoittaminen, analysointi ja neuvottelu, dokumentointi, validointi ja vaatimustenhallinta.

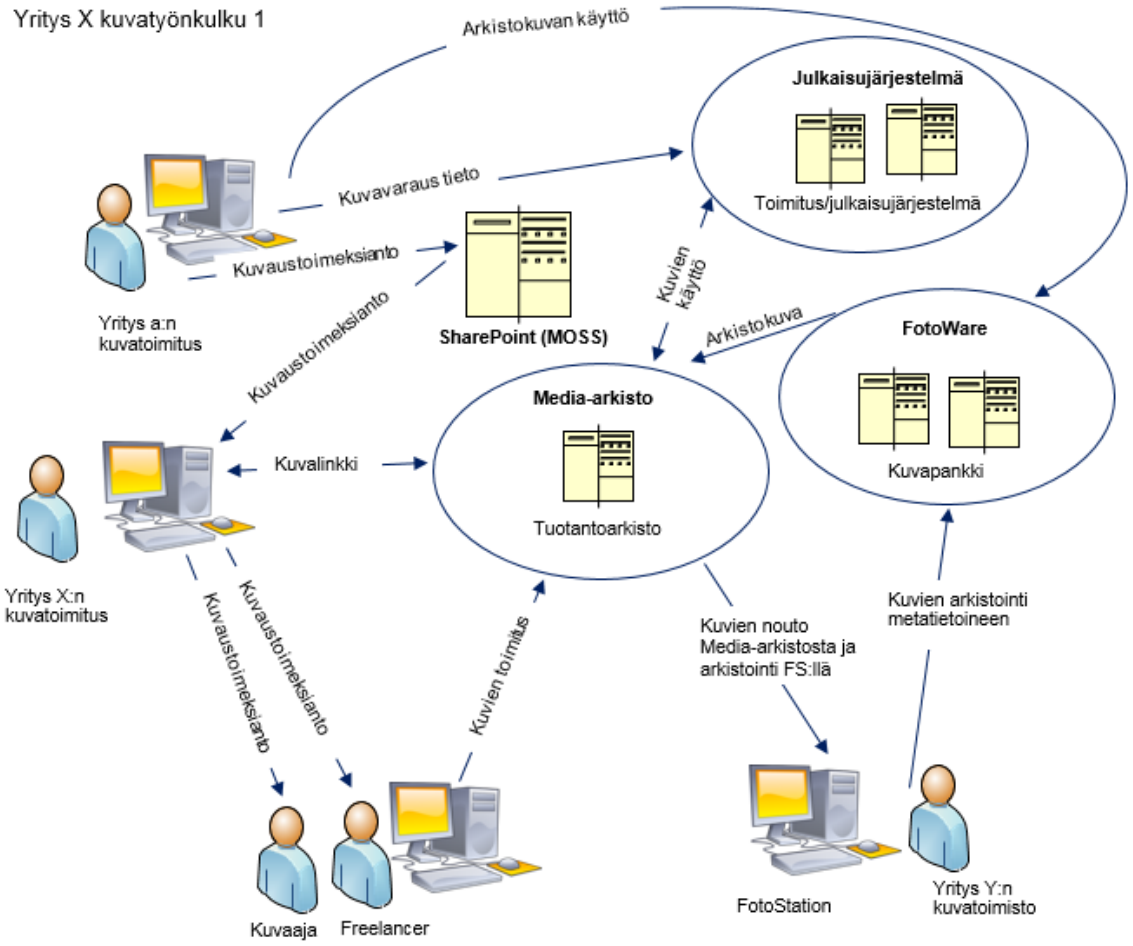
3.1 Vaatimusten kartoittaminen

Vaatimusten kartoittamisvaiheessa asiakasvaatimuksia kerättiin Haikalan & Märijärvenkin mainitsemilla tavoilla: tarkastelemalla ja analysoimalla organisaation tietojärjestelmien prosessikuvauksia ja toimintatapoja sekä workshopeilla, joissa vaatimuksia käytiin läpi eri sidosryhmien kanssa.

Yritys X:n tietojärjestelmien prosessikuvausten, ja toimintatapojen tarkastelun avulla yritimme ymmärtää sovelluksen käyttöliittymää, mahdollisia ongelmia, liiketoiminnan kontekstia, eri sidosryhmien tarpeita ja rajoitteita.

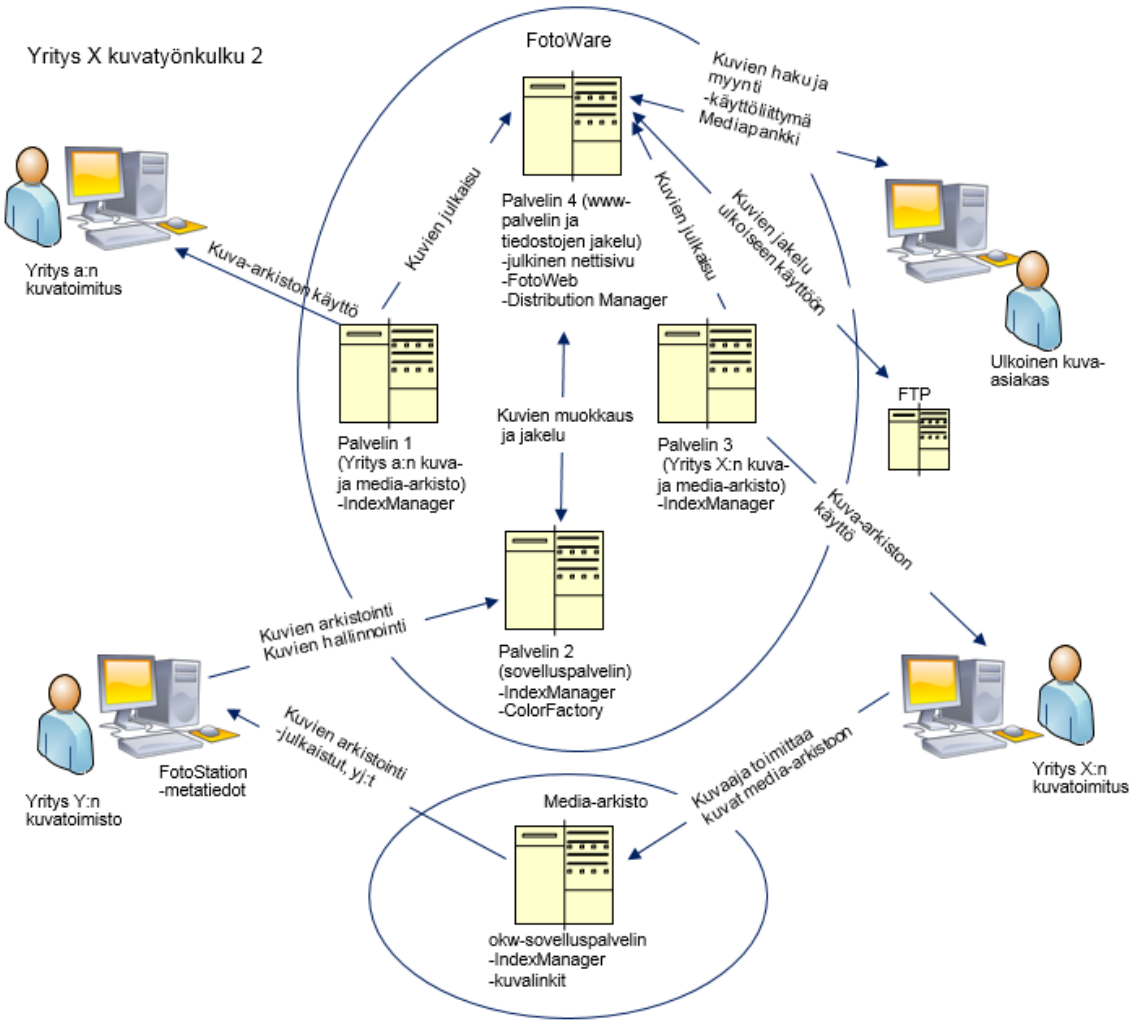
Järjestelmän toiminnallisista vaatimuksista kävimme aluksi läpi käyttöliittymät ja liittymät muihin järjestelmiin, sekä tutkimme miten eri sidosryhmät ovat yhteydessä järjestelmään ja miten he työskentelevät järjestelmän kanssa, sekä millä tavalla tieto virtaa käyttäjän ja järjestelmän välillä. Projektiryhmä yritti ymmärtää asiakkaan ongelman yksityiskohdat sekä työprosessit, jotka liittyivät järjestelmään, jotta kehitettävä tietojärjestelmä voisi tukea mahdollisimman hyvin liiketoiminnan tavoitteita ja loppukäyttäjien työprosessia.

Kuvatyönkulun ensimmäinen prosessikaavio (kuva 15) kuvasi Yritys X:n kuvatyönkulkuprosessia sellaisena, kuin prosessi oli sillä hetkellä. Kaavioon oli hahmoteltu eri järjestelmiin ja kuvatyönkulkuprosessiin liittyvät sidosryhmät. Prosessikaavion avulla tutkimme, onko nykyisessä palveluprosessissa jotain, jonka voisi jättää muuttuneessa prosessissa tekemättä, poistaa tai tehdä tehokkaammin.



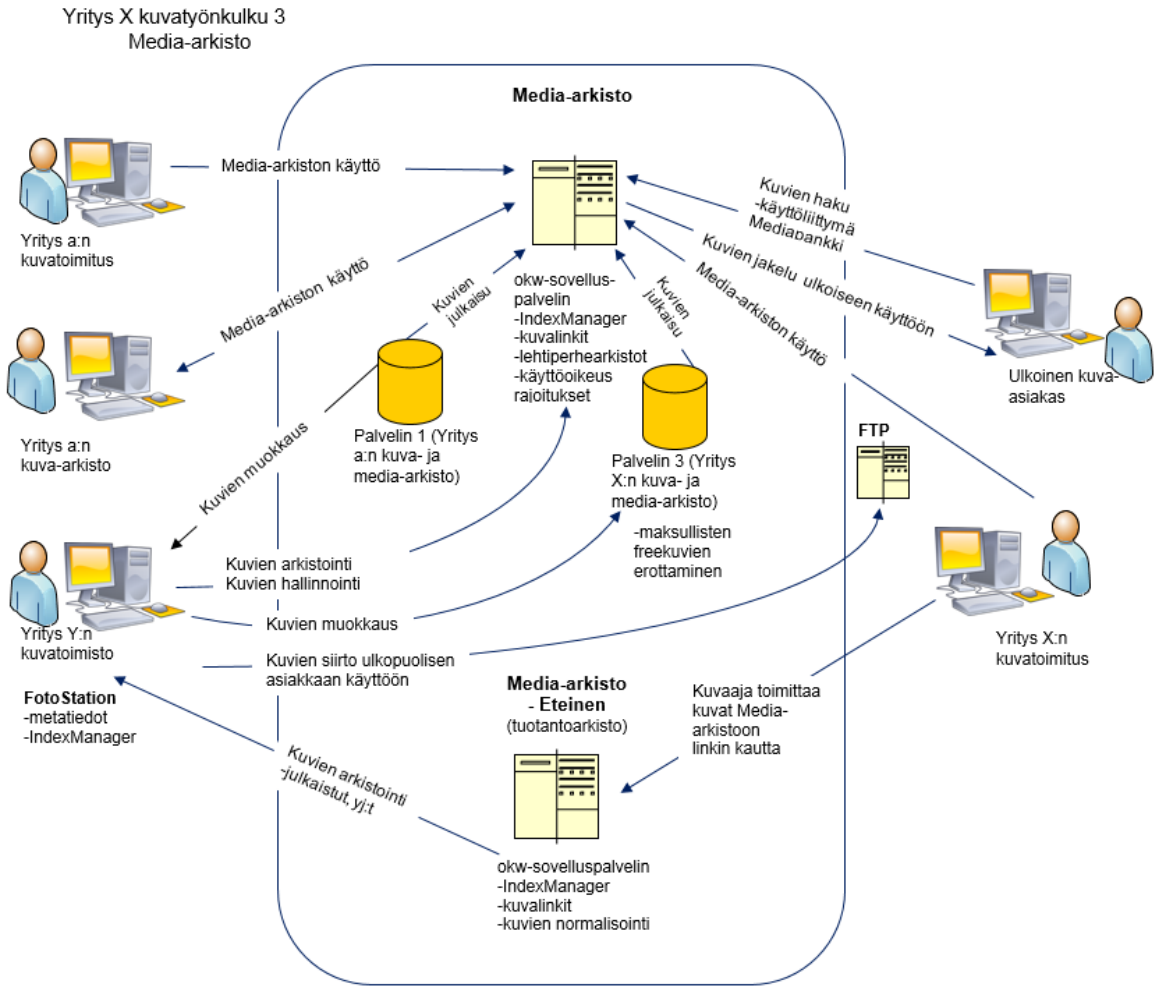
Kuva 15. Yritys X:n kuvatyönkulku 1 – nykytilanteen yleiskuvaus.

Kuvatyönkulun toisessa prosessikaaviossa (kuva 16) käytiin tarkemmin läpi kohdealueen, Yritys Y:n, järjestelmiin liittyvät palvelimet ja niillä ajettavat sovellukset. Lisäksi kaaviossa oli kuvattu järjestelmiin ja kuvatyönkulkuprosessiin liittyvät sidosryhmät, sekä kuinka Media-arkiston sovelluspalvelin linkittyy kuvien arkistointiprosessiin.



Kuva 16. Yritys X:n kuvatyönkulku 2 - nykytilanteen palvelinkuvaukset (FotoWare ja Media-arkisto), sekä ulkoiset ja sisäiset toimijat.

Sen jälkeen, kun olimme ensin tunnistaneet kahden prosessikuvauksen avulla (kuvat 15 ja 16) nykyisen kuvatyönkulkuprosessin sisäiset ja ulkoiset toimijat, prosessissa käytettävät palvelimet ja käyttäjien ja palvelimien väliset toiminnot sellaisena kuin ne ilmenee, asetimme prosessille halutun tavoitteen. Eli kuvasimme prosessin sellaisena, kuin halusimme sen toteutuvan uudessa toimintaympäristössä. Kuvatyönkulun kolmanteen prosessikaavioon (kuva 17), eli tavoitetilakaavioon oli hahmoteltu uusi kuvapankkiratkaisuna toimiva Media-arkistojärjestelmä ja kuvatyönkulkuprosessiin liittyvät toimijat sekä käyttäjien ja palvelimien väliset toiminnot. Käyttäjien toiminnot pysyivät samoina, kuin aiemminkin, vain kuvapankin ja kuvien arkistoinnin sovelluspalvelin vaihtui ja kuvapankin datan tietovarasto vaihtui uusille levyille.



Kuva 17. Yritys X:n kuvatyönkulku 3 – tavoitetilakaavio.

Sen jälkeen, kun olimme saaneet hahmotettua tulevan toimintaympäristön, aloimme kartoittaa yksityiskohtaisemmin asiakasvaatimuksia ja niiden liittymistä ohjelmistovaatimuksiin projektiryhmän workshopeissa.

Ensiksi määritimme sisäisen kuvapankin käyttöliittymää ja siihen liittyviä toimintoja. Alussa asiakkaan puolelta oli havaittavissa tyypillistä Kotonyan & Sommervillenkin mainitsemaa epätietoisuutta, kun asiakkaan piti määritellä tulevan uuden kuvapankin käyttöliittymää Media-arkistojärjestelmässä. Asiakkaan oli alkuun hieman vaikea hahmottaa, kuinka vanhan järjestelmän kuvapankkiin ja kuvien arkistointiin liittyvät toiminnot esitettäisiin ja toteutettaisiin uudessa järjestelmässä.

Käyttäjävaatimusten määrittelyvaiheen edetessä tuli esille, että kaikkia kuvapankkiin ja kuvien arkistointiin liittyviä ominaisuuksia ei ollut valmiina Media-arkistojärjestelmässä,

vaan ne jouduttaisiin räätälöimään sovellukseen. Näistä keskusteltiin kehittäjän kanssa ja tehtiin listaus, millaisia toimintoja sovellukselta vaadittaisiin, jotta kuvapankin sujuva käyttö ja myös arkistointityö onnistuisi Media-arkistojärjestelmässä. Järjestelmän pääkäyttäjän esittämiä vaatimuksia järjestelmän toiminnalle arkistointityön kannalta olivat seuraavasta listasta vaatimukset numero 1, 2, 5, ja 6. Kuvapankin toiminnallisuuteen vaadittavia ominaisuuksia, joita ei ollut valmiina Media-arkistojärjestelmässä, olivat vaatimukset numero 3, 4, 7 ja 8:

1. Kuvien nimeämispainike (liite 2).
2. Statuspohjaiset ajastetut arkiston siirtomoduulit.
3. Lokiraporttien saaminen kuvalatauksista (liite 3).
4. Synonyymisanaston saaminen järjestelmään.
5. Metadatan Hae ja korvaa-toiminto (liite 4).
6. Metadatan oletuspohjat (liite 5).
7. Statuspohjainen ikoni thumbnailkuvan yläpuolelle.
8. Kuvien hintatietojen esille saaminen.

Järjestelmän reunaehtoihin lukeutui käyttöoikeuksiin liittyviä asioita, kuten esimerkiksi se, että tietyllä ulkopuolisella asiakasryhmällä ei saa olla latausoikeuksia korkearesoluutiokuviin, vaan heillä on oikeus ladata ainoastaan näyttökuvia. Lisäksi lehtiperheillä on oikeus nähdä ainoastaan oman lehtiperheen käyttörajoitetut kuvat, mutta ei muiden lehtiperheiden käyttörajoitettuja kuvia.

3.2 Vaatimusten analysointi ja neuvottelu

Esitettyjä vaatimuksia analysoitiin yksityiskohtaisesti ja mietittiin vaatimusten tarpeellisuutta, sekä sitä kuinka esitetyt vaatimukset voidaan toteuttaa. Lisäksi pohdittiin mitkä asiakkaan luettelemista vaatimuksista on ylipäätään toteutettavissa tähän järjestelmään, mitkä ominaisuudet aiotaan toteuttaa tässä projketissa, mitkä voidaan jättää toteutettavaksi myöhemmässä vaiheessa ja mitkä vaatimukset voidaan luokitella kategoriaan ”tullaan toimeen ilmankin”.

Vaatimuksista neuvoteltiin asiakkaan kanssa ja vaatimuksista priorisoitiin tärkeimmät toiminnot, jotka olivat räätälöitävissä sovellukseen ensi vaiheessa. Tärkeimmät vaatimukset olivat numero 1, 2 ja 7: *Kuvien nimeämispainike*, *Statuspohjaiset ajastetut arkiston siirtomoduulit ja Statuspohjainen ikoni thumbnailkuvan yläpuolelle.*

Muutamien vaatimusten osalta jouduttiin toteamaan, että niitä ei pystytä toteuttamaan aikataulullisesti tässä projektissa, eikä ollut myöskään varmuutta, saadaanko niitä toteutettua myöhemminkään. Erityisesti vaatimuksen nro 4, *Synonyymisanaston saaminen järjestelmään* oli hyvin epätodennäköistä. Lisäksi vaatimus nro 3, *Lokiraporttien saaminen kuvalatauksista* asiakkaan esittämällä tavalla olisi vaatinut sen verran suuren räätälöinnin järjestelmässä olevaan raporttitoimintoon, että sitä ei pystytty toteuttamaan tämän kehitysprojektin aikana. Raporttitoiminnon osalta sovittiin, että sitä edistetään myöhemmässä vaiheessa, jossain toisessa kehitysprojektissa. Vaatimus numero 8, *Kuvien hintatietojen esille saaminen*, jäi myös toteutettavaksi myöhemmin, mikäli se olisi mahdollista.

Osa vaatimuksista kategorisoitiin listalle ”toive - ei pakollinen”, toteutetaan myöhemmässä vaiheessa mikäli se on kannattavaa. Tämän tyyppisiä vaatimuksia olivat vaatimukset numero 5 ja 6: *Metadatan Hae ja korvaa-toiminto* sekä *Metadatan oletuspohjat*.

Erityisesti vaatimuksen numero 1 nimeämispainikkeen osalta vaatimuksia jouduttiin tarkentamaan, jotta kyseinen toiminnallisuus saatiin toteutettua ja työmäärä painikkeen tekemiselle arvioitua.

3.3 Vaatimusten dokumentointi

Tässä kehitysprojektissa varsinaista vaatimusmäärittelydokumenttia ei tuotettu Pohjosen suosittelemalla tavalla, vaan dokumentaatio koostui pääasiassa projektiin kuuluvista dokumenteista, jotka kuuluivat projektipäällikön vastuulle, kuten projektisuunnitelmasta, projektin pöytäkirjoista, tilannekatsauksista ja loppuraportista. Projektidokumentaatioon kirjattiin toimeksiannon kuvaus, kohdejärjestelmälle asetetut tavoitteet ja reunaehdot sekä rajoitteet. Kohdejärjestelmän nykytilanteen kuvaukset tuotettiin prosessikaavioina, jotka kuuluivat niin ikään projektidokumentaatioon. Lisäksi liiketoimintaprosessin uimaratakaavio ja järjestelmäympäristön prosessikuvaukset dokumentoitiin projektipäällikön toimesta.

Toiminnalliset vaatimukset kuvattiin järjestelmän pääkäyttäjän toimesta osittain erillisinä dokumentteina kehittäjälle, mutta myös pelkästään suullisesti. Ei-toiminnalliset vaatimukset kuvattiin pelkästään suullisesti. Vaatimukset priorisoitiin suullisesti projektiryhmässä, mutta niitä ei numeroitu erilliseen dokumenttiin. Kaikkia vaatimuksia ei myöskään dokumentoitu itsenäisesti erillisinä dokumentteina, vaan ainoastaan vaatimus numero 1: *Kuvien nimeämispainike*, vaatimus nro 3: *Lokiraporttien saaminen kuvalatauksista* ja vaatimus nro 5: *Metadatan Hae ja korvaa-toiminto* ja vaatimus nro 6: *Metadatan oletuspohjat*.

Järjestelmän pääkäyttäjä dokumentoi kuvapankin toimintoja ja teki samalla kuvapankin käyttöohjeet, jotka toimivat myös koulutusmateriaalina ja osittain testausmateriaalina ennen käyttöönottovaihetta tehdyissä loppukäyttäjätesteissä.

Alkuvaiheen testaussuunnitelma sisältyi kehitysprojektin projektisuunnitelmaan, johon oli merkitty testeihin kuluva aika. Varsinaista testaussuunnitelmadokumenttia testauksesta ei kuitenkaan tehty, vaan asiat sovittiin suullisesti kuka testit tekee, milloin testit tehdään ja miten testit järjestetään. Sen sijaan kehittäjä teki projektiryhmälle kirjallisen ohjeen testiympäristön käytöstä.

Asiakasvaatimusten dokumentoimisessa ei noudatettu Haikalan & Märijärven suosittelemaa tapaa, jossa vaatimuksesta olisi pitänyt olla yksityiskohtaista tietoa, jotta se on tarkka, ymmärrettävä, testattava, jäljitettävä taaksepäin ja eteenpäin. Sen sijaan samaa yksityiskohtaista tietoa käytiin osittain workshopeissa suullisesti läpi. Vaatimukset, jotka kuvattiin jollakin muulla tavalla kuin suullisesti, kuvattiin kuvaruutukaappauksina, jotka oli selitetty luonnollisella kielellä (liitteet 2, 3, 4, 5).

Yhtään vaatimusta ei kuvattu UML-notaatiolla, eli käyttötapauslomakkeella, käyttötapauskaaviona tai riippuvuuskaaviona, jota suositellaan yleisesti kaikissa alan kirjallisuuslähteissä, joita käytettiin tämän opinnäytetyön tietolähteinä.

3.4 Vaatimusten validointi

Vaatimusmäärittelyn viimeinen vaihe, vaatimusten validointi eli kelpoistaminen, tehtiin projektiryhmän kanssa yhdessä siten, että toteutettujen vaatimusten oikeellisuutta ja laatua testattiin koko projektin alusta sen loppuun saakka järjestelmän pilottiympäristössä, jossa vaatimusten toteutuminen voitiin varmistaa. Validoinnissa ei käytetty varsinaisesti mitään tarkistuslistoja. Esimerkiksi nimeämispainikkeen kohdalla tarkistus tehtiin siten, että pilottiympäristössä testattavissa olevaa toiminnallisuutta verrattiin vielä tuotannossa olevan vanhan FotoWaren kuvien arkistointiohjelman vastaavaan toimintoon, jonka avulla vaatimuksen toteutuminen varmistettiin. Mikäli vaatimus ei vastannut asiakkaan vaatimuksia, kyseistä toiminnallisuutta tai ominaisuutta konfiguroitiin kehittäjän toimesta halutun mukaiseksi.

3.5 Vaatimustenhallinta

Prosessin aikana projektipäällikkö pyrki ensisijaisesti varmistamaan, että kehitettävä järjestelmä vastasi niitä vaatimuksia, jotka sille oli asetettu. Tietojärjestelmän omistajan vas-

tuulle kuului vaatimusmäärittelytyön valvonta sekä määrittelykuvauksien hyväksyminen. Projektipäällikkö vastasi kokonaisuudesta, johon kuului vaatimusmäärittely, projektin ositus, resursointi sekä viestintä ja yhteydenpito eri sidosryhmiin. Vaatimusmäärittelyprosessin kuluessa vaatimuksia tarkennettiin ja niistä pyrittiin saamaan selkeitä, yksiselitteisiä ja ristiriidattomia. Muutospyyntöjen hyväksyntä vaatimuksiin tapahtui projektin ohjausryhmässä.

Testaus

Testitapaukset valittiin lasilaatikkotestausmenetelmän mukaisesti, joka tarkoittaa sitä, että ensin tutustutaan ohjelman toteutukseen, jonka perusteella testitapaukset valitaan. Tutustuimme tässä projektissa ensin FotoWaren kuvapankkijärjestelmän ja arkistointiohjelman toteutukseen. Arkistojen siirtomoduulit ja kuvien nimeämispainike olivat kriittisiä arkistointityön kannalta, ja niiden toteutukselle haettiin mallia FotoWaren järjestelmästä. Kyseiset toiminnallisuudet pyrittiin toteuttamaan Media-arkistojärjestelmään niiltä osin kuin se oli mahdollista.

Testausta tehtiin projektin alkuvaiheessa ja loppuvaiheessa eniten. Projektin alussa, eli pienkehitystyön määrittelyvaiheessa, testaukset järjestelmän pilottiympäristössä kestivät noin viikon. Alkuvaiheessa pilottiympäristössä tehtiin käytettävyystestausta järjestelmän pääkäyttäjän ja projektipäällikön toimesta ketterällä kehitysmenetelmällä, joka tarkoittaa sitä, että kehittäjä kirjoitti ensin koodia, jonka jälkeen testaajat tekivät testit. Sen jälkeen koodiin tehtiin vaadittavia korjauksia ja jälleen testattiin. Tätä jatkui niin kauan, kunnes lopputulos oli hyväksyttävä. Tällä testauksella pyrittiin varmistamaan kuvapankin käyttöliittymän ja arkistojen siirtomoduulien toimivuus.

Projektin loppupuolella, kun kuvien nimeämispainike ja kuvien statussymbolit saatiin valmiiksi, alkoivat kyseisen toiminnallisuuden testaukset, johon osallistui järjestelmän pääkäyttäjän lisäksi kaksi loppukäyttäjää. Nämä testit veivät projektista eniten aikaa, noin kolme viikkoa. Nimeämispainiketta jouduttiin konfiguroimaan testauksen aikana muutamaan kertaan. Testit lopetettiin siinä vaiheessa, kun toiminnallisuus oli niin looginen ja käytettävä, että kyseisen toiminnallisuuden loppukäyttäjät hyväksyivät sen. Testauksessa tuli vastaan myös projektin aikataulu, koska uusi kuvapankki piti saada tuotantoon uuden aikataulun mukaisesti marraskuun alussa ja vanha järjestelmä piti ajaa alas joulukuun loppuun mennessä.

Viimeinen testaus, joka oli käytettävyystestaus, ajoittui projektin loppupuolelle vähän ennen kuin uudet toiminnallisuudet otettiin tuotantoon. Järjestelmän pääkäyttäjän tekemä kuvapankin käyttöohje toimi testausmateriaalina näissä loppukäyttäjätesteissä. Testitilanteeseen valittiin testihenkilöt sen perusteella, että heidän piti olla kuvapankin aktiivikäyttäjiä erilaisista käyttäjäryhmistä.

Testitilanne järjestettiin siten, että testaajilta kysyttiin suostumus testaukseen ja sovittiin heidän kanssaan tietty ajankohta testien suorittamiselle. Testaustilanteessa testaajalle kerrottiin lyhyesti testauksen kulku. Testaajille tehtiin pieni ohjeistus kuvapankin toiminnasta, jonka jälkeen ohjeistus tulostettiin paperille. Tämä ohjeistus toimi samalla käyttöönottovaiheen koulutusmateriaalina. Ohje otettiin mukaan testaustilanteeseen, jossa testaaja sai lukea ohjeen ensin kerran läpi ja sen jälkeen hän suoritti testin ohjeiden mukaisesti. Projektiryhmän kaksi jäsentä, järjestelmän pääkäyttäjä ja sovellusalueen asiantuntija, valvoivat testaajan vierellä, kun testihenkilö teki testin ja havainnoivat samalla käyttäjän toimimista, jolloin oli mahdollista saada selville tärkeää tietoa kuvapankin toteutuksen toimivuudesta tai toimimattomuudesta ja tietoa siitä, kuinka käyttäjä odotti järjestelmän toimivan. Testeissä tuli esille, että kuvapankin käyttöliittymä oli looginen ja käyttäjästävällinen, ja testaajat kokivat kuvapankin käyttölogiikan hyvin helpoksi ja selkeäksi.

4 Loppuyhteenveto

Tämän Yritys X:n ylläpitovaiheessa olevan tietojärjestelmän pienkehitystyö onnistui melko hyvin ja tietojärjestelmästä saatiin suurelta osin asiakkaan vaatimuksia vastaava, vaikka kohdeorganisaatio ei käyttänytkään selvästi määriteltyä ja standardisoitua vaatimusmäärittelyprosessia. Projektin suuri vahvuus oli kokenut projektipäällikkö, joka kykeni hallitsemaan projektia sen koko elinkaaren ajan, vaikka aikataulusta myöhästyttiinkin. Aikataulun siirtyminen ei kuitenkaan ollut projektipäälliköstä kiinni, vaan business-kriittinen päivitys jouduttiin tekemään ohi tämän projektin, jossa oli kiinni samat resurssit kuin tässäkin projektissa, ja projektin ajoittumisen vuoksi kesälomat venyttivät aikataulua lisää kahdella kuukaudella. Projektin budjettia ei kuitenkaan ylitetty, vaan projekti pysyi sille asetetun budjetin kehyksessä.

Myös järjestelmän pääkäyttäjän osuus projektin onnistumisen kannalta oli tärkeä, koska hän edusti vanhan järjestelmän asiantuntijaa, ja pystyi kuvaamaan halutut vaatimukset riittävällä tarkkuudella ja pääosin yksiselitteisesti ja ristiriidattomasti. Erityisen tärkeässä roolissa oli myös prototyyppilähestymistavan käyttäminen, jossa toteutettavan järjestelmän kehitysympäristöön luotiin uuden kuvapankin käyttöliittymä ja kyseisessä ympäris-

tössä pystyttiin testaamaan käyttäjävaatimuksiin liittyviä erilaisia moduuleita jo projektin alkuvaiheista lähtien. Siten moduuleihin pystyttiin tekemään tarvittavat korjaukset ketterästi projektin kuluessa.

Kaikkia vaatimuksia ei pystytty toteuttamaan tämän kehitysprojektin aikana, vaan osa vaatimuksista jäi toteutettavaksi myöhemmin. Vaatimusten määrittelyä helpotti olennaisesti se, että projektiryhmän ei tarvinnut lähteä määrittelemään uutta järjestelmää niin sanotusti puhtaalta pöydältä, vaan vanhaa, vielä tuotannossa olevaa järjestelmää pystyttiin hyödyntämään vaatimusmäärittelytyön pohjana. Toisaalta tämä sama asia oli myös jossain määrin rasite, koska vaatimusten määrittelyn alkuvaiheessa järjestelmän pääkäyttäjän oli vaikea irrottautua vanhan järjestelmän kaikista ”hienouksista” ja karsia niitä toimintoja, jotka eivät olleet aivan välttämättömiä uudessa järjestelmässä. Tähän ongelmaan vaikutti se, että pääkäyttäjä ei tuntenut ennestään uutta kohdejärjestelmää ja sen mahdollisuuksia. Vanhasta järjestelmästä oli hyötyä myös siten, että vaatimuksia pystyttiin testaamaan koko kehitysprojektin ajan vertaamalla uuteen järjestelmään tehtyjä toimintoja vanhan järjestelmän vastaaviin toimintoihin.

Analysoinnin perusteella havaittiin, että suurin puute kehitysprojektin osalta oli vähäinen dokumentointi siltä osin, kuin se liittyi vaatimusten määrittelyyn. Vaatimusten määrittelyvaiheessa ei käytetty valmiita dokumenttipohjia, eikä vaatimuksia myöskään mallinnettu UML- kaaviotekniikaa apuna käyttäen. Vaatimukset kuvattiin pääosin vain suullisesti, sekä osittain kuvaruutukaappauksina vanhasta järjestelmästä, joita selitettiin luonnollisella kielellä. Projektissa luotiin Haikalan & Märijärven suosittelemasta minimidokumentaatiosta projektisuunnitelma ja osittainen testaussuunnitelma. Sen sijaan projektissa ei luotu varsinaista vaatimusmäärittelydokumenttia ja suunnitteludokumenttia.

Projektissa vaatimusten dokumentoinnin vähäisyyteen vaikutti pienten henkilöresurssien, pienen projektin ja tiukan aikataulun lisäksi myös se, että projektipäällikkö ja tietojärjestelmän kehittäjä olivat kokeneita tämän tyyppisten projektien läpi viemisessä, joten vaatimusten määrittely pystyttiin toteuttamaan ilman kattavaa dokumentointimenetelmääkin.

Alan kirjallisuuden perusteellakin tuli esille, että vaatimusmäärittelyyn liittyvät dokumentit tulisi tuottaa projektin koon ja toteutettavan sovelluksen luonteen mukaisella tarkkuudella. Eli pienemmissä kehitysprojekteissa ja vähemmän kriittisissä sovellustoteutuksissa dokumentin tuottaminen on epämuodollisempaa, eikä kaikista raskainta ja kattavinta dokumentointimenetelmää välttämättä kannata käyttää. Sen sijaan esimerkiksi isoissa projekteissa tai kriittisissä järjestelmissä vaatimusmäärittelyyn liittyvät dokumentit tulisi tuottaa Haika-

lan & Märijärvenkin suosittelemalla tarkkuudella, jolloin kaikki vaatimukset analysoidaan, priorisoidaan, dokumentoidaan, validoidaan ja testataan esimerkiksi Kotonyan & Sommervillen vaatimusmäärittelyprosessin mukaisesti.

Kohdeorganisaatiolle suositellaan, että siinä vaiheessa, kun tietojärjestelmän kehitysprojektia ollaan käynnistämässä, täytyisi projektisuunnitelmaan kirjata vaatimusten määrittelylle tarvittavat resurssit, sekä aikaa että henkilöitä, jotta myös vaatimusmäärittelydokumentti ehdittäisiin tuottaa kehitysprojektin aikana.

Olisi myös tärkeää, että yhtiössä olisi erikseen vaatimusmäärittelijät, joilla on riittävä osaaminen ja ajantasainen koulutus esitutkimuksen, vaatimusmäärittelyjen ja järjestelmäanalyysien tekemiseen ja joiden vastuulle kyseessä olevat tehtävät selkeästi kuuluisivat. Jos kyseisiä henkilöitä ei ole yhtiössä, vaatimusmäärittelyprosessi jää helposti hyvin kevyeksi, eikä esimerkiksi dokumentointia tehdä ollenkaan, kuten Haikala & Märijärvi toteavat.

Projektissa havaittiin ajoittain kommunikointiongelmia ohjelmoijan ja järjestelmän pääkäyttäjän välillä. Molemmat käyttivät sellaista termistöä, jota toinen ei ymmärtänyt. Sen vuoksi olisikin hyvä tehdä Haikalan & Mikkosenkin suosittelema käsitteiden määrittelylista heti projektin alussa, johon kirjattaisiin projektissa käytettävä keskeinen termistö.

Yhtiössä on olemassa yhtenäiset toimintatavat ja vaatimusmäärittelydokumentit vaatimusmäärittelytyötä varten paperilla, mutta näitä toimintamalleja ei ole jalkautettu tarpeeksi tehokkaasti vaatimusmäärittelytyötä tekevien keskuuteen, koska yhtiössä työskennellään tällä hetkellä usealla eri tavalla. Toiset käyttävät näitä toimintamalleja vaatimusmäärittelytyössään ja toisilla taas on käytössään omia erilaisia menetelmiä. Esimerkiksi tässä projektissa vaatimusmäärittelyyn ei käytetty yhtiön omaa mallipohjaa (liite 6) tai muita yhtiön vaatimusmäärittelyyn kuuluvia ohjeistusdokumentteja. Siten yhteiset toimintatavat kannattaisi ottaa käyttöön kaikkien vaatimusmäärittelytyötä tekevien keskuudessa, jotta voidaan varmistaa, että yhtiön omat standardit dokumentin tuottamiseen täyttyvät. Näin pystytään varmistamaan, että kaikkien vaatimusmäärittelijöiden keskuudessa kaikki oleellinen tieto sisältyy tuotettuun vaatimusmäärittelydokumenttiin.

Olisi myös suositeltavaa dokumentoida yksittäiset vaatimukset esimerkiksi Robertsonin & Robertsonin mallipohjan avulla, jolloin jokaiselle vaatimukselle saataisiin yhteneväinen rakenne. Siten olisi helpompi tarkistaa, että jokaisella vaatimuksella on sille asetetut kriteerit. Näin vaatimuksista saataisiin laadukkaita, jäljitettäviä ja testattavia.

Hyväksytyt vaatimukset olisi suotavaa säilyttää esimerkiksi erillisessä vaatimustietokannassa. Siten kertaalleen laadittuja vaatimuksia pystyttäisiin hyödyntämään uudelleen tulevissa projekteissa, joissa on tarvetta samanlaisille tai samantyyppisille vaatimuksille, kuten Kotonya & Sommerville toteavat.

Yhtiön vaatimusmäärittelydokumentaatiossa on ohjeistettu, kuinka käyttötapauksen tekstuaalinen kuvaaminen tulee toteuttaa lomakepohjan avulla, mutta dokumentaatiossa ei ole neuvottu, kuinka vaatimukset tulee kuvata graafisesti UML-kaaviotekniikalla, kuten esimerkiksi käyttötapauskaaviona tai riippuvuuskaaviona. Tämä olisi tärkeä lisä ohjeistukseen luonnollisen kielen rinnalle, koska graafisten käyttötapausten avulla pystytään kuvaamaan havainnollisesti mitä käyttäjä pystyy tekemään järjestelmällä. Käyttötapauskaavioiden ja riippuvuuskaavioiden avulla saataisiin myös lisättyä ymmärrystä eri sidosryhmien välillä ja vältyttäisiin vaatimusten vääriltä tulkinnoilta.

Lisäksi yhtiön vaatimusten dokumenttiohjeistuksessa on mainittu minkälaisia luonnollisella kielellä kirjoitettujen vaatimusten tulee olla, mutta ohjeessa ei ole annettu esimerkkiä oikein muotoillusta vaatimuksesta luonnollisella kielellä. Ohjeistukseen voisi liittää muutaman malliesimerkin luonnollisella kielellä tuotetusta vaatimuksesta, sekä Pohlin & Ruppin listaamat asiat, jotka heidän näkemyksensä mukaan sisältävät kaikista oleellisimmat ongelmat vaatimusmäärittelijöiden työn kannalta, joihin vaatimusmäärittelijöiden tulisi kiinnittää huomiota vaatimuksia määritellessään. Näistä kaikista ongelmista voisi laittaa dokumenttiin esimerkit, kuinka ne oikeaoppisesti tulee määritellä. Dokumenttiin voisi myös lisätä Pohlin & Ruppin esittämän viisi vaiheisen vaatimusten kuvaamisen luonnollisella kielellä, jota he pitävät oikeana menetelmänä yksiselitteisten ja laadukkaiden vaatimusten kuvaamisessa. Jokaisesta viidestä vaiheesta voisi kuvata lyhyesti, kuinka kutakin vaihetta tulee käyttää ja loppuun tästä kuvausmallista tulisi liittää esimerkkikuva ohjeen havainnollistamiseksi.

Yhtiön vaatimusmäärittelydokumentin mallipohjasta puuttuu vaatimusten luokittelun yksi osa-alue: rajoitteet ja reunaehdot. Dokumenttiin tulisi lisätä oma osio myös tälle vaatimusluokalle.

Projektiin liittyvä dokumentaatio oli osittain vain sovelluksen kehittäjien hallussa. Jatkossa kaikki projektin dokumentaatio tulisi olla lisäksi tietohallinnon ja liiketoiminnan saatavilla, jotta dokumentaatiota voitaisiin hyödyntää myös seuraavissa projekteissa.

Yhtiön tulevissa projekteissa säästettäisiin sekä aikaa, että rahaa, jos järjestelmään liittyvät, jo kertaalleen selvitetyt asiat dokumentoitaisiin. Siten uudet (tai vanhat) tekijät voisivat hyödyntää jo olemassa olevaa dokumentaatiota, eikä selvitystyötä tarvitsisi tehdä uudestaan. Samalla vältyttäisiin myös siltä, että järjestelmään tai vaatimuksiin liittyvä tieto ei ole vain tiettyjen henkilöiden hallussa, jotka saattavat odottamatta poistua yhtiöstä, jolloin on vaarana, että tärkeää tietoa katoaa heidän mukanaan.

Oma oppimiseni

Valitsin aiheekseni vaatimusmäärittelyn, koska koin sen aihealueena mielenkiintoiseksi, mutta myös erittäin vaativaksi aihealueeksi. Halusin oppia mahdollisimman paljon vaatimusmäärittelyn prosessista ja erityisesti vaatimusmäärittelydokumentin tuottamiseen liittyvistä tekniikoista ja menetelmistä. Olen ollut vuosien varrella mukana erilaisissa tietojärjestelmän kehittämisprojekteissa, joissa olen määritellyt vaatimuksia asiakkaan näkökulmasta, mutta en ole osannut käyttää määrittelyyn yleisesti alalla käytettäviä prosessimalleja ja tekniikoita. En ole myöskään törmännyt työelämässä kyseisiin malleihin tai tekniikoihin aiemmin, joten oli erittäin mielenkiintoista käydä läpi eri lähteistä vaatimusmäärittelyn prosessia ja siihen liittyviä menetelmiä, joista on minulle varmasti paljon hyötyä tulevissa tietojärjestelmien kehitysprojekteissa.

Opinnäytetyöprosessin aikana olen saanut kirjallisuuden perusteella erittäin kattavan ja hyvän kuvan siitä, mitä vaatimusten määrittely on ja mitä vaatimusmäärittelyprosessin lopputuotoksen, eli toiminnallisten vaatimusten määrittelydokumentin tuottamiseen vaaditaan. Vaatimusmäärittelyprosessissa ei voi varmaankaan liikaa painottaa kehitettävän järjestelmän sidosryhmien ja erityisesti loppukäyttäjien tärkeää roolia, jotka ovat parhaita asiantuntijoita uuden järjestelmän vaatimusten määrittämisessä.

Tämän projektin myötä ymmärrän myös kuinka tärkeää on, että projektilla on kokenut projektipäällikkö, joka hallitsee projektin sen koko elinkaaren ajan, ja joka hallitsee myös vaatimusmäärittelyprosessin ja sitä kautta vaatimuksiin tulevat muutokset. Lisäksi näen erittäin tärkeänä, että vaatimusmäärittelytyötä tekevien henkilöiden koulutus pidetään ajantasaisena ja riittävänä, jotta vaatimusmäärittelyt ovat tarpeeksi laadukkaita, jolloin ne tuottavat järjestelmään asiakkaan haluamat vaatimukset ja kehitettävällä järjestelmällä pystytään tarjoamaan optimaalista arvoa sen omistajalle.

Lähteet

Fowler, M. & Scott, K. 2002. UML. Docendo Finland Oy.

Haikala, I. & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. 12., uudistettu painos. Talentum. Helsinki.

Haikala, I. & Märijärvi, J. 2006. Ohjelmistotuotanto. 11. painos. Talentum. Helsinki.

Helsingin yliopiston www-sivut, 2009. 582104 - Ohjelmistojen mallintaminen, käyttötapauksiin perustuva vaatimusmäärittely.
Luettavissa: https://www.cs.helsinki.fi/u/pohjalai/ke09/ohma/slides/ohma_04-vaatimusmaarittely.pdf Luettu: 22.5.2016.

Hirsjärvi S., Remes P. & Sajavaara P., 2009. Tutki ja kirjoita. 15., uudistettu painos. Tammi. Helsinki.

JUHTA – Julkisen hallinnon tietohallinnon neuvottelukunta 2012. JHS 173 ICT-palvelujen kehittäminen: Vaatimusmäärittely. Luettavissa: http://www.jhs-suositukset.fi/suomi/jhs173 Luettu: 1.6.2016.

Kotonya, G. & Sommerville, I. 2004. Requirements Engineering Processes and Techniques. John Wiley & Sons. Great Britain.

Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. 1. painos. Docendo.

Pohl, K. & Rupp, C. 2011. Requirements Engineering Fundamentals. Rocky Nook Inc. Santa Barbara.

Robertson, S. & Robertson, J. 2013. Mastering the Requirements Process. Third Edition. Addison-Wesley. Pearson Education Inc. New Jersey.

Liitteet

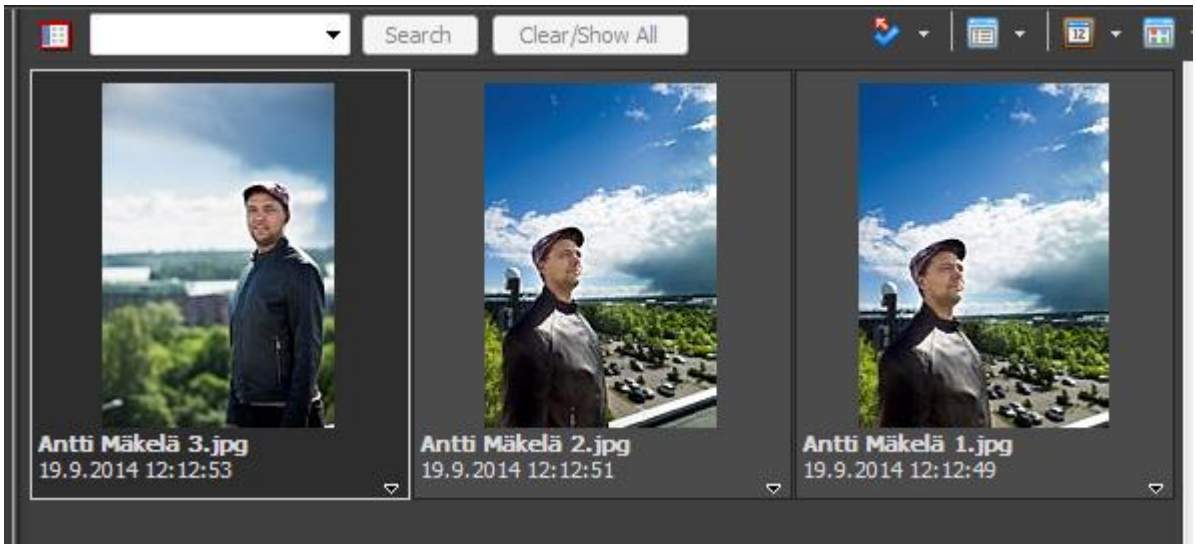
Liite 1. Snow Card – vaatimusten dokumentointipohja

(Robertson & Robertson 2013, 30)

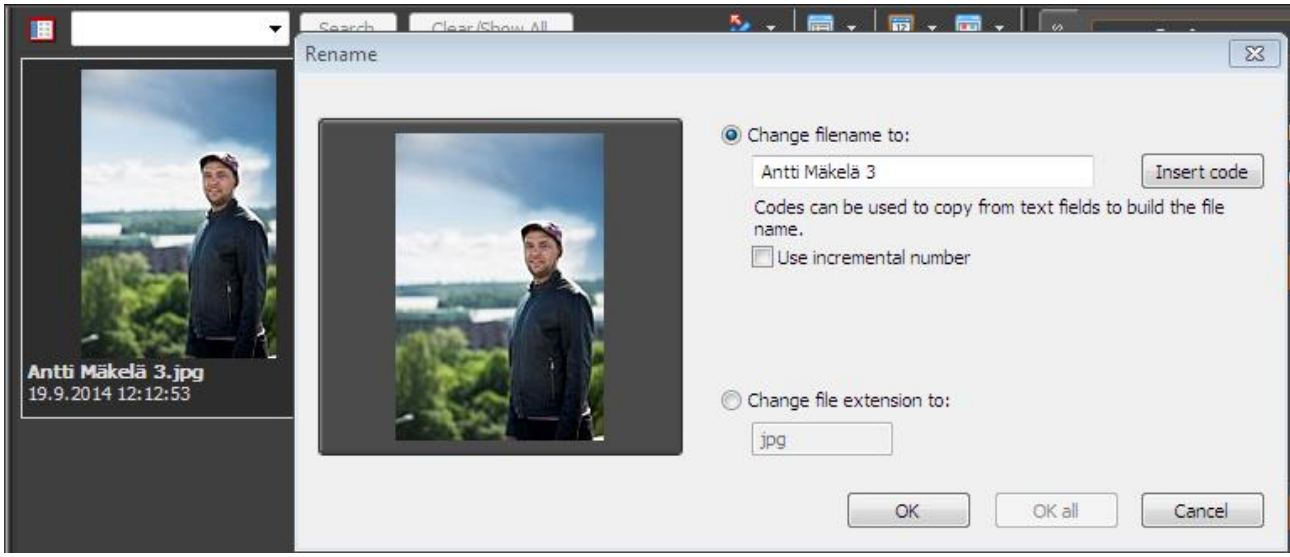
Lomakepohja vaatimuksen dokumentointia varten	
Attribuutti	**Selitys**
Vaatimus #	Vaatimuksen yksilöllinen tunniste
Vaatimustyyppi	Onko kyseessä toiminnallinen vaatimus, ei-toiminnallinen vaatimus vai rajoitus?
Tapahtuma/BUC/PUC #	Lista sidosryhmän tapahtumista/liiketoiminnan käyttötapauksista/tuotteen käyttötapauksista, jotka tarvitsevat tätä vaatimusta.
Vaatimuksen kuvaus	Kuvaa vaatimus yhdellä lauseella.
Vaatimuksen perustelu	Miksi vaatimus pitäisi hyväksyä?
Vaatimuksen esittäjä	Kuka vaatimuksen on esittänyt?
Hyväksymiskriteeri	Aseta vaatimukselle mittari, jolla voidaan testata, täsmääkö toteutettu ratkaisu alkuperäiseen vaatimukseen.
Asiakkaan tyytyväisyys	Asiakkaan tyytyväisyys jos vaatimus toteutetaan tuotteeseen (asteikolla 1-5).
Asiakkaan tyytymättömyys	Asiakkaan tyytymättömyys, jos vaatimusta ei toteuteta tuotteeseen (asteikolla 1-5).
Prioriteetti	Vaatimuksen tärkeys asiakkaalle.
Konfliktit	Jos tämä vaatimus toteutetaan, mitä vaatimuksia jää toteuttamatta?
Mahdollinen muu tietotarve	Liittyykö tähän vaatimukseen jotain dokumentteja tai kuvauksia, jotka selittävät tätä vaatimusta.
Versio	Luonti, muutokset, poistot jne.

Liite 2. Vaatimus 1 - Kuvien nimeämispainike

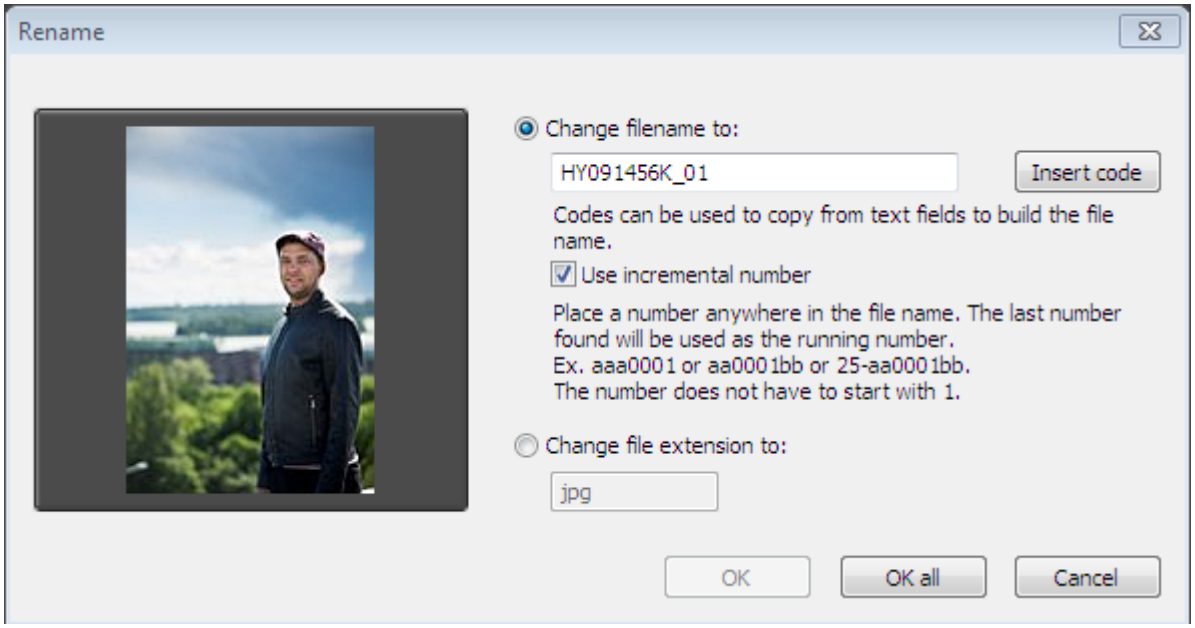
1. Kuvien arkistoija työskentelee kansioissa, joissa on eri määrä kuvia, riippuen siitä, montako kuvaa kyseiseen juttuun on kuvattu tai paljonko kuvia on ladattu kansioon.
2. Käyttäjä ottaa työn alle kansion kerrallaan.



3. Käyttäjä valitsee kansion, jossa on 3 kuvaa, jotka hän aikoo nimetä uudelleen.
4. Käyttäjä **aktivoi** kansion kaikki kuvat, jotka täytyy nimetä uudelleen.

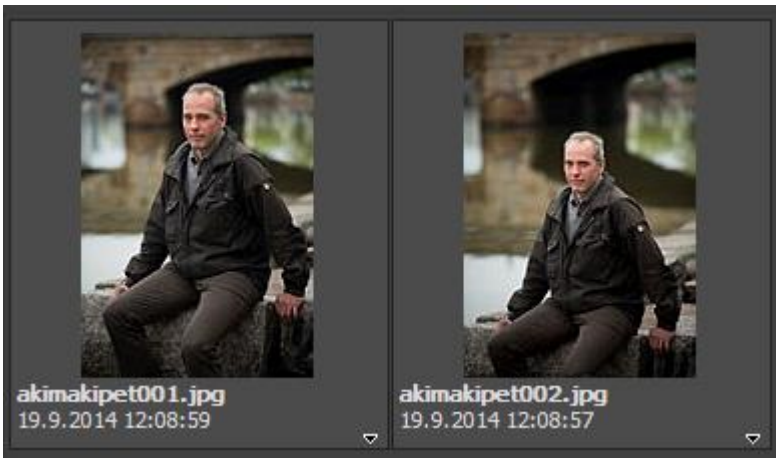


5. Käyttäjä painaa **F2**-näppäintä, jolloin näytölle aukeaa **Nimeä/Rename**- popup-ikkuna. Ikkunassa näkyy preview-kuva, jonka tiedostonimeä ollaan muuttamassa, sekä **Change filename to** –kentässä kuvassa jo olemassa oleva tiedostonimi.

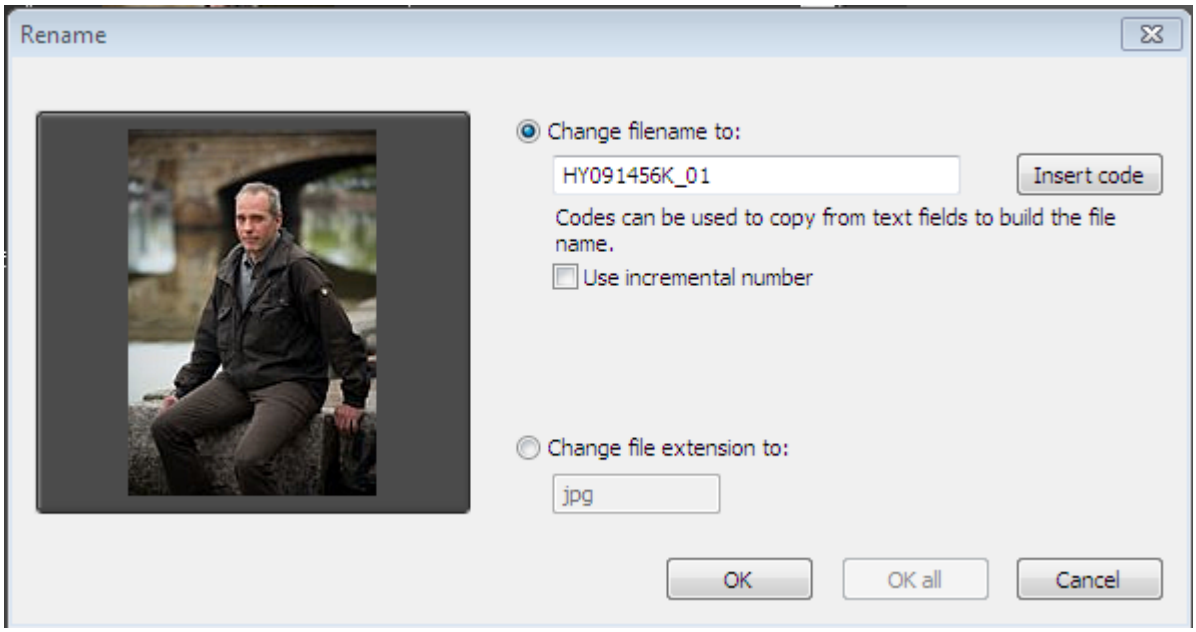


6. Käyttäjä kirjoittaa **Change filename to**: riville haluamansa tiedostonimen: HY091456K_01 ja laittaa rastin kohtaan ” **Use incremental number**”.
7. Lopuksi käyttäjä painaa **OK all**, jolloin kaikki valitut 3 kuvaa nimetään juoksevasti samalla logiikalla.

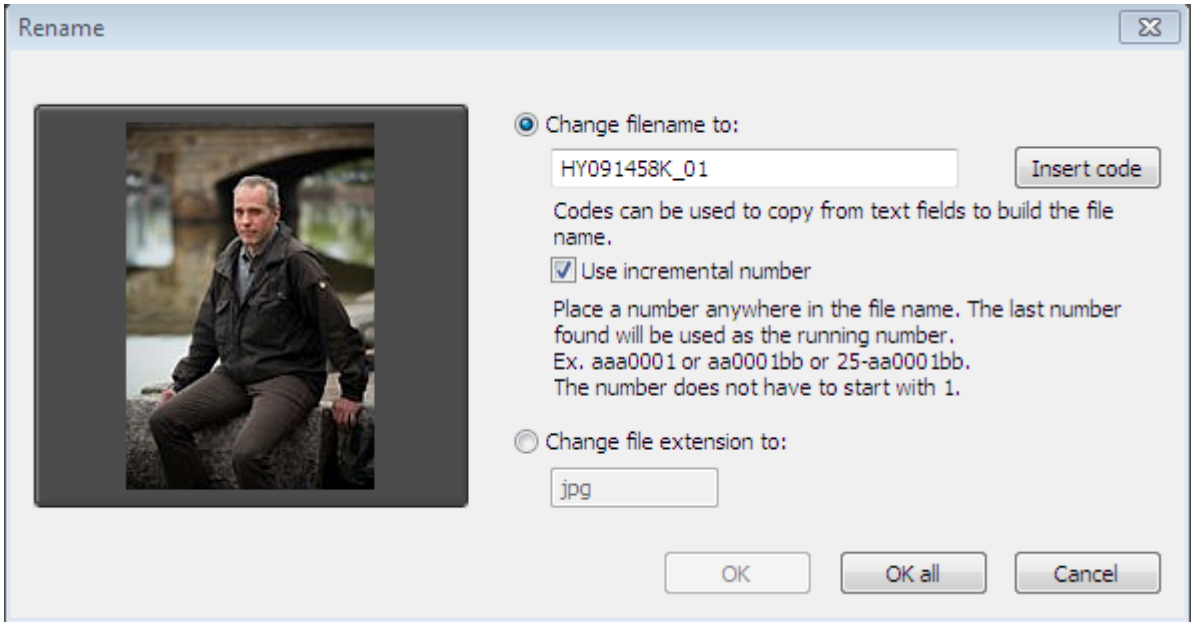
8. Käyttäjän OK-painalluksen jälkeen näkymä palaa samaan kuvavalikoimaikkunaan, josta käyttäjä lähti liikkeelle. Nyt kuvien alapuolella näkyy juuri muutettu tiedosto-nimi.



9. Seuraavaksi käyttäjä menee työskentelemään uuteen kansioon, jossa on kaksi kuvaa, jotka sijaitsevat sivulla 58. Käyttäjä **aktivoi** kansion kaikki kuvat, jotka täytyy nimetä uudelleen.



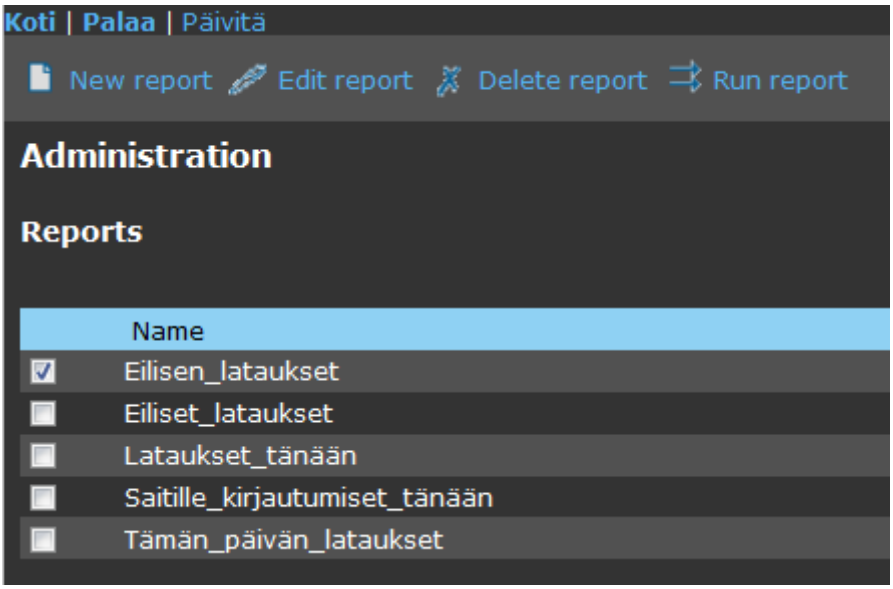
10. Käyttäjä painaa **F2**-näppäintä ja nimeää kuvan edellisen kuvasarjan logiikalla. Käyttäjä laittaa edellisen kuvan tiedostonimen CTRL+V –toiminnolla **Change filename to**: kenttään ja muuttaa tiedostonimestä vain **yhden** numeron. Eli sivunumeron, joka on nyt 5**8**.



11. Kuvassa sivunumero 58 muutettu ja rasti *use incremental number* -kohdassa, jolloin tiedostonimi ajetaan kaikkiin valittuihin kuviin.
12. Seuraavaksi käyttäjä nimeää 4 ylijäämäkuvaa, jotka liittyvät tähän juttuun.
13. Käyttäjä muuttaa loppuosaan KY-tunnuksen HY0914KY, sekä aloittaa ylijäämien laskuri-numeroinnin numerosta 05, koska se on seuraava ns. ”vapaa ylijäämänumero”. Käyttäjä laittaa rastin ”**Use incremental number**” ruutuun ja painaa **OK all –painiketta.**

Liite 3. Vaatimus 3 - Lokiraporttien saaminen kuvalatauksista

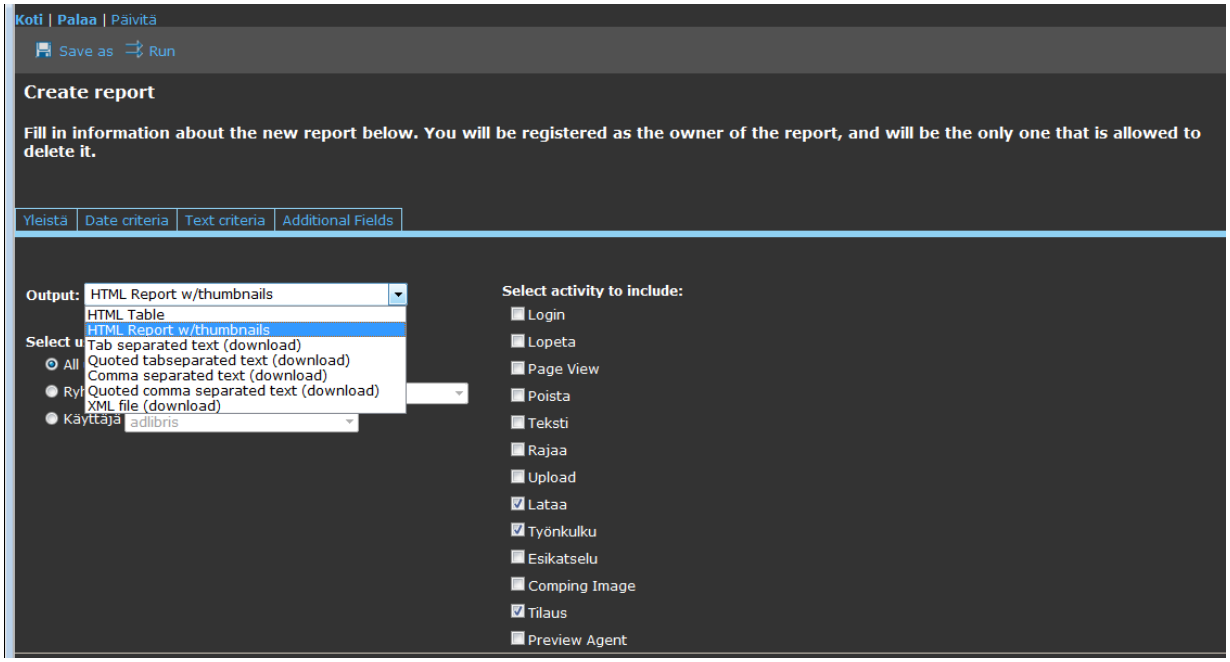
Raporttityökalua tarvitaan siihen, että saadaan käyttäjien kuvalataukset kuvapankista, jotta tiedetään mitä asiakasta laskutetaan mistäkin kuvasta.



Admin-käyttäjällä on mahdollisuus luoda valmiita raportteja, joita voi halutessaan myös muokata tai poistaa. Raporteissa on taustalla valittu latauksiin halutut asiat: users/downloads/pvm/additional fields to include in report.

Admin-käyttäjän luomia valmiita raporttityyppejä:

- Kaikki eiliset lataukset
- Eiliset lataukset suodatettuna
- Kaikki tämän päivän lataukset
- Tämän päivän lataukset suodatettuna



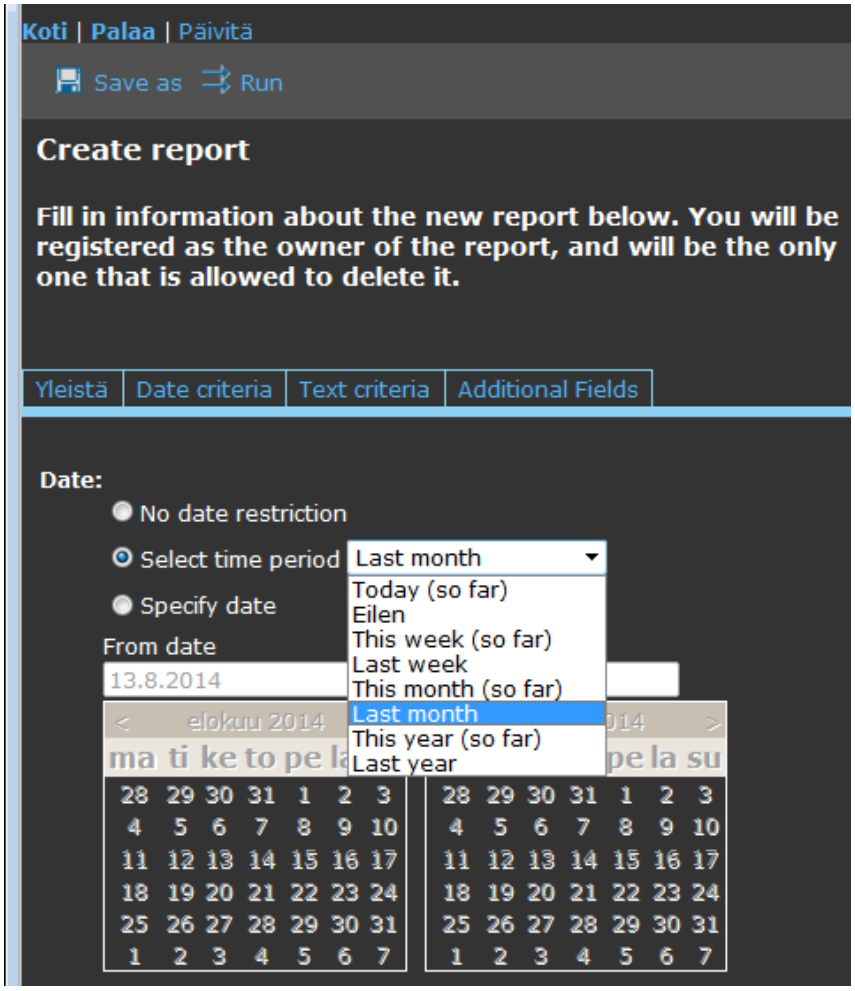
Yleistä-välilehden näkymä:

New Report /**Create Report** -toiminnolla raportti on mahdollista tulostaa mm. HTML-muodossa, jossa on sormenpääkuvat mukana, tekstinä tai XML-tiedostona.

Lataukset saadaan ajettua:

1. All users / Kaikkien käyttäjien osalta yhdellä kertaa (oletusvalinta)
2. Ryhmittäin
3. Yksittäisen käyttäjän osalta

Select activity to include / Ajettavalle raportille voidaan valita listalta halutut aktiviteetit. Käytetyimpiä aktiviteetteja ovat: lataa, työnkulku ja tilaus.

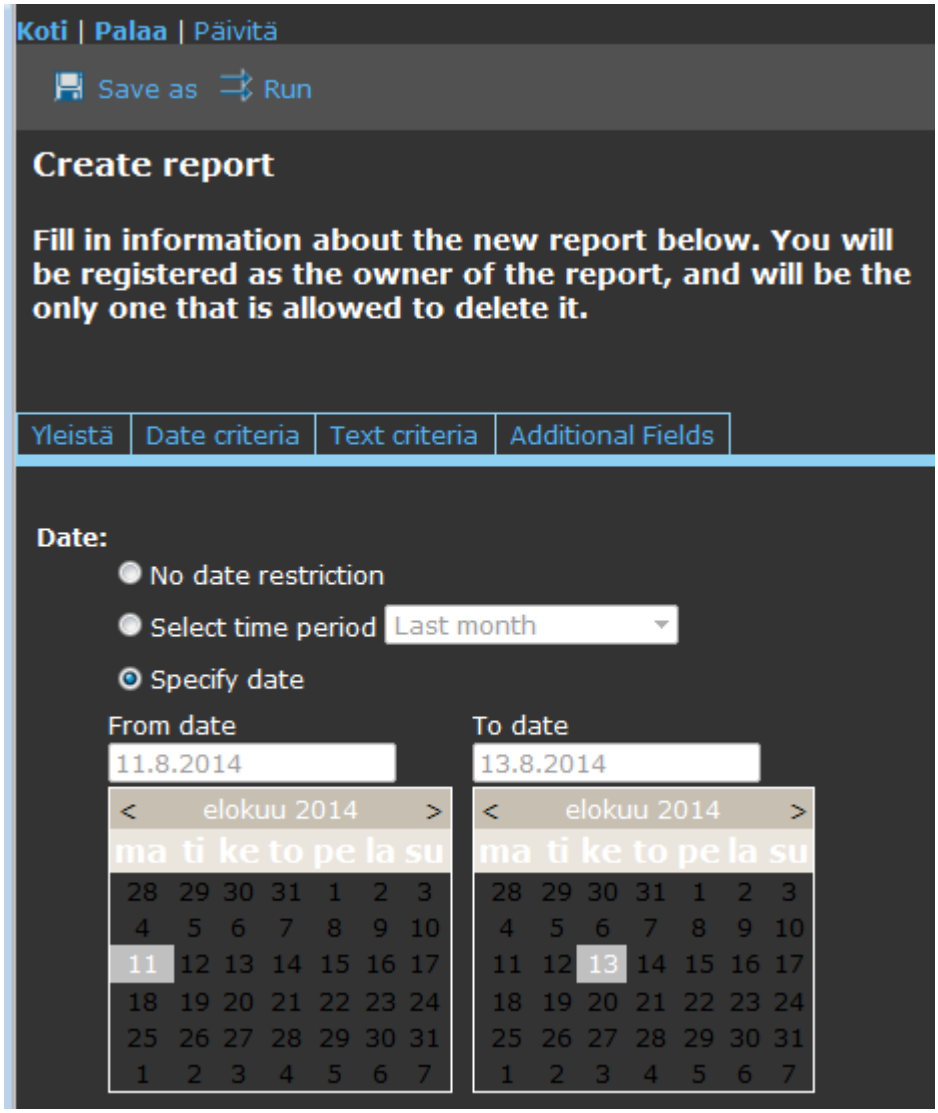


Date criteria -välilehden näkymä:

Raportti voidaan ajaa kolmen eri aikaperiodin mukaan:

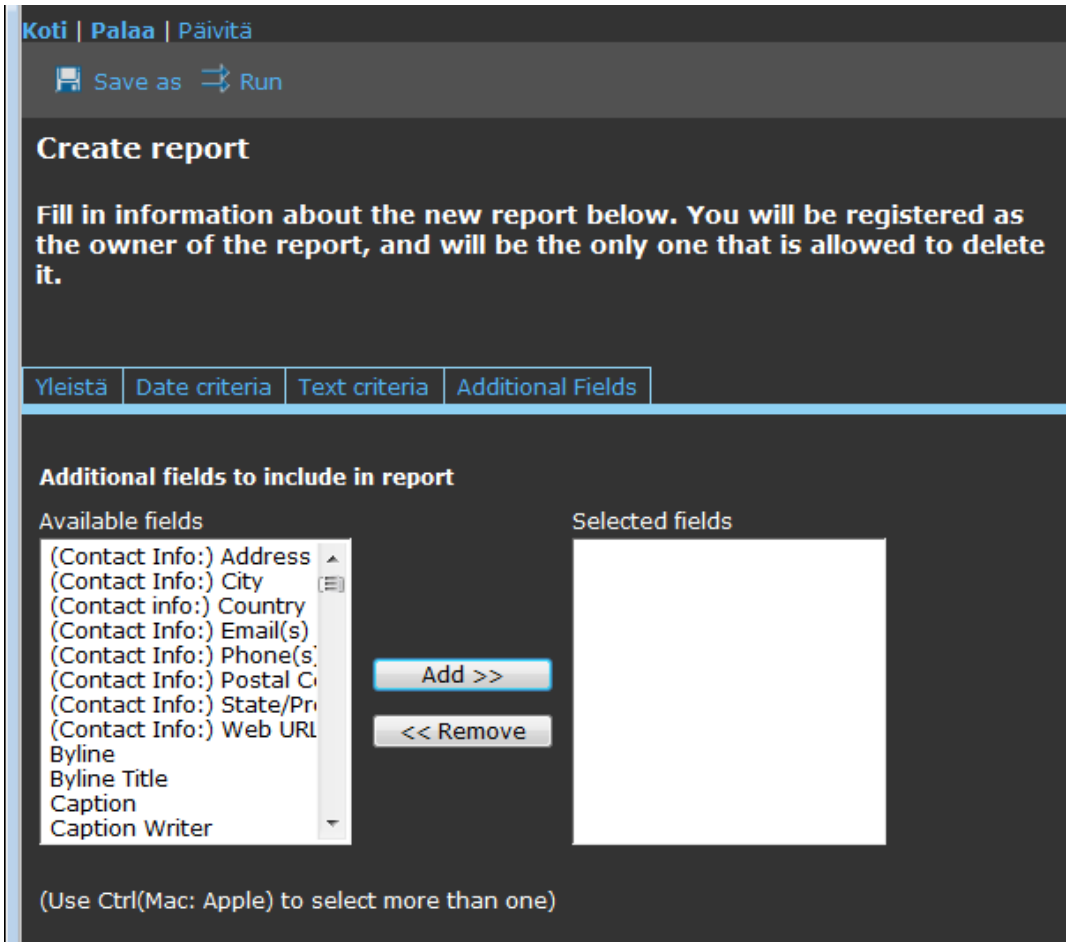
1. **No date restriction** / Päiväyksellä ei ole väliä
2. **Select time period** / Valitaan haluttu aikaperiodi, jossa on valittavissa kahdeksan oletusarvoa: tämä päivä (tästä ajanhetkestä eteenpäin), eilinen, tämä viikko, edellinen viikko, tämä kuukausi, edellinen kuukausi, tämä vuosi, edellinen vuosi
3. **Specify date** – From date → To date / Valitaan tietty päivä tai aikaväli kalenterista

Esimerkissä valittu vaihtoehto 2 **Select time period** → edellisen kuukauden osalta.



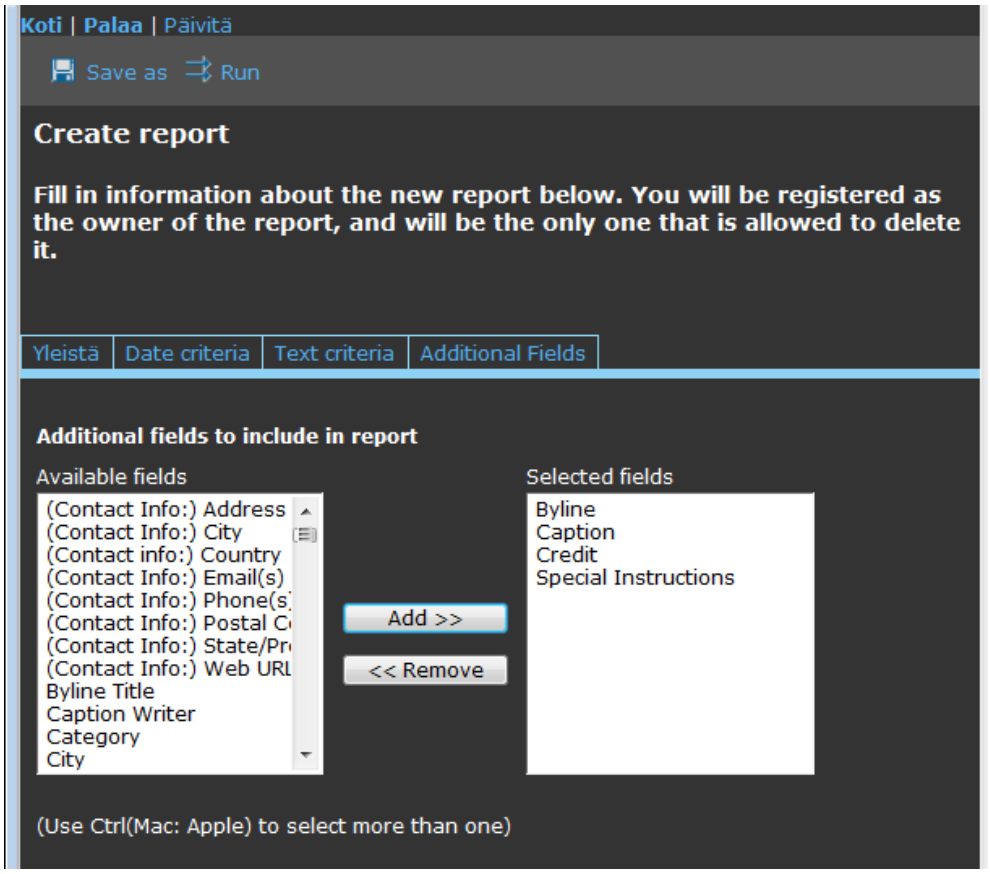
Date criteria -välilehden näkymä:

Esimerkissä valittu 3. vaihtoehto: **Specify date** – From date → To date / Valitaan kalenterista tietty aikaväli, jolta raportti ajetaan.

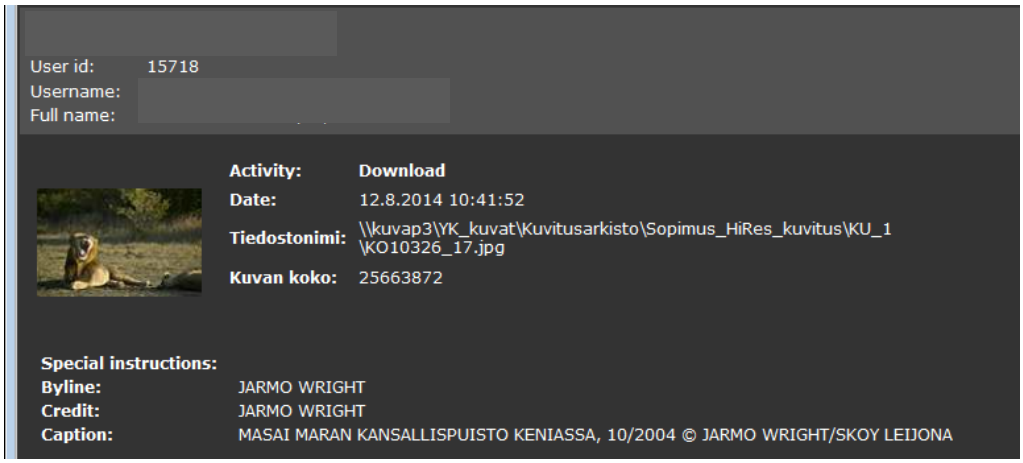


Additional fields -välilehden näkymä:

Raportille voidaan valita halutut kentät, jotka tulostetaan raporttiin. Kaksoisklikkaamalla **Available fields** -kenttää, saa halutun kentän siirtymään **Selected fields** –kenttään. Valittuja kenttiä voi myös halutessaan poistaa.



Esimerkissä raportille on valittu kuvaajan tiedot, kuvatekstin tiedot, kuvan oikeuksien tiedot ja kuvaan liittyvien erityisehtojen kenttä.

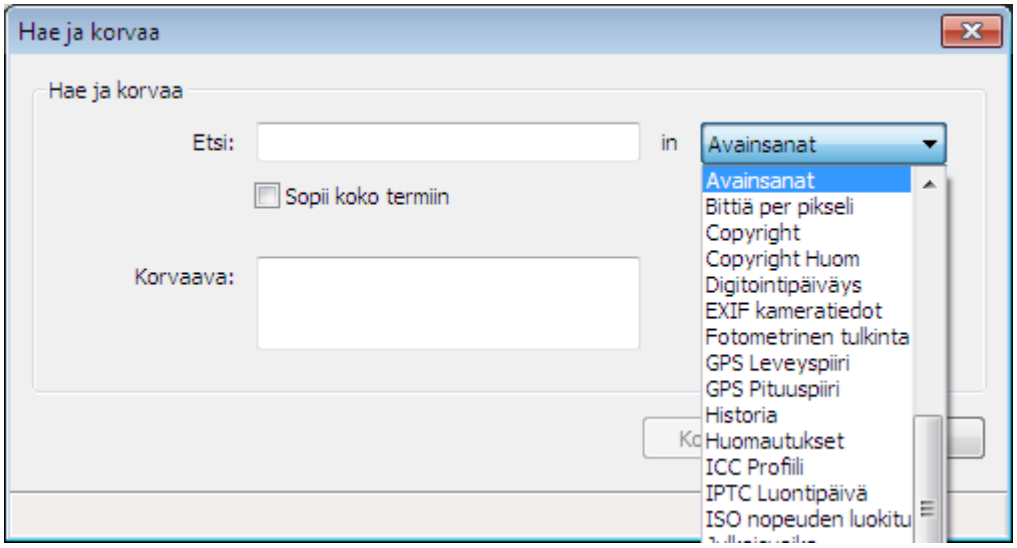


Seuraavaksi raportti ajetaan kaikilla edellä valituilla ominaisuuksilla painamalla →**Run** –painiketta.

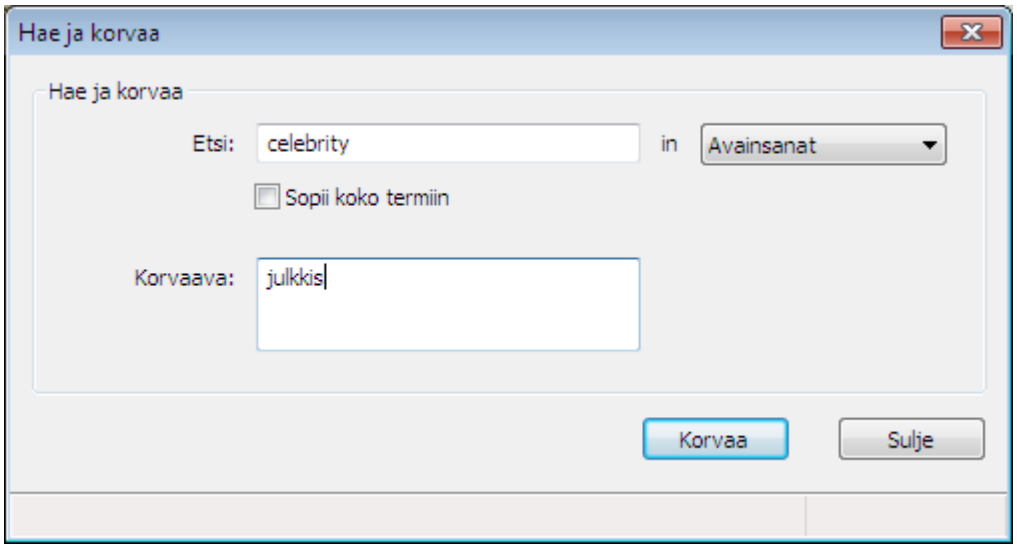
HTML-raporttiin tulostuu kuvan lataajan tiedot: user id, username, full name, tieto mihin lehteen kuvat on ladattu, jotta tietää, mitä lehteä/toimitusta laskutetaan ja lataukseen liittyvät muut tiedot, ladattujen kuvien sormenpääkuvat, kuviin liittyvät tiedot, jotka haluttiin tulostaa, otanta latauksista siltä aikaväliltä, joka raportille valittiin.

Raportin voi ajaa myös toiminnolla: **Run Tab separated text (download) report**
Raportti tulostetaan tesktimuodossa, jonka voi viedä exceliin, jos tarvitaan statistiikkaa eri käyttäjiin/ryhmiin/kuvaajiin/periodeihin liittyen.

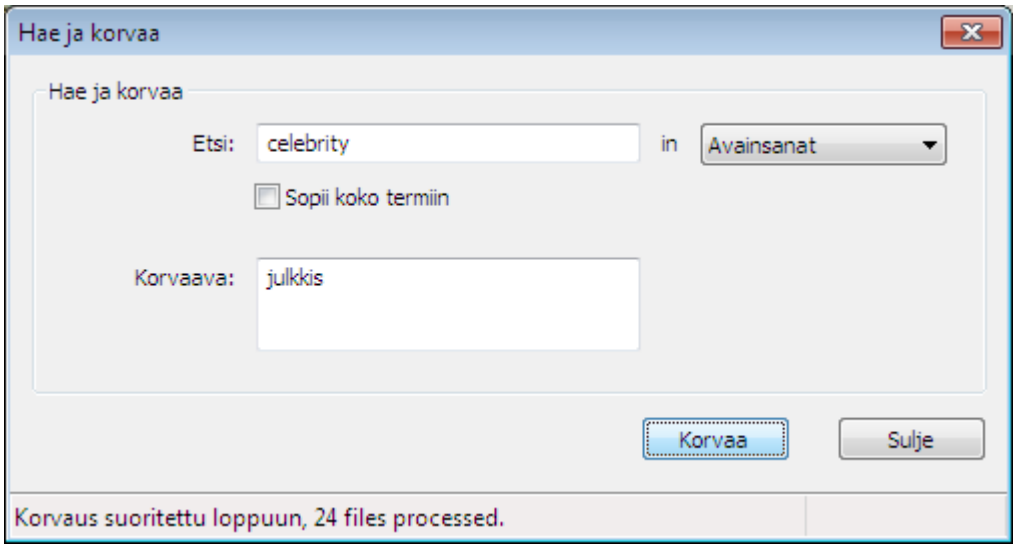
Liite 4. Vaatimus 5 - Metadatan Hae ja korvaa-toiminto



1. Käyttäjä kirjoittaa Etsi-kenttään halutun sanan tai kokonaisen lauseen, joka halutaan etsiä.
2. Käyttäjä valitsee alasvetovalikosta metadatakentän, johon etsittävä sana/teksti kohdistetaan.



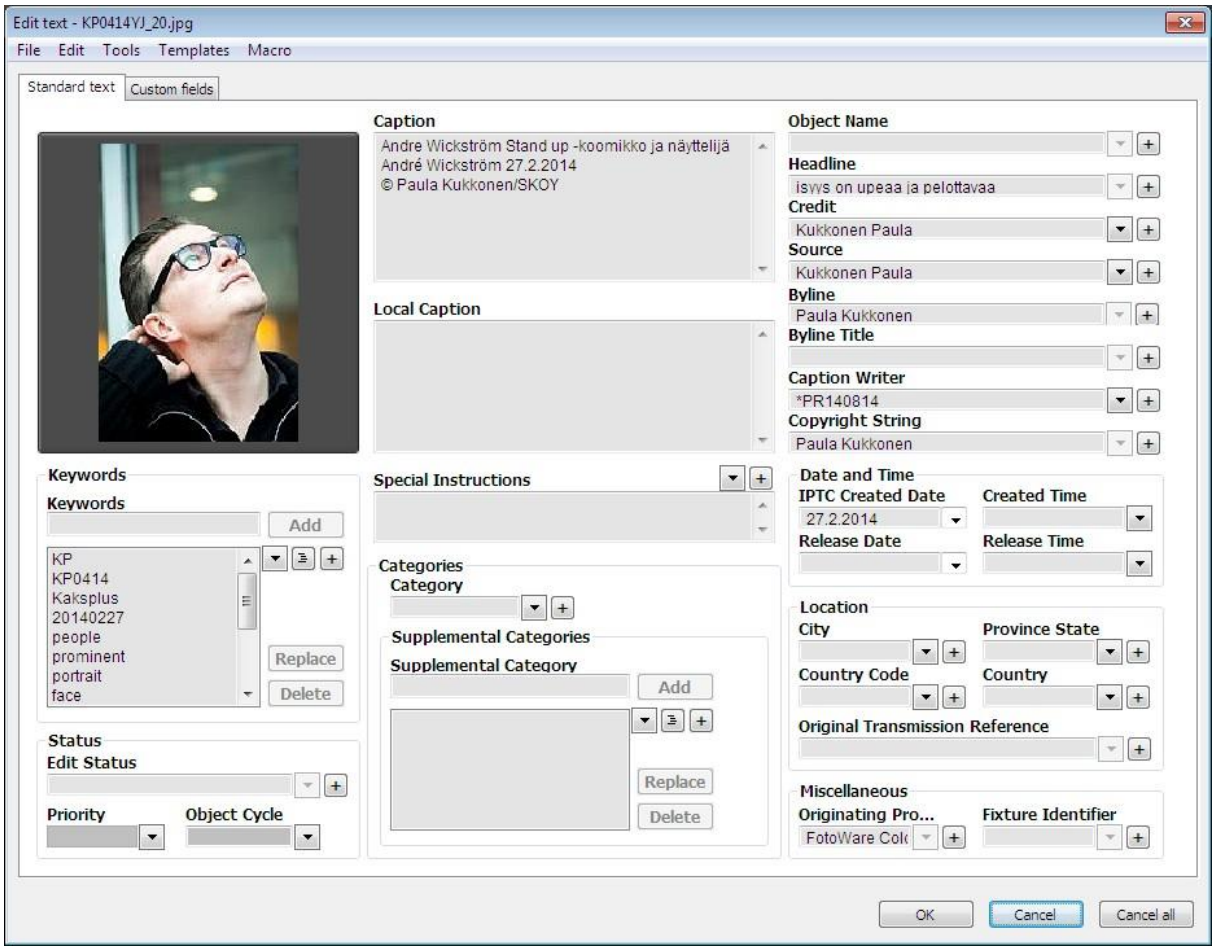
3. Käyttäjä kirjoittaa Korvaava-kenttään halutun sanan/lauseen, jolla etsittävä teksti korvataan ja painaa **Korvaa**-painiketta.



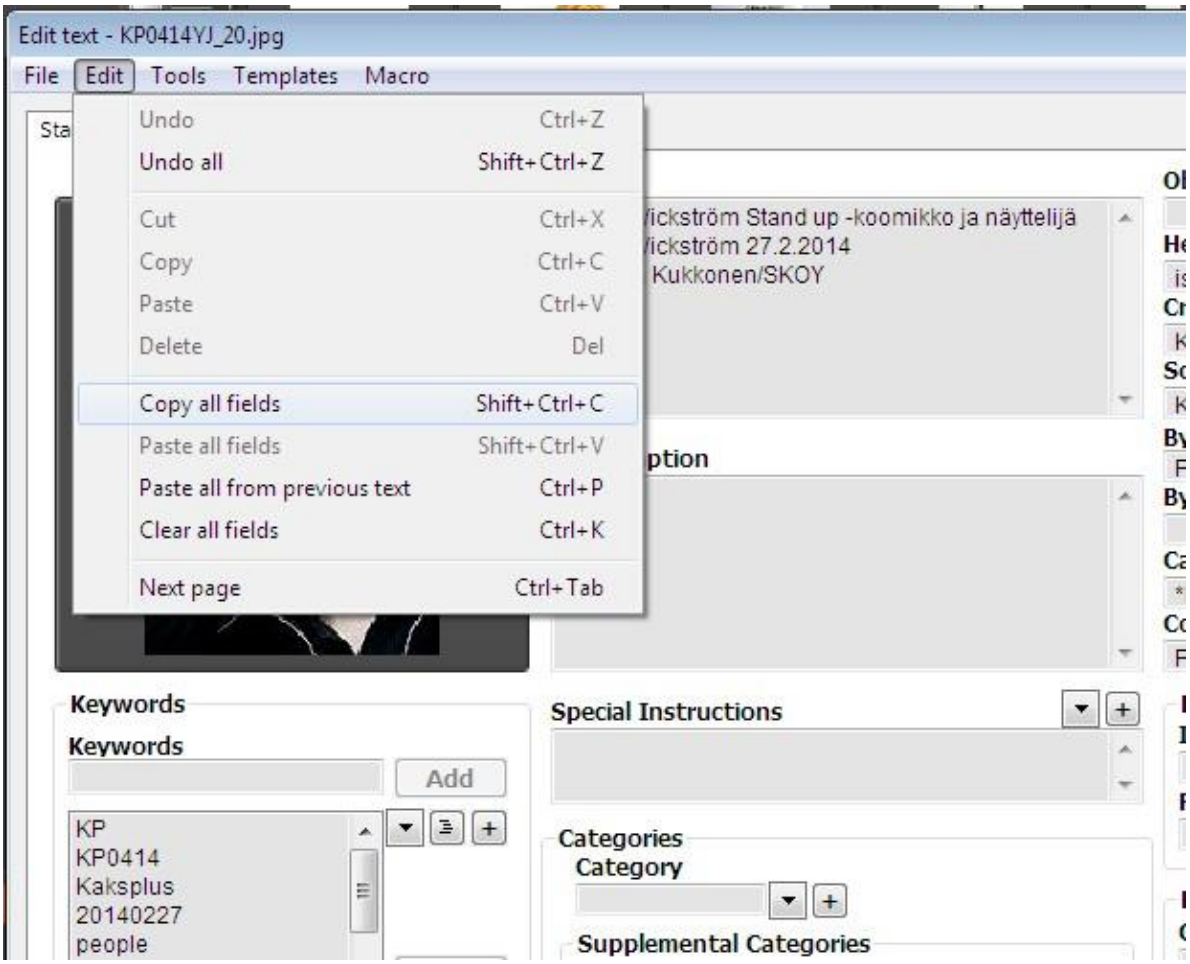
4. Ohjelma ilmoittaa lopuksi, montako korvausta on suoritettu.
5. Etsi ja korvaa -toiminnon voi kohdistaa samaan kenttään useaan kertaan.

Liite 5. Vaatimus 6 - Metadatan oletuspohjat

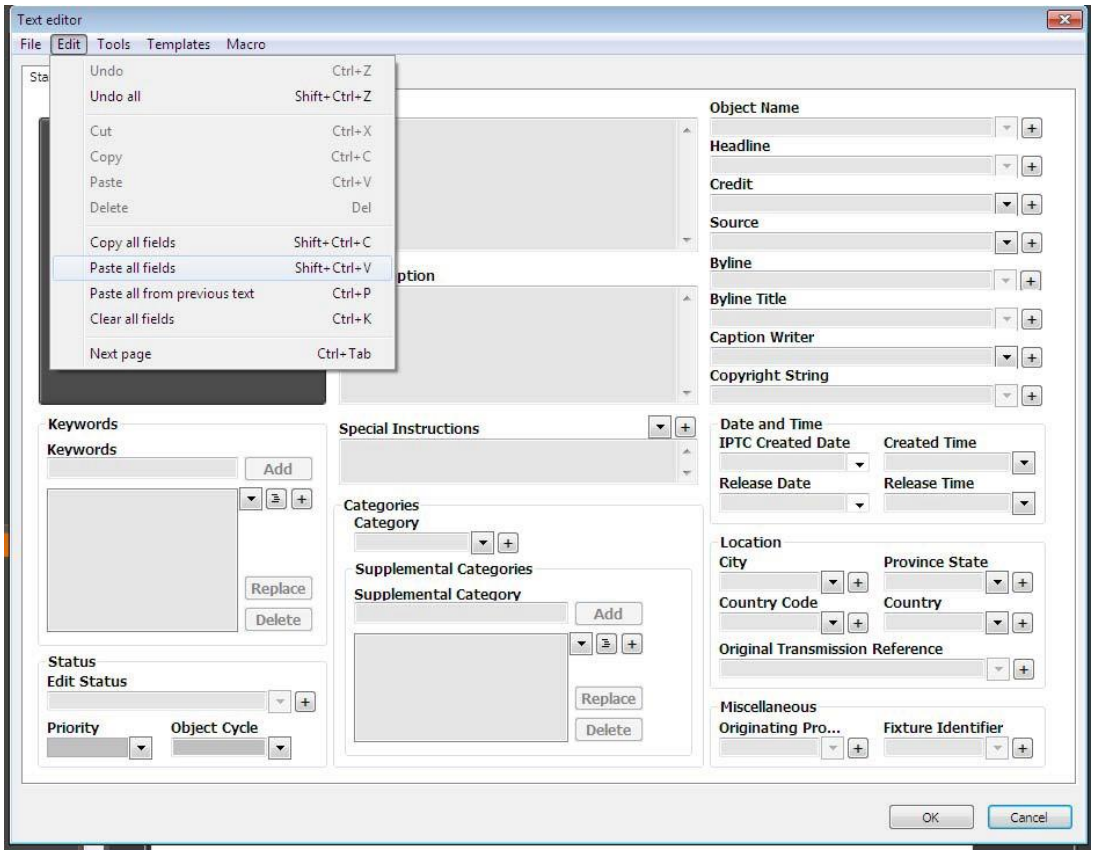
1. Käyttäjä aktivoi halutun kuvan, josta haluaa kopioida tekstiä toiseen kuvatiedostoon.



2. Käyttäjä avaa kuvan metadataikkunan, jossa näkyy kaikki kuvan metadata, sekä kuvan preview-tiedosto.



4. Käyttäjä kopioi kuvan kaikkien metadatakenttien tiedot leikepöydälle toiminnolla > **Edit** > **Copy all fields**
5. Käyttäjä aktivoi kuvat, joihin haluaa liittää äsken kopioimansa metadatan, ja valitsee toiminnon: **Metadata > Add to selected files**



5. Käyttäjälle aukeaa tekstieditori, jossa käyttäjä liittää äsken kopioimansa tekstipohjan uusiin kuviin toiminnolla **Edit > Paste all fields > OK**

6. **TAI** käyttäjä voi kirjoittaa tekstiä haluttuihin tyhjiin metadatakenttiin, joiden tiedot halutaan liittää useaan kuvaan yhtä aikaa **OK** –painikkeella.

Liite 6. Yhtiön vaatimusmäärittelydokumentin mallipohja

Vaatimusmäärittely

[Järjestelmä]

[Versio]

Tekijä	
Luontipäivä	
Viimeksi muuttanut	
Viimeisin muutospäivä	16.2.2017
Julkinen / Sisäinen / Luottamuksellinen / Salainen	

Versiohistoria

Versio	Päivämäärä	Henkilö	Muutokset	Tila
0.1			Alustava versio	
0.2				

Johdanto

Dokumentin tarkoitus

[Tässä kappaleessa kerrotaan lyhyesti seuraavat asiat:

Projektin nimi

Määrittelyn kohde

Mihin aineistoon määrittely perustuu (esimerkiksi liiketoimintayksiköltä saatu konseptin kuvaus, tms.)

Kenelle dokumentti on tarkoitettu]

Muutoksen tarkoitus

[Mitä järjestelmä yleisellä tasolla tekee?

Kuka/ketkä/missä yhteydessä järjestelmää käytetään ja mihin tarkoitukseen?]

Termit ja lyhenteet

Termi	**Selitys**

Viitteet ja liittyvät dokumentit

Viite1	Dokumentin täydellinen nimi, tekijä, versio, pvm, mistä dokumentti/viite löytyy
Viite2	Dokumentin täydellinen nimi, tekijä, versio, pvm, mistä dokumentti/viite löytyy

Yleiskuvaus

Toiminta

Yleinen yhteenveto järjestelmän toiminnasta.

Käyttäjät

Toimintaympäristö

[Tässä kappaleessa kuvataan laajempi kokonaisuus, johon järjestelmä liittyy. Tässä kappaleessa voidaan tarvittaessa kuvata myös laitteistoympäristö.]

Oletukset

[Kuvaa tässä oletukset, joiden pohjalta vaatimusmäärittely on tehty.]

Toiminnalliset vaatimukset

[Vaatimukset voidaan luokitella esimerkiksi seuraavan taulukon mukaisesti:

Luokittelu	**Symboli**	**Kuvaus**
Mandatory	**M**	Toiminnallisuus on välttämätön Vaiheen 1 lanseerauksessa. Lanseerausta ei voida tehdä ilman tätä ominaisuutta.
Important	**I**	Toiminnallisuus on erittäin tärkeä, mutta sen toteutus ei saa vaarantaa Vaiheen 1 lanseerausta. Ominaisuus on toteutettava viimeistään vaiheessa 2.
Wish	**W**	Kyseessä on nice-to-have – toiminnallisuus, voidaan toteuttaa mikäli aikaa jää.
Later	**L**	Vaatimus on syytä ottaa huomioon teknisen arkkitehtuurin suunnittelussa, mutta ominaisuuden toteutusta ei ole vielä aikataulutettu.

Vaatimus 1: Vaatimuksen 1 otsikko

Kuvaus: Vaatimuksen 1 tarkempi kuvaus

Perustelu: Perustelut vaatimukselle 1

Luokittelu: **(M)**]

Käyttötapaukset

[Luvun alussa mahdollisesti use case diagram ja sen jälkeen käyttötapausten tekstuaaliset kuvaukset. Käyttötapauksia voidaan täydentää toiminnallisen suunnittelun yhteydessä.

Käyttötapaus 1: Käyttötapauksen 1 nimi

Esiehdot

Järjestelmältä vaadittava tila, josta käyttötapaukseen tullaan.

Toimija

Toimijan rooli, esimerkiksi Käyttäjä

Laukaisija

Laukaisija tämän käyttötapauksen alkamiseen

Peruspolku

1. Ensimmäinen askel
2. Toinen askel

Poikkeukset

1a

Vaihtoehtoinen polku

1a

Säännöt

Related Business Rules

Jälkiehdot

Tilanne käyttötapauksen suorittamisen jälkeen, kertoo minkälaiseen tilanteeseen toimijat ovat käyttötapauksen jälkeen joutuneet]

Järjestelmän rakenne

Käyttöliittymät

[Tässä kappaleessa kuvataan vaatimukset käyttöliittymille.]

Ulkoiset liittymät

[Tässä kappaleessa kuvataan integraatiot tämän määrittelyn ulkopuolella oleviin järjestelmiin.]

Ei-toiminnalliset vaatimukset

[Tässä luvussa kuvataan järjestelmälle asetettavat laadulliset vaatimukset, joita voivat olla esimerkiksi:

- Suorituskyky: Vasteajat eri tyyppisille tehtäville / pyynnöille, tapahtumien määrä / aikayksikkö, yhtäaikaisten käyttäjien lukumäärä
- Käyttäjämäärät
- Vikasietoisuus
- Käyttäjäryhmät ja käyttöoikeudet (esimerkiksi käyttöoikeuksien tarkistaminen, salasanojen käsittely, käyttörajoitukset, henkilötietojen käsittely)
- Toiminnan jatkuvuus (esimerkiksi käyttökatkos max 24h, varmistukset otettavat kerran vrk:ssa, varajärjestelmä/varakone tarvitaan, lokikäsittely).
- Tietosuojaan liittyvät vaatimukset esimerkiksi tietoliikenteen ja tietojen salaamisen suhteen.
- Parametrointi ja ympäristön hallinta, mitä ohjelmiston tarjoamia palveluita on pystyttävä muuttamaan ilman ohjelmointia (esimerkiksi tiedostojen koon muuttaminen, päätteiden lisääminen, toimintojen lisääminen, käyttöliittymän virittäminen, kielen vaihtaminen
- Ylläpidettävyys, joustavuus, luotettavuus, siirrettävyys, tukitarpeen minimointi.]