SOVELLUSTEN YHTEENSOPIVUUS UUTEEN KÄYTTÖJÄRJESTELMÄÄN SIIRRYTTÄESSÄ

Case Oikeushallinnon tietotekniikkakeskus

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, 17.12.2012

Henri Lemetyinen

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Visamäki
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Henri Lemetyinen **Vuosi** 2012

Työn nimi Sovellusten yhteensopivuus uuteen käyttöjärjestelmään siirryttäessä - Case Oikeushallinnon tietotekniikkakeskus

TIIVISTELMÄ

Tämän opinnäytetyön toimeksiantajana toimi Oikeushallinnon tietotekniikkakeskus, OTTK. Työ tehtiin osana käyttöjärjestelmän vaihtoprojektia ja tarkoituksena oli tutkia Legacy-sovellusten yhteensopivuusongelmia siirryttäessä XP:stä Windows 7:aan. Uuden käyttöjärjestelmään mukanaan tuomia toimivuusongelmia tunnistettiin jo projektin alkuvaiheessa, joten ratkaisumallien suunnittelu oli tarpeen. Tavoitteena oli löytää ongelmiin parhaiten soveltuvat ratkaisuvaihtoehdot. Perinteisestä case-tutkimuksesta poiketen tarkoituksena oli tarjota näkökulmia myös toimeksiantajan ulkopuolisille tahoille. Tähän pyrittiin tarkastelemalla ongelmia ja käytettävissä olevia työkaluja yleisellä tasolla, sitomatta niitä suoraan yksittäisiin sovelluksiin.

Teoriaosuudessa käsiteltiin käyttöjärjestelmän vaihtoa ja miten tämä vaikutti sovellusten toimintaan sekä pohdittiin, mitä käyttöönotto on. Koska työ liittyi vahvasti käyttöjärjestelmäprojektin testausvaiheeseen, tarkasteltiin myös sovellustestausta ja sen hyödyntämistä OTTK:n tapauksessa. Jäljempänä tuotiin esiin käyttöjärjestelmän uudistamisesta johtuvia yhteensopivuusongelmia ja käytiin läpi eräitä toimivuusongelmien etsimiseen sekä korjaamiseen soveltuvia työkaluja.

Toimeksiantajan teettämää esiselvitystä sekä testauksesta saatuja tuloksia käytettiin toteutusvaiheessa käyttömallien suunnitteluun ja lopullisten toimenpide-esitysten tuottamiseen. Ongelmallisia sovelluksia tunnistettiin testausprojektin aikana neljä, joista kahdelle etsittiin soveltuvimmat käyttömallit.

Aikataulullisista syistä johtuen projektin ensimmäinen testausvaihe meni päällekkäin työaseman kehityksen kanssa, joten työasemamalliin voidaan joutua tekemään toimivuuteen vaikuttavia muutoksia vielä testauksen jälkeen. Vaikka todennäköisyys on pieni, tämä tulee ottaa huomioon tuloksia tarkasteltaessa.

Avainsanat Käyttöönotto, yhteensopivuus, testaus, Windows

Sivut 28 s. + liitteet 3 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Visamäki
Degree Programme in Business Information Technology
Systems work

Author Henri Lemetyinen **Year** 2012

Subject of Bachelor's thesis Application compatibility in operating system migration - Case Oikeushallinnon tietotekniikkakeskus

ABSTRACT

The commissioner of the thesis was Oikeushallinnon tietotekniikkakeksus, OTTK. This report was a part of a migration project where the commissioner replaced its old Windows XP operating system with new Windows 7. The main purpose was to research and apply applicable solutions for the compatibility problems caused by legacy software. Few compatibility issues were found right at the beginning of the project and this resulted in more detailed research within the testing phase. Deviated from traditional case study, this research aimed at providing value also for the third parties and not solely to the commissioner. This goal was accomplished examining compatibility fixes in general.

The theory part covered the operation system migration and how it affected the operation of the software. The theory part also included pondering about what software deployment is. Because the thesis was affiliated with the project's testing phase, it dealt with issues in software testing and its utilization in the OTTK's case. Certain tools made specifically for researching and solving compatibility issues were covered in the later chapters of theory.

Results of the research made before the project, were utilized when planning applicable compatibility fixes and lastly for creating the final statements. In total four programs with compatibility issues were found and the compatibility fixes were made for two of these programs.

The first testing phase overlapped with the desktop development due to the project's schedule. This means that some modifications may be needed after the testing process and these changes may affect software compatibility. Though the chances for this are minor, this should be considered when viewing the results.

Keywords Deployment, compatibility, software testing, legacy software

Pages 28 p. + appendices 3 p.

SANASTO

ACT
Application Compatibility Toolkit. Microsoftin julkaisema työkalu, joka mahdollistaa yhteensopivuusongelmien etsimisen sekä korjaamisen.

APP-V
Application Virtualization. Teknologia, jonka avulla sovelluksia voidaan virtualisoida ja hallita keskitetysti.

Legacy-sovellus
Ohjelma, joka on suunniteltu vanhan käyttöjärjestelmäarkkitehtuurin mukaan.

MDOP
Microsoft Desktop Optimization Pack. Kokoelma teknologioita sovellus- ja työpöytävirtualisointiin, järjestelmän diagnosointiin sekä verkkoympäristön hallintaan.

MED-V
Microsoft Enterprise Desktop Virtualization. Työpöydän virtualisointiin tehty ratkaisu, jossa virtuaalityöpöytä integroituu isäntäkäyttöjärjestelmään. Käytetään mm. yhteensopivuusongelmien ratkaisuna.

POC
Proof Of Concept. Järjestelmästä rakennettu yksinkertainen, yleensä karsittu kokoonpano. Käytetään, kun halutaan varmistua, että järjestelmä toimii ja täyttää sille asetetut vaatimukset.

UAC
User Account Control. Käyttäjätilien valvonta on teknologia, joka esiteltiin Windows Vistan mukana. UAC estää järjestelmään kohdistuvien muutosten tekemisen peruskäyttäjältä.

UE-V
User Experience Virtualization. Työympäristön virtualisointiratkaisu. Käyttäjälle tarjotaan tämän henkilökohtainen työpöytäympäristö riippumatta, mille fyysiselle laitteelle käyttäjä kirjautuu.

Sektori
Oikeushallinnonalan virastot on jaettu kymmeneen eri sektoriin. Täten eräiden sovellusten ja palveluiden dedikointi voidaan tehdä sektorikohtaisesti.

Shim
Shimmit ovat kirjastoja, jotka toimivat ohjelmointirajapinnassa muuttaen sovelluksen käyttöjärjestelmälle lähettämiä kutsuja tarkoituksena paikata eräitä yhteensopivuusongelmia.

SUA
Standard User Analyzer. Työkalu, jonka avulla voidaan havaita puutteellisista peruskäyttäjän oikeuksista johtuvat yhteensopivuusongelmat.

SISÄLLYS

1 JOHDANTO

Lähtökohta, jolle tämä opinnäytetyö perustuu, on toistuva trendi IT-alalla. Koska tietotekniikan ja tietojärjestelmien kehitys on jatkuvaa, jokainen järjestelmä tulee ennemmin tai myöhemmin elinkaarensa päähän ja sille joudutaan etsimään seuraaja. Varsinkin käyttöjärjestelmäpuolella kehitys on suurten julkaisutapahtumien ja niiden saaman mediahuomion ansiosta konkreettisimmin havaittavissa. Opinnäytetyössäkin käsiteltävien Windows käyttöjärjestelmien elinkaari on Microsoftin elinkaarikäytännön mukaan yrityskäytössä kymmenen vuoden pituinen, jonka jälkeen päivitys uuteen versioon on edessä. Voidaankin todeta, että mitä useammista osista organisaation IT-infrastruktuuri koostuu, sitä useammin eteen tulee tarve korvata käytöstä poistuvat sovellukset.

Uuden korvaavan tietojärjestelmän käyttöönotto vanhassa infrastruktuurissa sisältää aina riskin yhteensopivuusongelmista, mikä puolestaan korostaa testausprosessin merkitystä. Vaikka työssä käsiteltävät yksittäiset ohjelmistot vanhenevat ja vaihtelevat tapauskohtaisesti, opinnäytetyössä tarkasteltavat valintakriteerit ja työkalut ovat käyttökelpoisia myös muissa vastaavissa projekteissa. Tähän on pyritty tarkastelemalla useampia eri ratkaisumenetelmiä, sitomatta niitä yksittäiseen sovellukseen.

Opinnäytetyössä pyritään löytämään vastaukset seuraaviin tutkimuskysymyksiin:

- Miten sovellusten toimivuus voidaan kartoittaa tehokkaasti?
- Mitkä ovat toimintamenetelmät yhteensopivuusongelmien ratkaisemiseksi?
- Miksi valittuihin ratkaisuihin päädyttiin?

Oikeushallinnon tietotekniikkakeskus tuottaa ministeriölle ja hallinnonalan virastoille tietotekniikan kehittämis-, asiantuntija-, tuotanto-, hankinta- ja muita tukipalveluja (Oikeushallinnon tietotekniikkakeskus 2011). Tietotekniikkakeskuksen päätoimipiste on Hämeenlinnassa, jossa työskentelee noin 100 henkilöä. Ylläpidettäviä työasemia hallinnon alalla on noin 10 000 ja toimipisteitä 400. Koska hallinnonala koostuu lukuisista eri virastoista, myös käytössä olevia ohjelmia on runsaasti, eivätkä kaikki toimi ongelmitta päivitetyssä käyttöjärjestelmässä.

Opinnäytetyön aihe tulee toimeksiantajalla käynnissä olevasta projektista, jossa oikeusministeriön hallinnon alan Windows XP -työasemat päivitetään Windows 7 -käyttöjärjestelmään. Projektin esiselvityksessä on tullut esiin tieto, etteivät eräät käytössä olevat ohjelmat tule toimimaan uudessa ympäristössä. Ohjelmat testataan uudessa työympäristössä ja toimimattomuusongelmien laatu kartoitetaan. Testauksesta johdetut tulokset osoittavat tässä työssä käsiteltävät sovellukset. Tarkastelun ulkopuolelle jäävät oheislaitteiden sekä ajureiden yhteensopivuus.

2 OIKEUSHALLINNON KÄYTTÖJÄRJESTELMÄPROJEKTI

Tämä opinnäytetyö on osa meneillään olevaa käyttöjärjestelmäprojektia, mikä ohjaa suurelta osin aineiston keräämisprosessia. Toimeksiantaja on tehnyt uuteen käyttöjärjestelmään siirtymisestä esiselvityksen yhteistyössä asiantuntijan kanssa. Esiselvitys asettaa hyvin selkeät reunaehdot niin projektissa, kuin opinnäytetyössä käsiteltäville sovelluksille ja käyttöjärjestelmille.

2.1 Oikeushallinnon sovellukset

Yhteenlaskettuna koko hallinnon alalla on käytössä useampi sata sovellusta tai sovellusten eri versioita, jotka tulevat siirtymään uuteen käyttöjärjestelmään. Sovellukset on jaettu kolmeen eri ryhmään riippuen, ovatko ne hallinnon, ministeriön vai rikosseuraamusalan käytössä. Tämän lisäksi on jonkin verran yhteisessä käytössä olevia sovelluksia.

Koska ohjelmien määrä on suuri, kaikki sovellukset on priorisoitu asteikolla 1-3. Yksittäisen ohjelman sijoitus asteikolla riippuu siitä, kuinka laaja sovelluksen käyttäjäryhmä on, kuinka usein sovellusta käytetään ja kuinka kriittistä tehtävää sovelluksella suoritetaan, vrt. sähköpostin lähetys ja kuvankäsittely. On olemassa sovelluksia, jotka ovat käytössä vain yksittäisessä virastossa tai toimipisteessä. Tämän kaltaisten sovellusten prioriteetti on luonnollisesti matalampi kuin käyttäjämäärältään satojen tai tuhansien kokoisen sovelluksen.

Hallinnonalan virastot on jaettu kymmeneen eri sektoriin, mistä johtuen eräiden sovellusten tapauksessa puhutaan sektorikohtaisuudesta. Usein nämä sovellukset ovat sellaisia, joita kaikki kyseiseen sektoriin kuuluvat virastot käyttävät. Prioriteetiltaan ne ovat tällöin hyvin tärkeitä sektorin toiminnalle, mutta eivät välttämättä ole merkittäviä toiselle sektorille. Legacy-sovellusten osuus tällaisissa tapauksissa on suhteellisen pieni ja ne eivät muista sektorikohtaisista sovelluksista poiketen ole käytössä kaikissa virastoissa. Tästä syystä näiden ohjelmien prioriteetti ei ole yhtä korkea kuin suurissa sovelluksissa.

Käyttäjämäärältään suurimmat järjestelmät testataan erillisinä projekteina, eikä niiden toimivuuteen oteta kantaa Legacy-testaajien toimesta. Tällaiset sovellukset ovat usein niin suuria, ettei projektiryhmällä ole riittäviä resursseja niiden testaamiseen. Nämä järjestelmät koostuvat sovelluksesta ja siihen lisätyistä laajennoksista, joista tuotantokäytössä on lukuisia variaatioita. Erillisenä toteutettava testaus tuo mukanaan ajansäästöä ja vapauttaa käyttöjärjestelmäprojektin ydinhenkilöt kehitystyöhön.

2.2 Esiselvitys

Perinteisen case-tutkimuksen tarkoituksena on luoda tutkimustietoa työn kohteena olevasta järjestelmästä, mikä johtaa usein siihen, että tällainen tutkimusmenetelmä palvelee vain kohdeorganisaatiota. Opinnäytetyön tavoitteena on kuitenkin tarkastella yhteensopivuusongelmia ja niihin löy-

dettyjä ratkaisuja yleisellä tasolla, jossa niitä ei sidota vain yksittäiseen sovellukseen. Vaikka hallinnonalan infrastruktuuri kattaa yli 10 000 työasemaa, työkaluja käsiteltäessä pyritään huomioimaan niiden käyttökelpoisuus myös pienemmissä ympäristöissä. Tätä kautta pystytään palvelemaan toimeksiantajan lisäksi laajempaa kohderyhmää.

Esiselvityksessä ei oteta kantaa Legacy-sovellusten toimivuuteen, vaan tämän todetaan olevan helposti tarkastettavissa suoraan sovellustoimittajalta. Usein voidaan ajatella, että toimivuuden toteaminen on yksinkertaista, koska kriteerit ovat hyvin konkreettisia. Esimerkiksi: ”Käynnistyykö sovellus?”, ”Suoriutuuko se sille annetuista tehtävistä?” Kuitenkin sovellustoimittajan näkökulma ohjelmaa kohtaan on aina subjektiivinen, eikä toimivuuden tarkastelussa ole mahdollista ottaa huomioon asiakkaan infrastruktuurin vaikutusta. Käytännössä myös jokainen Legacy-sovellus testataan projektissa, jotta toimivuudesta voidaan varmistua.

Projektin aikataulun mukaisesti sovellustestausvaihe aloitetaan keväällä 2012. Lähes samanaikaisesti alkaa Legacy-sovellusten käyttömallien suunnittelu. Tämä on mahdollista testauksen luonteen ansiosta, koska suunnittelu pystytään aloittamaan välittömästi ensimmäisten ongelmatapausten ilmaantuessa. Sovellusmuutosten toteutusvaihe on ajoitettu välittömästi testauksen jälkeen, jolloin se alkaa suunnitteluvaiheen ollessa vielä käynnissä. Järjestelyllä saavutetaan ajallisia säästöjä ja se tuo lisävarmuutta muutosten toteuttamiseen, koska ajallisesti vaiheet osuvat peräkkäin ja testausvaiheen aikana hankittu tieto on välittömästi hyödynnettävissä. (Esiselvitys 2011, 21.)

2.3 Käyttöjärjestelmän päivitys

Tässä luvussa käyttöjärjestelmän päivityksellä tarkoitetaan kokonaan uuden version käyttöönottoa, joka on tarkasteltavan projektin tapauksessa siirtyminen Windows XP -käyttöjärjestelmästä Windows 7:ään. Käyttöjärjestelmän päivitys tulee ajankohtaiseksi yrityksissä viimeistään siinä vaiheessa, kun vanhan käyttöjärjestelmän tuki lopetetaan palveluntarjoajan toimesta. Syitä uuden version käyttöönottoon ovat usein järjestelmän tietoturvallisuuden takaaminen sekä uusien sovellusten tai toimintamenetelmien asettamat tekniset vaatimukset. Oikeushallinnon tapauksessa päätös uuteen käyttöjärjestelmään siirtymisestä annettiin XP:n tuen loppumisen sekä sitä kautta syntyvien tietoturvauhkien takia. (Esiselvitys 2011, 10–11.)

Koska käyttöjärjestelmän vaihtaminen koskee kaikkia 10 000 henkilöä, joilla on työasema, tietoteknisen osaamisen taso vaihtelee henkilöistä jotka suorittavat substanssiosaamistaan vain totutulla tavalla aina tietotekniikan ammattilaisiin. Ensimmäisenä mainituilla henkilöillä voi esiintyä herkästi muutosvastarintaa tuntemattomaan järjestelmään siirryttäessä ja tämä olisikin otettava huomioon muun muassa järjestämällä tarvittavaa koulutusta halukkaille. Pohjonen (2002, 50) listaa muutosvastarinnan syiksi muun muassa inhimillisen uusien asioiden vierastamisen, pelon omien työtehtävien menettämisestä sekä uusien toimintatapojen opettelemisesta. Vaihtelevasta osaamistasosta johtuen loppukäyttäjien koulutuksen tarve on otettu huomioon jo projektin esiselvitysvaiheessa.

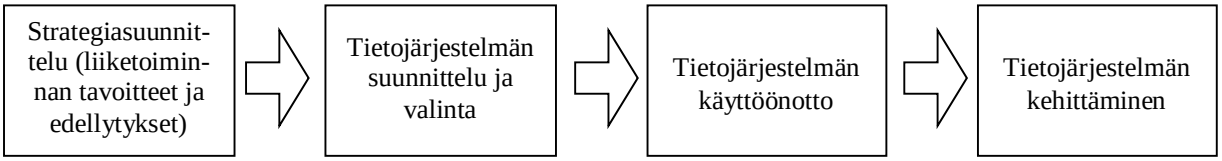
Nykyisen käyttöjärjestelmän korvaajaa valitessa huomioitiin myös muut käyttöjärjestelmäalustat, mutta niiden todettiin aiheuttavan huomattavia kustannuksia sovellusten yhteen toimivuuden kannalta. Alustan vaihtoa ei täten pidetty toteuttamiskelpoisena vaihtoehtona. Lopulliseen vertailuun valikoituivat Windows 7 pro sekä enterprise versiot, joista jälkimmäisen todettiin olevan kustannustehokkain ratkaisu pitkällä aikavälillä. Lisäksi enterprise versioon kuuluvien ominaisuuksien, kuten tiedostojen salaamisen sekä etäyhteysratkaisun katsottiin tuovan lisäarvoa, mikä osaltaan vaikutti version valintaan.

Hallinnonalan käyttämät laitteet ovat leasing-sopimuksella vuokrattuja ja keskimääräinen käyttöaika yksittäisellä työasemalla on kolmesta viiteen vuotta. Koska laitekanta on ajanmukaista tai uusiutumassa, oli relevanttia pohtia versionvalinnan aikana myös käyttöjärjestelmän bittisyyttä. Konkreettisin 64-bittisen käyttöjärjestelmän valinnalla saavutettava hyöty olisi laitteiston ram-muistimäärän hyödyntäminen. 32-bittinen Windows 7 käyttöjärjestelmä pystyy käsittelemään vain 4 gigatavua käyttömuistia verrattuna 64-bittiseen järjestelmään, jossa raja tulee vastaan 192 gigatavun kohdalla. Tällä hetkellä uusimmissa hallinnonalan leasing-koneissa fyysisen muistin määrä on 4 Gt, joten raja ei tulisi vielä vastaan. Ottaen kuitenkin huomioon käyttöönotettavan Windows 7 kymmenen vuoden mittaisen elinkaaren, laitekantaa tullaan varmuudella uusimaan ja suuntautuminen tulee kohdistumaan tehokkaampaan laitteistoon. 64-bittisen käyttöjärjestelmän valintaa vastaan olivat erään sovelluksen toimimattomuus tässä versiossa sekä mahdolliset ongelmat vanhempien oheislaitteiden ajureiden kohdalla. (Esiselvitys 2011, 10.)

Windows 8 oli mukana esiselvitysvaiheessa ja eräs hyöty uusimman käyttöjärjestelmän valinnasta olisikin ollut sen pitkä elinkaari. Esitys käyttöjärjestelmän valinnasta on tehty kuitenkin jo syys–marraskuussa 2011, jolloin käyttöjärjestelmää ei ollut vielä julkaistu kuin testikäyttöön ja lopullisen työpöytäversion toiminnallisuuksista ei ollut täyttä varmuutta. Uudistetun käyttöliittymän katsottiin osaltaan lisäävän riskiä sovellusten yhteensopivuuden kanssa, eikä testaukseen tarvittava työ vähenisi tässäkään siirtymässä. Suurin este 7:n yli hyppäämiselle oli se, ettei sovellusten testaukselle jäisi riittävästi aikaa muun muassa Windows 8 myöhäisen julkaisuajankohdan takia. (Esiselvitys 2011, 23.)

3 KÄYTTÖÖNOTTOPROJEKTI

Uuden tietojärjestelmän hankinta voi lähteä liikkeelle niin organisaation sisäisestä tarpeesta kehittää toimintaansa kuin ulkoisista tekijöistä, esimerkiksi järjestelmän vanhenemisesta. Sisäisten tekijöiden taustalla on usein tarve tehostaa toimintaa ja saavuttaa sitä kautta taloudellisia säästöjä. Projektin alkuvaiheessa asetetut tavoitteet on syytä miettiä tarkoin, sillä tehokkuutta arvioitaessa näitä tavoitteita tullaan vertaamaan siihen, miten uusi järjestelmä todellisuudessa toimii tuotantokäytössä. Kuvassa yksi on esitetty käyttöön otettavan tietojärjestelmän elinkaari aina suunnittelusta ylläpitovaiheeseen.
(Nurminen, Reijonen, Vuorenheimo 2002, 3)

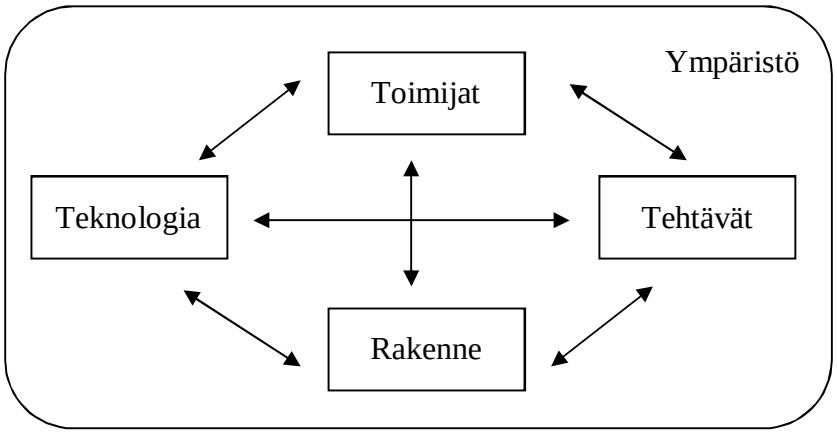


Kuva 1. Tietojärjestelmän käyttöönottoprosessi käyttäjäorganisaation näkökulmasta

Tietojärjestelmän kehittämisen vaiheista vaatimusmäärittelyn tekeminen on erityisen merkittävää. Jopa 75 % kaikista kehittämisprojekteista on arvioitu epäonnistuvan ja jokaisessa näistä projekteista vaatimusmäärittely on osoittautunut puutteelliseksi. (JUHTA - Julkisen hallinnon tietohallinnon neuvottelukunta 2009, 9.)

Syitä yksittäisen käyttöönottoprojektin epäonnistumiseen pystytään helposti listaamaan, mutta tällaisten syiden vaikuttavuus on hyvin tapauskohtaista. Toimintaympäristö sekä -tavat vaihtelevat organisaatiokohtaisesti, joten syy, joka vaikutti epäonnistumiseen toisessa organisaatiossa, ei välttämättä ole yhtä kriittinen tekijä toisessa ympäristössä. Tästä syystä epäonnistumisista kerätyn tiedon hyödynnettävyys muissa projekteissa on kyseenalaista. (Nurminen ym. 2002, 5–6.)

Uuden järjestelmän käyttöönotto vaikuttaa aina useaan osatekijään koko organisaation laajuisesti. Scottin (1987) edelleen kehittämä kuvaus Leavittin timantista (kuva 2) on malli, josta voidaan nähdä neljän osatekijän: rakenne, tehtävät, teknologia ja toimijat olevan vuorovaikutuksessa keskenään. Scottin lisäämä muutos on viides muuttuja, ympäristö, joka asettaa reunaehdot kaikille neljälle osatekijälle.



Kuva 2. Edelleen kehitelty versio Leavittin timantista.

Jos kaaviota tarkastellaan projektissa toteutettavan käyttöönoton näkökulmasta, huomataan ympäristön muodostavan kriteerit teknologian valitsemiselle, laitteistolle sekä sovelluksille. Uusi käyttöjärjestelmä puolestaan vaikuttaa niin työtehtävien suoritustapaan kuin tehtävien suorittajiin. Toisaalta kokonaan uusi teknologia voi vaikuttaa tehtävien rakenteeseen tai työnjakoon. (Nurminen ym. 2002, 5–6.)

3.1 Sovelluksen käyttöönotto

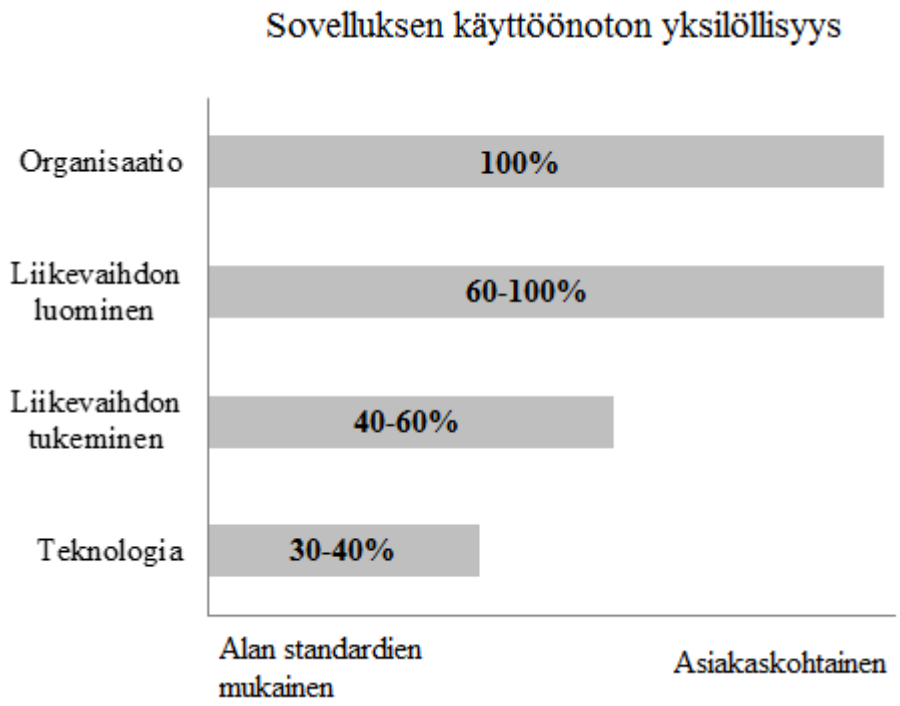
Beaubouef (2009, 11–12) puhuu kirjassaan liiketoimintaratkaisusta, joka koostuu kolmesta tekijästä: teknologiasta, liiketoimintaprosessista sekä ihmisistä. Kun uutta ratkaisua lähdetään ottamaan käyttöön, edellä mainituista kolmesta tekijästä merkittävimpään rooliin nousee ihmiset. Mikäli organisaation työntekijät eivät ole sitoutuneet uuden järjestelmän käyttöönottoon, on hyvin todennäköistä, että muutos tulee epäonnistumaan. Nurminen ym. (2002, 2) mainitsevatkin yhdeksi merkittäväksi käyttöönoton muuttujaksi johdon sitoutumisen. Ongelmana kuitenkin on käsitteen määrittely, se miten sitoutuneisuutta mitataan projektin aikana.

Eräs Beaubouef:n (2009, 15) korostamista asioista on sovelluksen käsitteleminen osana koko liiketoimintaratkaisua. Tällöin onnistuneesta käyttöönotosta ei voida puhua vielä siinä vaiheessa, kun sovellus on asennettu testaus- tai tuotantoympäristöön ja se toimii teknisesti. Käyttöönotto on onnistunut vasta silloin, kun edellä mainittujen kriteerien lisäksi sovellus integroituu osaksi liiketoimintaprosessia. Ongelma tulee esiin varsinkin kaupallisten sovellusten osalla. Jokaisen organisaation liiketoimintaprosessit sekä IT-infrastruktuuri eroavat toisten organisaatioiden vastaavista. Kaupallisen sovelluksen kehittäjä joutuu tämän takia pohtimaan kysymystä, miten sovellus saadaan tukemaan mahdollisimman montaa erilaista infrastruktuuria. Ratkaistakseen tämän ongelman, toimittajat usein keskittyvät vain sovelluksen teknologiaan, jolloin liiketoimintaprosessit sekä organisaatio jäävät huomiotta. Ratkaisu on toki perusteltua, kaupallisella sovelluskehittäjällä kun ei ole mahdollisuutta keskittyä yksittäiseen infrastruktuuriin tai liiketoimintamalliin.

3.2 Käyttöönoton eri näkökulmia

Beaubouef (2009, 16) on laatinut suuntaa antavan taulukon (taulukko 1), jossa käsitellään kaupallisten sovellusten käyttöönottoa siitä näkökulmasta, millä osa-alueilla käyttöönottoprojektit ovat yksilöllisiä ja mihin alueisiin voidaan soveltaa valmiita standardeja.

Taulukko 1. Käyttöönottoprojektin yksilöllisyys liiketoimintaprosessin eri osa-alueilla



Jakauma sisältää kolme aiemmin mainittua tekijää, joista ihmiset eli kohteena oleva organisaatio on jokaisessa projektissa yksilöllinen. Liiketoimintaprosessit voivat olla jo joiltain osin yleistettävissä. On kuitenkin yrityskohtaista, miten yksilöllisiä nämä prosessit ovat. Yleistettävin tekijä on teknologia, josta hieman yli puolet voidaan toteuttaa alan standardien mukaisilla menetelmillä. Tämä onkin ratkaiseva tekijä sille, miksi kaupalliset sovellustoimittajat keskittyvät pääasiassa teknologiaan eivätkä pyri tarjoamaan yksilöityjä ratkaisuja.

Vastakohtana kaupallisten sovellusten käyttöönotolle voidaan pitää ohjelmistotoimittajalta tilattua, tiettyyn ympäristöön dedikoitua sovellusta. Tällöin kehittäjä tuntee tarkasti, minkä liiketoimintaprosessin osana sovellus tulee toimimaan ja pystyy keskittymään laajempaan kokonaisuuteen. Mitä enemmän taustalla olevasta infrastruktuurista tiedetään, sitä helpompi integroinnista syntyviä yhteensopivuusongelmia on välttää.

Organisaation liiketoimintaprosessien nykytila on huomioitava uutta sovellusta käyttöönotettaessa. Vaikka kaupallinen sovellus voidaan hankkia vain yhtä prosessia tehostamaan, se saattaa tarjota mahdollisuuden myös muiden toimintamallien muuttamiseksi. Asiakkaan näkökulmasta voidaan helposti ajatella, että hankinnasta on saatava ns. kaikki irti, eli paras mahdollinen hyöty. Tämä tarkoittaa usein kaikkien mahdollisten toiminnalli-

suuksien hyödyntämistä. Ongelmaksi tällaisessa ajattelumallissa muodostuu alkuperäinen suunnitelma. Hyvässäkään suunnitelmassa ei pystytä huomioimaan lennosta mukaan otettujen toiminnallisuuksien vaikutusta lopputulokseen, eikä organisaatiolla ole mahdollisuutta varautua alkuperäistä laajempiin muutoksiin. Käytännön esimerkiksi sopii tässä tapauksessa toimisto-ohjelmisto. Yritys hankkii uuden taulukkolaskentaohjelman ja on varautunut yksinomaan tähän muutokseen. Lisenssin mukana tulee mahdollisuus vaihtaa myös tekstinkäsittelysovellus ja tämä mahdollisuus hyödynnetään. Myöhemmin eteen tulee ongelma, koska tekstidokumentit eivät aukea sovelluksessa oikein. Muutos aiheuttaa huomattavaa lisätyötä yhteensopivuuden korjaamiseksi. Vaarana on hukata muutoksella aikaansaatu tehokkuus uusien ongelmien ratkomiseen. (Beaubouef 2009, 22.)

4 SOVELLUSTEN TOIMIVUUDEN TESTAUS

Sovellustestauksen lähtökohtana on testata ohjelman toimintaa kaikilla mahdollisilla syötteillä tai niiden variaatioilla, joita sovellukselle voidaan antaa. Käytäntö kuitenkin osoittaa perusteellisen testaustavan olevan mahdoton toteuttaa, koska yksinkertaisessakin sovelluksessa kaikkien mahdollisten yhdistelmien läpikäynti vie loputtomasti aikaa. Sen mistä näkökulmasta testausta lähestytään, ei tulisi vaikuttaa lopputulokseen, mutta saattaa tehostaa prosessia. Myers, Badgett, Thomas & Sandler (2004, 6) kirjoittavat sovellustestauksen olevan prosessi, jonka tarkoituksena on todistaa, ettei testattava ohjelma toimi. Kirjallisuudessa testauksen määrittelystä tulee eteen hyvin usein myös vastakkainen lähestymistapa: tarkoituksena on todistaa sovelluksen toimivuus. Ensimmäistä tapaa voidaan pitää lähtökohtaisesti tehokkaampana, koska tällöin taustalla on ajatus löytää sovelluksesta virheitä sen sijaan, että yritettäisiin todistaa, ettei niitä ole. Tätä päätelmää tukee myös psykologinen näkökulma; testaajan asennoituminen testausprosessiin. Mikäli taustalla olevana ajatuksena on virheiden löytäminen, henkilö valitsee usein sellaiset testitapaukset, joiden avulla se on todennäköisempää. Vastakohtaisessa ajattelumallissa henkilö valitsee mieluummin testitapaukset, jotka todennäköisemmin todistavat toimivuuden, kun päättyvät virhetilanteisiin.

Ero onnistuneiden ja epäonnistuneiden testitapausten tulkinnassa riippuu vastaavasti valitusta näkökulmasta. Hyvin helposti ajatellaan, että testitapaus on onnistunut, kun sen avulla ei löydetä virhettä. Toisaalta testitapaus kategorioidaan epäonnistuneeksi silloin, kun virhetilanne löytyy. Jälleen kyse on tulkintatavasta, jonka voidaan ajatella olevan psykologisesti harhaanjohtava. Testausprojektia ei voida kutsua onnistuneeksi sillä perusteella, ettei sovelluksesta löytynyt virheitä. Projekti on onnistunut vasta, kun virheet on löydetty, koska virheetöntä sovellusta ei oletusarvoisesti ole olemassa. Oikeushallinnon tietotekniikkakeskuksen projektissa testilomakkeeseen kirjataan hylätty tai hyväksytty riippuen, toimiiko ohjelma ongelmitta uudessa käyttöjärjestelmässä. Tällöin yksittäisten testitapausten tulkintatapa ei tule esiin projektissa, vaan lähestymistapa on mahdollisimman neutraali, alkuperäisen kysymyksen ollessa toimiiko sovellus vai ei.

Jokaiselle testitapaukselle tulee kirjata siitä odotettava lopputulos. Mikäli tavoiteltavaa tulosta ei aseteta etukäteen, testitapauksen luotettavuus on kyseenalainen. Koska tavoitteena on saavuttaa jokin ennalta määrittelemätön lopputulos, epäselvissä tapauksissa riski virheelliseen tai hätäiseen tulkintaan kasvaa. Kun lopputulos on ennalta määritelty, testitapaus on helppo luokitella onnistuneeksi (Myers ym. 2004, 14–15). OTTK:n testausprojektissa ei ole määritelty ennalta odotettavia lopputuloksia, koska yhteinen testauslomake ei anna mahdollisuutta sovelluskohtaisten testitapausten tekemiseen. Riippuen yksittäisistä testitapauksista, sovelluksen hyväksyminen tai hylkääminen on testaajan harkinnan varaista.

Sovelluksen kehittäjän ei tulisi suorittaa testausta, koska ongelmana on liian subjektiivinen suhtautuminen ohjelmaa kohtaan. Sovelluskehittäjä voi tulla sokeaksi omille virheilleen, koska tietää jo etukäteen, mitä ohjelman tulee tehdä ja miten testauksessa tavoiteltaviin lopputuloksiin päästään. Liiallisen subjektiivisuuden ongelma on myös psykologinen. Kun kehittämiseen on käytetty riittävästi aikaa, omien virheiden löytäminen ei ole miellyttävää ja testausprosessi voi helposti muuttua toimivuuden todisteluksi sen sijaan, että virheitä etsittäisiin aktiivisesti. Kun kehittäjä-käsitettä tarkastellaan laajemmin, puhutaan koko organisaatiosta. Organisaatio on vastuussa tekijöistä, kuten käytössä oleva aika ja resurssit. Mitä pidemmäksi sovellusprojekti venyy, sitä enemmän se lisää kustannuksia. Testauksen aikana saatetaan vältellä mahdollisia ongelmakohtia, koska niiden korjaaminen vaatii sekä aikaa että rahaa. (Myers ym. 2004, 16–17.)

Testitapauksia kirjoitettaessa ei tule käsitellä vain sellaisia syötteitä, joiden oletetaan olevan oikeita, vaan testattavana tulee olla myös virheellisillä syötteillä tehdyt tapaukset. Tällaiset testitapaukset paljastavat usein sovellusvirheitä, joiden olemassaoloa ei muuten saataisi selvitettyä. Virheelliset syötteet auttavat lisäksi selvittämään, ettei ohjelma tee jotakin, mitä sen ei pitäisi tehdä. (Myers ym. 2004, 17–18.)

Riippumatta lähestymistavasta, jota testausprojektissa käytetään, löydetyt virheet dokumentoidaan ja raportoidaan sovelluksen kehittäjälle, joka korjaa ongelmatapaukset. Korjausprosessin jälkeen tarkistetaan tehtyjen toimenpiteiden vaikutus, jotta voidaan varmistua siitä, ettei korjaus aiheuttanut uutta ongelmaa.

4.1 Sovelluksen testaaminen käyttöönottovaiheessa

Testausprosessit voidaan jakaa kahteen kategoriaan, dynaamiseen ja staattiseen testaukseen. Testaus on dynaamista, kun testitapaukset suoritetaan käyttämällä testattavaa sovellusta ja vastaavasti staattista silloin, kun testitapaukset toteutetaan ilman testattavan ohjelman suorittamista. Staattinen testaaminen tapahtuu etsimällä virheitä ohjelmakoodista. Menetelmän avulla on mahdollista löytää sovelluksen loogiset virheet, mutta käyttöympäristön aiheuttamat ongelmat jäävät huomiotta. (Saleh 2009, 221–222.)

Koska käyttöjärjestelmäprojektissa käsitellään kaupallisia sovelluksia, joiden lähdekoodit eivät ole saatavissa, testauksen tulee olla dynaamista.

Toisaalta virheiden etsinnässä tarkoituksena on löytää käyttöympäristöön liittyvät yhteensopivuusongelmat, mikä ei ole staattisilla menetelmillä edes mahdollista.

4.2 Dynaaminen testaus

Dynaamisen testausprojektin kulku alkaa testaussuunnitelman kirjoittamisella. Suunnitelmaan kirjataan yksittäiset testitapaukset ja niistä odotettavat lopputulokset. Testitapaukset kuvaillaan sillä tarkkuudella, ettei niitä suorittaessa jää tulkinnanvaraa. Tarkkuus on oleellista, koska testauksen suorittaja voi olla jokin muu kuin suunnitelman kirjoittaja, mikä johtaa epäselvissä tapauksissa testitapausten tulkintaan. Epätarkat kuvaukset sekä tulkinta kyseenalaistavat testitulosten luotettavuuden; tulokset eivät välttämättä ole toistettavissa. (Saleh 2009, 222–223.)

Tarkasti kuvatut testitapaukset palvelevat myöhemmässä vaiheessa mahdollisesti suoritettavaa regressiotestausta, jonka lähtökohtana on testata sovellukseen tai ympäristöön tehtyjen muutosten vaikutusta. Mikäli aikaisemman testausprojektin tulokset on kirjattu riittävällä tarkkuudella, ongelmakohtiin palaaminen on huomattavasti nopeampaa. Tämän projektin tapauksessa aiemmasta dokumentaatiosta ei ole merkittävää apua, koska migraatio Windows XP:n ja 7 välillä tuo esiin täysin uudet ongelmatekijät verrattuna hallinnonalan aikaisempaan käyttöjärjestelmäuudistukseen.

Testitapaukset priorisoidaan, jotta resurssien kohdistaminen on mahdollisimman tehokasta. Usein testausprojekteissa ongelmaksi muodostuu käytössä oleva aika, jolloin kaikkea testitapauksia ei välttämättä pystytä toteuttamaan. Priorisointi perustuu kohteena olevan ohjelman toiminnallisuuksiin ja niiden kriittisyyteen. Mitä tärkeämmässä roolissa jokin toiminnallisuus on, sitä merkittävämpää on sen testaaminen. Kriittisimmät virheet voivat olla luonteeltaan sellaisia, että ohjelmaa ei saada toimimaan lainkaan. Alhaisemman prioriteetin virheet eivät estä sovelluksella työskentelyä. Ne voivat olla myös kierrettävissä toimimalla vaihtoehtoisella tavalla. (Saleh 2009, 222–223.)

Koska dynaamisella testauksella tarkoitetaan ohjelman suorittamista ja sen toimintojen testaamista, usein yksittäistä testitapausta edeltää sarja muita toimintoja. Testitapaus voi vaatia sisäänkirjautumisen tai muun edeltävän toiminnon suorittamista. Tulkinnanvaraisuuden vähentämiseksi testitapauksia edeltävät toimenpiteet kirjataan testausdokumenttiin. Kirjaaminen tehostaa varsinaista testausta ja lisää luotettavuutta, koska on toistettavissa tarkasti alun perin tarkoitetun tavan mukaan. Esimerkkinä voidaan käsitellä kirjautumisen vaativaa sovellusta. Mikäli ohjelmaan on mahdollista kirjautua eritasoisilla käyttäjätunnuksilla, oikean käyttäjätason mainitseminen on oleellinen osa luotettavasti suoritettua testitapausta. (Saleh 2009, 222–223.)

4.3 Testauksen kulku projektissa

Projektiin kuuluvan testauksen lukeminen yksittäiseen testauskategoriaan ei ole aivan yksiselitteistä ja menetelmä vaihtelee hieman lähteestä riippuen. Watkins (2001, 89) mainitsee käyttöjärjestelmäpäivityksen yhteydessä suoritettavan testauksen olevan regressiotestauksen eräs muoto. Testauksen voidaankin todeta olevan yhdistelmä useita menetelmiä.

Riippumatta testauksen oikeellisesta kategorioinnista, yksi ominaisuus erottaa tämän projektin tavanomaisesta sovellustestauksesta. Testausvaiheessa löydettyjen virheiden korjausprosessi ei ole yhtä suoraviivainen, mitä sovellustestauksessa tavanomaisesti noudatetaan. Koska kaupallisia sovelluksia on useita ja sovellustoimittajat eivät ole prosessissa mukana välittömästi, virheiden korjausvastuu osoitetaan lähtökohtaisesti projektiorganisaatiolle. Legacy-sovellusten virheiden korjaus ei myöskään tapahdu sovellusta, vaan käyttöjärjestelmää ja toimintaympäristöä muuttamalla.

Toinen testausprosessit erottava näkökulma liittyy virheiden etsimiseen. Tavanomaisen sovellustestauksen tarkoituksena on löytää virheitä yksinomaan testattavasta sovelluksesta (Myers ym. 2004, 6), kun projektiin liittyvässä testauksessa tarkoituksena on löytää virheitä sovelluksen asennuksesta ja käyttöjärjestelmän sekä sovelluksen yhteensoveltuvuudesta.

Jokaisen sovelluksen testaamiseen käytetään yhteistä testauslomaketta, joka kattaa muutaman yleisimmän toiminnon testaamisen, kuten sisään- ja uloskirjautumisen, tiedostojen avaamisen sekä tulostamisen. Kuten edellä mainitusta havaitaan, ensimmäisessä vaiheessa ei mennä kovin syvälle yksittäisen ohjelman toimintoihin. Tuloksista ei kirjata erillistä raporttia, vaan testitapaukset dokumentoidaan testauslomakkeeseen. Testausprojektin luonteesta johtuen oletettavia lopputuloksia ei ole kirjattu, eikä vertailua täten voida suorittaa. Yhteisenä oletuksena jokaiselle sovellukselle voidaan pitää tilaa, jossa ohjelma toimii ongelmitta. Riippuen siitä, päästäänkö tavoitteeseen, sovelluksen ilmoitetaan olevan joko hyväksytty tai hylätty. Hylkäämisen syy(t) ovat nähtävillä lomakkeessa olevien onnistui tai ei onnistunut -vastausten muodossa sekä vaihtelevista, vapaamuotoisista sanallisista kuvauksista.

Useimpien projektissa käytettävien testausmenetelmien kanssa on suositeltavaa, että ohjelmat testattaisiin vasta valmiilla työasemamallilla. Aikataulullisista syistä sovelluksia joudutaan kuitenkin testaamaan jo työaseman kehityksen alkuvaiheessa. Ongelma liian varhaisessa vaiheessa aloitetun testaamisen kannalta on työympäristöön tehtävät muutokset. Kehityksen alkuvaiheessa muutokset ovat usein laajempia ja jopa toimintatavat voivat muuttua. Loppuvaiheessa tehdyt muutokset taas ovat luonteeltaan pienempiä korjauksia. Mikäli muutos on tarpeeksi suuri, tai niitä joudutaan tekemään useita, ne voivat vaikuttaa sovelluksen toimintaan, eikä aikaisessa vaiheessa suoritettua testausta voida pitää luotettavana.

Testausprosessi on jaettu projektissa kahteen eri vaiheeseen, jolloin vain ensimmäinen testauskierros osuu päällekkäin työaseman kehityksen kanssa. Ensimmäisen kierroksen tavoitteena on saada kerättyä listaus sellaisista ohjelmista, joiden kanssa ilmenee niin asennus- kuin käynnistysvaikeuksia

sekä ongelmia yleisimpien toimintojen, kuten tiedostojen avaamisen kanssa.

Jälkimmäisessä sektorikohtaisessa testausvaiheessa käytetään valmista työasemamallia, jolloin käyttöjärjestelmään myöhemmin kohdistuvien muutosten tarve on mahdollisimman pieni. Toisella testauskierroksella jokaisen sovelluksen toiminnallisuudet testataan yksityiskohtaisemmin. Testaukseen käytetään henkilöitä, jotka työskentelevät sovellusten kanssa päivittäin. Sovelluksen käyttämiseen voidaan tarvita esimerkiksi käyttäjätunnus ja siihen liittyvät oikeudet, eikä vastaavanlaisia testauskäyttöön tarkoitettuja tunnuksia ole kaikissa tapauksissa saatavilla.

4.4 Käytettävät testausmenetelmät

Asennustestauksessa sovellus asennetaan työasemaan, jonka jälkeen varmistetaan, että mahdolliset liittymät muihin sovelluksiin tai laitteistoon ovat toimintakunnossa. Myös asennusprosessia tarkastellaan, jotta voidaan varmistua kaikkien sovellukseen kuuluvien komponenttien asentuminen. (Watkins 2001, 22.)

Mustalaatikkotestausta käytetään kolmansien osapuolien kehittämien, niin sanottujen kaupallisten sovellusten tarkasteluun. Projektissa käsiteltävät Legacy-ohjelmat ovat esimerkki tällaista sovelluksista, joita ei ole rakennettu vain yksittäisen asiakkaan it-infrastruktuuria varten. Koska tällaisten sovellusten rakennetta, eli lähdekoodia ei ole mahdollista tarkastella, puhutaan ohjelmasta mustana laatikkona, johon annetaan syöte ja tämän jälkeen tarkastellaan lopputulosta tietämättä tarkemmin, millaisten teknisten ratkaisujen kautta siihen päädytään. Tekniikan luonteesta johtuen suoritettavat testit valitaan käyttötapausten perusteella. Yksittäinen käyttötapaus voi olla esimerkiksi tulostaminen tai sovellukseen kirjautuminen. (Watkins 2001, 17.)

Vuorovaikutteisen testauksen, thread testing menetelmän tarkoituksena on testata järjestelmää tavalla, joka muistuttaa mahdollisimman paljon lopullista tuotantokäyttöä. Projektissa tällainen testaus tarkoittaa esimerkiksi yksittäisen sovelluksen testaamista sellaisen käyttäjän avulla, jolla on oikeat käyttöoikeudet ja joka käyttää sovellusta päivittäisten työtehtäviensä tekemiseen. (Watkins 2001, 17.)

Yhteensopivuustestauksessa tarkastellaan sovellusten sekä käyttöjärjestelmän välistä toiminnallisuutta. Vaikka lähes kaiken projektiin liittyvän testauksen voisi lukea yhteensopivuuskategoriaan kuuluvaksi, käytetään projektissa merkittäviä osia myös muista menetelmistä. (Watkins 2001, 23.)

4.5 Testitulosten kirjaaminen ja tarkastelu

Käytössä olevista sovelluksista koostettiin prioriteettilista, joka helpotti testaajien yhteistyötä sekä resurssien jakamista. Sovelluslistauksen avulla varmistettiin, ettei usea henkilö työskentele saman sovelluksen parissa, eikä jo testattua sovellusta oteta työn alle tarpeettomasti. Ensimmäisessä testausvaiheessa sovellusten parissa työskenteli noin kymmenen henkilön ryhmä. Yksi testaajista oli valittu vastuuhenkilöksi ja hänen työnään oli testausaikataulusta huolehtiminen. Henkilön vastuulla oli lisäksi varmistua siitä, ettei yksikään sovellus jää testaamatta.

Ensimmäisen yhteensopivuutta testaavan kierroksen aikana sovellukset jaettiin niiden käyttöympäristön mukaan rikosseuraamusalan, oikeushallinnon sekä ministeriön ohjelmiin. Testaajat saivat vastuulleen sovellukset siitä kategoriasta, josta vastasivat myös projektin ulkopuolella. Tämä mahdollisti aiemman sovellustuntemuksen hyödyntämisen testausprojektissa.

Jokaisesta sovelluksesta käytiin läpi testauslomakkeen sisältämät vaiheet, jonka jälkeen sovellus joko hylättiin tai hyväksyttiin riippuen testitapausten lopputuloksesta. Testituloksista pidettiin kirjaa prioriteettilistan avulla. Hyväksytty sovellus merkittiin listaan vihreällä ja hylätty sovellus punaisella värikoodilla. Listausta tarkastaessa yhteensopivuusongelmia aiheuttaneet sovellukset olivat helposti havaittavissa ja alkuvaiheessa tehdyn priorisoinnin ansiosta päätös jatkotoimenpiteistä helposti tehtävissä.

Testauksen vastuuhenkilö on osoittanut opinnäytetyössä tarkasteltavat sovellukset, tarkoituksena ratkaista näiden yhteensopivuusongelmat. Testaajien löytämät virhetilanteet kirjattiin testauslomakkeeseen, minkä ansiosta toimivuusongelmaa pystyi lähestymään välittömästi oikeilla työkaluilla. Tällöin ongelmien ratkaisemiselle varattua aikaa ei ollut tarvetta käyttää virheiden uudelleen etsimiseen.

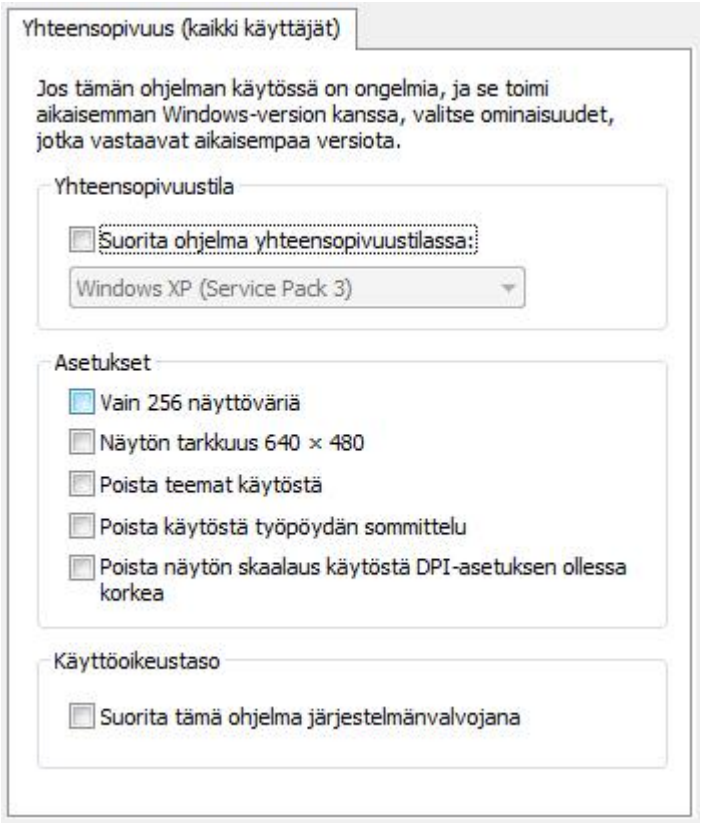
5 YHTEENSOPIVUUSTYÖKALUT

Vistan alkuperäisen julkaisun myötä ohjelmistotalo julkaisi lukuja, jotka kertoivat sovellusyhteensopivuudesta. Niin kaupallisten kuin ilmaissovellusten tapauksessa toimivuusongelmia korjattiin Service Pack 1 julkaisussa. Yrityksille tämä julkaisu oli merkittävä parannus, koska se sisälsi korjaukset yli 150 sovellukseen. Microsoftin julkaisema käyttömukavuuden kehitysohjelma mahdollisti toimivuusongelmista raportoimisen ja ohjelmistotalon mukaan yli 500 raporttia aiheuttaneille laitteille luvattiin julkaista ajuripäivitys. (Thurrott 2008, 123)

Kuvassa kolme esitelty käyttöjärjestelmään sisäänrakennettu yhteensopivuustila on toiminto, jonka avulla yleisimmät ongelmanaiheuttajat ovat korjattavissa yksinkertaisilla menetelmillä. Kun yhteensopivuustila-asetus on valittuna ja sovellus tekee kyselyn käyttöjärjestelmän versionumerosta, ohjelmalle lähetetään listalta valitun käyttöjärjestelmäversion tiedot.

Windows 7 uusittu graafinen ulkoasu Aero saattaa aiheuttaa eräitä yhteensopivuusongelmia. Graafiset asetukset koskevat laajemmalti tietokonepelejä, mutta voivat muodostua ongelmaksi myös toimistosovelluksissa. Graafiset yhteensopivuusongelmat eivät jokaisessa tapauksessa estä sovelluksen suorittamista, mutta ne voivat muuttaa ulkoasua niin, että käyttöliittymän kaikkia komponentteja ei näytetä tai ikkunoiden skaalaus ei toimi oikein, jolloin osia sovelluksesta voi jäädä näytön ulkopuolelle. (Understanding Application Compatibility n.d..)

Käyttöoikeustason valinta liittyy käyttäjätilien valvontaan UAC:iin. Sovellukset, jotka kirjoittavat suojattuihin hakemistoihin esimerkiksi Program Files -hakemistoon, vaativat järjestelmänvalvojan oikeudet toimiakseen. Valintaruutu ei heikennä tietoturvaa mahdollistamalla käyttöoikeuksien kiertämisen, vaan toimii siten, että jokaisella käynnistyskerralla käyttäjältä kysytään järjestelmänvalvojan tunnuksia.



Kuva 3. Erään sovelluksen yhteensopivuustila-asetukset.

Microsoft on listannut sivuillaan (Understanding Application Compatibility n.d.) eräitä Windows 7 tehtyjä muutoksia, jotka voivat vaikuttaa sovellusyhteensopivuuteen. 64-bittisissä käyttöjärjestelmissä on mahdollista ajaa 32-bittisiä sovelluksia WOW64-rekisteriemulaattorin avulla. Tämä asettaa kuitenkin rajoitukset 16-bittisten sovellusten käyttämiselle. Projektissa tehdyn esiselvityksen mukaan yksi 16-bittinen sovellus ei näin ollen tulisi toimimaan 64-bittisessä ympäristössä. Toinen tehty parannus, Windows Display Driver Model WDDM liittyy käyttöjärjestelmän graafisiin ominaisuuksiin. Grafiikkaongelmat liittyvät pääosin tietokonepeleihin, mutta ongelmia voi esiintyä myös kannettavien työasemien kanssa. Käyttäjille näkyvistä muutoksista korkean tarkkuuden asetukset, High DPI Awareness saattaa sekoittaa sovelluksen ulkoasun ja hankaloittaa käyttöä. Ominaisuus mahdollistaa näytöllä olevan tekstin suurentamisen ilman tarvetta resoluution pienentämiseen.

Parantaakseen järjestelmän suorituskykyä sekä luotettavuutta Microsoft on jättänyt uusista käyttöjärjestelmistä pois vanhentuneita tekniikoita. Eräs projektissa esiin tullut ongelma liittyy käyttöjärjestelmästä pois jätettyyn ohje-toimintoon. Sovelluksen ohjeiden lukeminen ei ole mahdollista, koska vanhoja .hlp-päätteisiä tiedostoja ei tueta. Ratkaisuna tähän ongelmaan Microsoft tarjoaa internetsivuillaan WinHlp32.exe laajennusta, joka lisää tuen vanhoille ohjetiedostoille. Näin toimimalla ohjelmistotalo kehottaa siirtymään pois vanhoista tekniikoista, mutta parantaa Legacy-sovellusten käytettävyyttä. (Understanding Application Compatibility n.d..)

5.1 Yhteensopivuusongelmien syyt

Microsoftin julkaistessa Vista käyttöjärjestelmän alettiin sovellusten yhteensopivuuteen kiinnittää huomiota enemmän myös ohjelmistotalon puolelta. Uusi käyttöjärjestelmä sisälsi niin suuria muutoksia, että vanhat XP:lle tehdyt sovellukset eivät aina toimi ongelmitta. Yhdeksi esimerkiksi voidaan ottaa käyttöjärjestelmän versionumerointi. Kun XP:n versionumero oli 5.x, se vaihtui Vistassa 6.0. Näin pelkkä ohjelmaan sisäänrakennettu versiontarkastus voi aiheuttaa ristiriidan, jonka takia sovellukset eivät toimi uudessa ympäristössä. (Windows 7 and Windows Server 2008 R2 Application Quality Cookbook 2009.)

Huomattava osa projektissa havaituista ongelmista johtuu Windows 7 turvallisuusasetuksista, jossa tiettyjen järjestelmähakemistojen sekä rekisterin muokkaaminen on estetty peruskäyttäjältä. XP aikana työaseman paikallinen käyttäjätili kuului oletusarvoisesti järjestelmänvalvojat -ryhmään, jolloin suojattujen hakemistojen kanssa työskennellessä ei esiintynyt ongelmia. Koska tämä toiminta-ajatus muuttui Vistan julkaisun myötä, myös yhteensopivuusongelmiin jouduttiin kiinnittämään huomiota (Thurrott 2008, 286). Esimerkkinä voidaan tarkastella sovellusta, jonka pääkäyttäjä asentaa suojattuun järjestelmähakemistoon c:\program files ja joka aina käynnistyksen yhteydessä tarkistaa sekä asentaa palvelimelta ladattavat päivitykset. Peruskäyttäjällä on oikeus käyttää sovellusta normaaliin työskentelyyn, mutta päivitystilanteessa asennushakemistoon kirjoittaminen ei onnistu ja sovelluksen suorittaminen lakkaa virheilmoitukseen.

5.2 UAC-käyttäjätilien valvonta

Yksi Windows Vistan mukana julkaistuista uusista ominaisuuksista oli käyttäjätilien valvonta, UAC. UAC vaati käyttöjärjestelmää kohtaan tehtävien muutosten jaottelun kahteen kategoriaan. Ensimmäinen sisältää toiminnot joihin vaaditaan pääkäyttäjän käyttöoikeudet. Jälkimmäiseen kategoriaan kuuluvat toiminnot, jotka voidaan suorittaa peruskäyttäjän oikeuksilla. Jokaiselle työasemaan kirjautuneelle peruskäyttäjälle annetaan tietyn tason oikeudet, jotka käyttäjätilien valvonta tarkistaa aina järjestelmään kohdistuvien muutospyyntöjen kohdalla (Thurrott 2008, 286). Perusajatus Windows 7:ssa on ajaa jokaisen, myös pääkäyttäjän istunto rajoitetuilla käyttöoikeuksilla. Kun käyttäjätilien valvonta havaitsee muutospyynnön, joka koskee käyttöjärjestelmää, UAC pyytää korotettuja oikeuksia. Käyttöoikeuksien kyselyikkunan sisältö vaihtelee tapauskohtaisesti riippuen sen hetkisistä oikeuksista sekä suoritettavasta tehtävästä. (Panek 2010, 335.)

Käyttäjätilien valvonta joutui Vistan julkaisun myötä kiistellyn ominaisuuden asemaan. Käyttöoikeuksien kyselyä pidettiin työskentelyä häiritsevänä ominaisuutena, kun ponnahdusikkunan lailla usein esiin tuleva näkymä vaati käyttäjältä toimenpiteitä. Ominaisuutta on paranneltu edelleen Windows 7 myötä muun muassa sallimalla peruskäyttäjän suorittaa useampia järjestelmää koskevia tehtäviä, sekä tarjoamalla ylläpitäjille mahdollisuuden hallita ilmoitusten näyttämistä ryhmäkäytäntöjen avulla (What’s New in User Account Control 2009). Toisaalta Vistalle sekä 7

kehitetyt ohjelmat on tehty tietoisina käyttäjätilien valvonnasta, mikä on edelleen vähentänyt käyttöoikeuksien kyselyjä sekä ongelmatilanteiden määrää tavanomaisessa työasemakäytössä. UAC onkin ominaisuus, jota Windows 7:ssa ei oletusarvoisesti saa pois käytöstä kuten Vistassa. OTTK:n 7-ympäristössä käyttäjätilien valvonta näkyy loppu-käyttäjille oletusasetuksillaan.

Suorita järjestelmänvalvojana toiminto nopeuttaa merkittävästi korotetuin oikeuksin ajettavan sovelluksen suorittamista. Peruskäyttäjän ei tarvitse kirjautua ulos työasemasta, jotta järjestelmänvalvoja pääsee suorittamaan ohjelman, vaan käyttöoikeudet voidaan korottaa myös rajoitetun istunnon aikana. Ongelmana yritysympäristössä on järjestelmänvalvojan käyttöoikeuksien jakaminen. Turvallisuussyistä peruskäyttäjä ei tiedä järjestelmänvalvojan salasanaa, joten käyttöoikeuskyselyä varten paikalla pitää olla henkilö, jolla on tiedossa kirjautumistunnukset. Mikäli jonkin tuotannossa olevan sovelluksen toimivuusongelma ratkaistaan käyttöoikeuksia korottamalla, UAC suorittaa kyselyn jokaisella kerralla, kun sovellus avataan. Suurissa ympäristöissä tällainen järjestely on, ellei mahdoton, niin hankala toteuttaa, koska käyttöoikeuskyselyjä voi tulla yhden päivän aikana satoja. Ympäristössä, jossa käyttäjällä on tiedossa järjestelmänvalvojan tunnukset, ominaisuus on toimiva sekä yksinkertainen ratkaisu käyttöoikeusongelmiin.

Windows 7 yhteensopivissa ohjelmissa käyttöoikeuksien kyselyyn on kiinnitetty erityisesti huomiota ja jo sovelluksen kehitysvaiheessa tulee miettiä, mihin toimintoihin tarvitaan korotettuja oikeuksia ja mitkä jokainen käyttäjä voi suorittaa. Jotkin Legacy-sovellukset tarkistavat käyttöoikeudet ja voivat vaatia järjestelmänvalvojan tasoisia tunnuksia, vaikka sovellus ei teknisesti tekisi näillä mitään.

5.3 UAC-virtualisointi

Useat Legacy-ohjelmat on suunniteltu kirjoittamaan järjestelmähakemistoihin, jotka ovat Windows 7:ssa suojattuja ja joita on mahdollista muokata vain pääkäyttäjän oikeuksilla. Tätä yhteensopivuusongelmaa varten on kehitetty käyttäjätilien valvonnan virtualisointiratkaisu. Kun suojattuun kansioon yritetään kirjoittaa peruskäyttäjän oikeuksilla, käyttöjärjestelmä havaitsee ongelman ja ohjaa tiedostot profiilikohtaiseen virtualstore hakemistoon. Sovelluksen tallentamat työtiedostot voivat olla esimerkiksi profiilikohtaisia asetuksia tai käyttäjän luomia tiedostoja. Koska käyttäjän luomien tiedostojen tallennushakemiston ei tarvitse sijaita asennushakemistossa, tietoturvaa lisätäkseen käyttöjärjestelmä ei tue työtiedostojen tallennusta suojattuihin hakemistoihin. UAC virtualisointi uudelleenohjaa tiedostojen lisäksi rekisteriin HKLM\Software tehtävät kirjoitukset haaraan HKCU\Software\Classes\VirtualStore. Tiedostojen uudelleenohjaus on aina käyttäjäkohtaista, jolloin yhden käyttäjän tekemät muutokset eivät näy muille samaan työasemaan kirjautuville käyttäjille. (Inside Windows Vista User Account Control 2007.)

5.4 Työkalut

Microsoft on parantanut Legacy ohjelmien tukea edelleen Windows 7 julkaisun myötä muun muassa julkaisemalla käyttöjärjestelmään sisäänrakennetun XP-moden. Ominaisuus on suunnattu pääosin yksityiskäyttöön, ei ratkaisuksi yrityksen käyttämien sovellusten yhteensopivuusongelmiin. Tähän tarkoitukseen on suunniteltu Microsoft Desktop Optimization Pack, MDOP. Paketti sisältää työkalut, joiden avulla käytössä olevat sovellukset, tai koko työympäristö voidaan virtualisoida. Merkittävin ero työkalujen välillä on niiden skaalautuminen usean käyttäjän ympäristöön. Infrastruktuurin hallitseminen sekä ylläpito on yksinkertaisempaa keskitetyn MDOP-työkalun avulla. Kumpaa tahansa ratkaisua käytetään, tuotantomallin rakentaminen vaatii suuressa ympäristössä huomattavia resursseja.

5.4.1 Application Compatibility Toolkit

MDOP:n ulkopuolelle jää vielä yksi suuri kokonaisuus, Application Compatibility Toolkit, ACT. Se on saatavilla ilmaiseksi ja se sisältää työkaluja alkaen yrityksen sovellusportfolion kartoittamisesta ja päättyen korjausmenetelmien ylläpitoon tuotantokäytössä.

ACT sisältää usean eri tason työkaluja, jos korjausten suorittamista tarkastellaan automaation näkökulmasta. Yksinkertaisin työkalu on Standard User Analyzer Wizard. Ohjelmalle kerrotaan, mitä ongelmallista sovellusta halutaan testattavan, jonka jälkeen käyttäjä tekee ohjatun toiminnon mukaiset toimenpiteet ja lopuksi SUA Wizard tarjoaa ratkaisuehdotukset käyttäjän valintojen mukaan. Mikäli tarvitaan kehittyneempää työkalua, jolla ongelman aiheuttaja voidaan osoittaa yksityiskohtaisesti, apuna voidaan käyttää Standard User Analyzeria (SUA). SUA:n toimintaperiaate on sama kuin edellä mainitussa työkalussa, mutta ylläpitäjällä on mahdollisuus vaikuttaa enemmän sovelluksen suorittamiseen liittyviin asetuksiin, kuten käyttäjän käyttöoikeustasoon. Ohjelma tarjoaa korjausehdotusten lisäksi raportin kaikista havaituista ongelmista, joten ylläpitäjän on mahdollista tarkastella hyvin yksityiskohtaisella tasolla ongelman aiheuttaneita virheilmoituksia. Koska käyttöoikeuksien valvonta on yksi merkittävimmistä yhteensopivuusongelmien aiheuttajista, Standard User Analyzerin avulla voidaan tarkastella sekä korjata yksinomaan tähän ominaisuuteen liittyviä ongelmia. Työkalun avulla ei voida korjata muita virhetilanteita. (Windows 7 and Windows Server 2008 R2 Application Quality Cookbook 2010.)

Mikäli tarvitaan vielä yksityiskohtaisempia tapoja toimimattomuusongelmien ratkaisemiseksi, voidaan käyttää Compatibility Administrator -työkalua. Ohjelman avulla käyttäjällä on mahdollisuus valita yksittäisten yhteensopivuuskorjausten, shimmien, tasolla mitä niistä toimimattomaan sovellukseen käytetään. Yhdelle sovellukselle on mahdollista osoittaa useita korjauksia, mutta testausvaiheessa on suositeltavaa kokeilla näiden vaikutusta mieluummin yksi kerrallaan. Tämä toimintatapa koskee myös aiemmin mainittuja SUA Wizardia sekä SUA:ta.

5.4.2 Shimmit

Shimmit ovat korjauksia, jotka toimivat käyttöjärjestelmän ja sovelluksen välillä ohjelmointirajapinnassa. Kun sovellus tekee kyselyn käyttöjärjestelmälle, shimmi ohjaa tämän itselleen ja palauttaa muokatun vastauksen takaisin sovellukselle. Esimerkkinä voidaan tarkastella Windowsin versionumeron tarkistusta. Jotkin Legacy-sovellukset ovat ohjelmoitu vertaamaan käyttöjärjestelmän versionumeroa valmiiksi asetettuun arvoon. Kun versionumero kasvaa uuden käyttöjärjestelmän myötä, tarkistuksesta syntyy yhteensopivuusongelma. On mahdollista, ettei sovellus aiheuta muita toimivuusongelmia ja mikäli tarkistus voidaan kiertää, ohjelma saadaan toimimaan myös uudessa käyttöjärjestelmässä. Kun sovellus tekee kyselyn versionumerosta, shimmi voidaan ohjelmoida palauttamaan jokin muu kuin käyttöjärjestelmän todellinen arvo. Ohjelma ei erota, tuleeko vastaus käyttöjärjestelmältä vai shimmiltä. (Finn, Gibson, & Van Surksum 2011, 36.)

Shimmejä voidaan luoda Application Compatibility Toolkitin avulla, joka sisältää 361 käytettävissä olevaa yhteensopivuuskorjausta. Korjaukset ovat aina sovelluskohtaisia ja ne voidaan tarkasti määritellä toimimaan vain tietyssä sovellusversiossa. Mikäli shimmejä käytetään toimivuusongelmien korjaamiseen tuotantoympäristössä, on kaksi tapaa jakaa ne loppukäyttäjille. Yhteensopivuuskorjaukset tallennetaan tietokantaan ja jokaista sovellusta varten julkaistaan uusi kanta. Tämä on yksinkertainen tapa jakaa korjaukset työasemiin. Mikäli korjattavien sovellusten lukumäärä on suuri, kaikki shimmit voidaan tallentaa yhteen tietokantaan. Tällöin vältytään lukuisten erillisten kantojen hallinnalta. Keskitetty jakelutapa vaatii resursseja toteutusvaiheessa, mutta vastaavasti helpottaa ylläpitovaiheessa tarvittavaa työtä. Vaihtoehtoisten jakelumallien ansiosta yhteensopivuuskorjauksia voidaan hyödyntää mahdollisimman tehokkaasti sekä pienissä, että suurissa ympäristöissä.

5.4.3 Process Monitor

Ohjelman avulla käyttäjä voi seurata työasemassa käynnissä olevia prosesseja sekä rekisterin, tiedostojärjestelmän ja verkon käyttöä. Työkalua ei ole suunniteltu erityisesti yhteensopivuutta silmällä pitäen, mutta sitä voidaan hyödyntää, kun toimivuusongelman syyt halutaan eritellä yksityiskohtaisesti. Sovellus valvoo kaikkia työaseman tapahtumia ja halutut voidaan suodattaa näkyviin. Process Monitor on tehty valvontaa varten, eikä sen avulla ole mahdollista luoda korjauksia, kuten esimerkiksi Compatibility Administrator työkalun avulla.

5.4.4 XP-mode ja Microsoft Desktop Optimization Pack

Windows XP-mode on Microsoftin tarjoama ratkaisu yhteensopivuusongelmiin. XP-mode on kuitenkin suunnattu enemmän yksityiskäyttöön kuin yrityksille. XP-mode on Windows 7 työasemaan asennettava virtuaalikone, jonka avulla 7:ssa toimimattomat ohjelmat voidaan suorittaa virtuaalisessa XP ympäristössä. XP-moden varjopuolena voidaan nähdä ylläpidol-

liset- sekä tietoturvaongelmat. Koska ominaisuuden käyttäminen vaatii edelleen XP:n asentamisen, ylläpidollinen työmäärä kasvaa. (Finn ym. 2011, 70.) Tämän lisäksi virtuaaliasennus ei poista sitä tosiasiaa, että XP:n tuki loppuu 8.4.2014. (Windows lifecycle fact sheet). Ominaisuus voidaankin mieltää mieluummin siirtymäkauden ratkaisuksi kuin pysyväksi toimintomalliksi.

Yrityskäyttöön XP-moden sijasta Microsoft tarjoaa MED-V, Microsoft Enterprise Desktop Virtualization ratkaisua. MED-V on osa Microsoft Desktop Optimization Pack:ia (MDOP), joka on käytettävissä Software Assurance sopimuksen tehneille yrityksille. MDOP sisältää kolme eri tason virtualisointiratkaisua, joista Application Virtualization (APP-V) on tarkoitettu sovellusten virtualisointiin, Microsoft Enterprise Desktop Virtualization (MED-V) työpöydän virtualisointiin ja User Experience Virtualization (UE-V) voidaan yksinkertaistettuna ajatella käyttäjän profiilin ja asetusten virtualisointina.

Kolmesta yllä mainitusta työkaluista MED-V soveltuu parhaiten ongelmallisten Legacy-sovellusten korjausratkaisuksi. Siinä missä APP-V:n avulla voidaan ratkaista keskenään ristiriitoja aiheuttavien sovellusten ongelmat, MED-V tarjoaa ratkaisun yksittäisen sovelluksen ja käyttöjärjestelmän välisiin ongelmiin. MED-V:n yksinkertainen Proof of Concept (POC) kokoonpano vaatii palvelimen, jonka avulla hallitaan virtuaalisia työtiloja, Management Console -työaseman, jolla luodaan virtuaaliset työasemamallit sekä Client sovelluksen, joka asennetaan loppukäyttäjän työasemaan.

Mitä suurempi ympäristö on kyseessä sitä suurempaan rooliin vikasietoisuus ja järjestelmän kuormituksen kestävyys nousevat. Edellä esitetyssä yksinkertaisessa mallissa palvelinta käytetään kahden roolin ylläpitämiseen, mikä ei ole mahdollista suurissa ympäristöissä. Toimivan ja vikasietoisen kokoonpanon rakentaminen OTTK:n ylläpitämään ympäristöön yli 10 000 työasemalle vaatiikin huomattavasti POC-kokoonpanoa suurempia resursseja. Tämän kokoisen tuotantoympäristön minimikokoonpano sisältää viisi palvelinta sekä jokaiseen työasemaan asennettavat client-sovellukset, mikä kasvattaa laitteistovaatimuksia merkittävästi. (Infrastructure planning and design.)

Tuotannossa olevien ohjelmien toimivuudesta Windows 7:ssa on tehty alustava kysely jo projektin esiselvityksessä. Sovellusten toimittajat ovat ilmoittaneet, toimivatko sovellukset suoraan uudessa käyttöjärjestelmässä tai voivatko he vaihtoehtoisesti tarjota korjauksen nykyiseen ohjelmaan. Esiselvitys on tarjonnut karkean arvion toimimattomien ohjelmien lukumäärästä ja sitä kautta on voitu arvioida korjauksiin tarvittavat resurssit. Esiselvityksen lista ei ole vedenpitävä, koska yhtään sovellusta ei ole listausvaiheessa vielä testattu uudessa työympäristössä, kyseessä on pelkkä arvio toimivuudesta. Testauksen lähdettyä liikkeelle, käytäntö on osoittanut useamman sovelluksen toimivan vastoin toimittajan antamaa tietoa. Toisaalta jokin OTTK:n työympäristössä oleva muu sovellus tai toiminnallisuus on voinut estää aiemmin toimivaksi ilmoitetun sovelluksen käyttämisen. Esiselvityksen tekeminen on tarjonnut arvion projektiin liittyväs-

tä testauksen määrästä ja se on auttanut priorisoimaan tietyt sovellukset testausjärjestyksessä korkeammalle. Selvitys ei kuitenkaan vähennä testaukseen liittyvää työmäärää, koska jokainen Legacy-sovellus testataan työympäristössä toimittajan arviosta huolimatta. Tarkoituksena on muodostaa organisaatiolle sisäinen näkemys vaadittavista korjaustoimenpiteistä ja niiden laajuudesta.

5.4.5 Etätyöpöytä

Mikäli sovellus ei toimi uudessa käyttöjärjestelmässä, eräs vaihtoehto on hyödyntää Windowsin etätyöpöytäominaisuutta. Ohjelma asennetaan tällöin yksittäiseen koneeseen, jonka käyttöjärjestelmässä Legacy-sovellus toimii ongelmitta. Loppukäyttäjä voi kirjautua ohjelmaa isännöivään koneeseen ja käyttää sovellusta verkkoyhteyden avulla. Ylläpidollisesti yksittäinen Legacy-käyttöjärjestelmä vaatii vähemmän resursseja kuin jokaiseen työasemaan asennettava XP-mode ja sopii erityisesti pienten käyttäjämäärien sovelluksille.

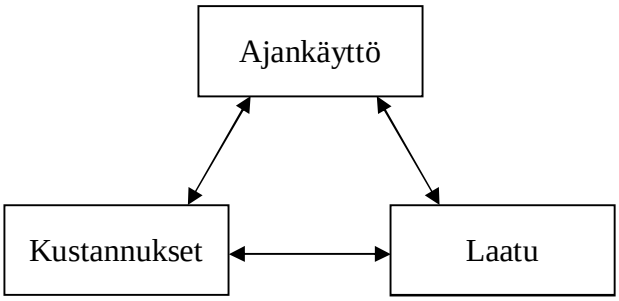
Merkittävin ongelma etätyöpöytäratkaisua käytettäessä on sovelluksen yhteensopivuus usean yhtäaikaisen käyttäjän tapauksessa. Mikäli sovellusta ei ole suunniteltu tukemaan päällekkäisiä istuntoja, sen toiminnassa voi esiintyä ongelmia, kuten tiedostojen tuhoutuminen. Loppukäyttäjä saattaa havaita myös suorituskyvyn heikkenemistä, jos ohjelma ei pysty käsittelemään päällekkäisiä pyyntöjä tehokkaasti. Tämä on ongelma varsinkin hitaiden verkkoyhteyksien päässä olevissa työasemissa. Lisäksi sovelluksen virtualisointi etätyöpöytäyhteyden läpi voi aiheuttaa ongelmia fyysisten oheislaitteiden, kuten tulostinten toiminnassa. (Application Compatibility 2009.)

5.5 Käyttömallien kustannukset

Vaikka edellä mainittuihin sovelluksiin ei tarvitsisi hankkia maksullisia lisenssejä, käytännössä mitään käsitellyistä ratkaisuista ei voida pitää täysin ilmaisena vaihtoehtona. Jokainen ratkaisuvaihtoehto vaatii omanlaisen infrastruktuurin rakentamisen, esimerkiksi uuden palvelimen ylläpitämisen tai asiakassovelluksen asentamisen tuotannon työasemiin. Ennen kuin valittu ratkaisumalli voidaan ottaa käyttöön, sen toimivuus tulee testata, mihin vaaditaan myös resursseja. Etenkin suurissa ympäristöissä tuotantoon siirryttäessä laitteistovaatimukset kasvavat merkittävästi, koska palvelun tulee olla vikasietoinen ja tarjolla jokaiselle käyttäjälle. Ylläpidon ja myöhempien muutostarpeiden huomioiminen etukäteen auttavat oikean käyttömallin valinnassa. Sovellukset jotka ovat käytössä vain yksittäisissä toimipisteissä, tai muutamalla käyttäjällä pystytään ylläpitämään paikallisesti, mutta mitä suuremmalle käyttäjäkunnalle palvelua tarjotaan, sitä merkittävämmässä roolissa on keskitetty hallinta. Edellä käsitellyistä työkaluista löytyy sopivimmat ratkaisuvaihtoehdot kumpaankin skenaarioon.

6 RATKAISUMALLIEN VALINTA

Yksittäiseen yhteensopivuusongelmaan voi löytyä useita ratkaisuja, jolloin oikean menetelmän valitseminen ei ole aivan suoraviivaista. Valinta tapahtuu tällöin etsimällä kompromissi kuvassa neljä esiintyvien kolmen osakokonaisuuden välillä.



Kuva 4. Ratkaisumallin valintaan vaikuttavat osatekijät

Voidaan ajatella, että yksi kokonaisuus riippuu kahden jäljelle jäävän osan suhteesta. Mitä enemmän halutaan säästää kustannuksissa, sitä enemmän tulee vähentää ajankäyttöä sekä tinkiä ratkaisujen määrästä ja/tai laadusta.

Lopullinen ratkaisu näiden kolmen osakokonaisuuden välillä riippuu tarkasteltavasta ohjelmasta. OTTK:n tekemä sovellusten prioriteettilista toimii suuntaa antavana työkaluna. Mitä korkeammalle sovellus on priorisoitu, sitä enemmän toimivan ratkaisun löytyminen korostuu rahan- sekä ajankäytön kustannuksilla. Mikäli sovellus on luokiteltu hyvin alhaisen prioriteetin ohjelmaksi, huomattavasti yksinkertaisempi ratkaisu riittää.

Käyttäjätilien valvontaan kuuluva järjestelmähakemistojen kirjoitussuojaus on usein kierrettävissä yksinkertaisella menetelmällä. Jos sovellus työskentelee suojatussa hakemistossa, eikä virtualstoren hyödyntäminen ole mahdollista, ongelma voidaan yksinkertaisimmillaan ratkaista lisäämällä peruskäyttäjälle muokkausoikeus suojattuun hakemistoon. OTTK on linjauksellaan kuitenkin pyrkinyt minimoimaan Program Files -hakemiston käyttöoikeuksien muokkaamisen.

6.1 Ohjelmat

Ensimmäisen testausvaiheen aikana toimivuusongelmia löytyi neljästä sovelluksesta, joista kahta tarkastellaan tässä yksityiskohtaisemmin. Eniten yhteensopivuusongelmia aiheuttava tekijä oli järjestelmäkansioiden rajoitetut käyttöoikeudet. Kummassakin tapauksessa ongelmat saatiin ratkaistua oikeuksia muokkaamalla. Tavoitteena oli löytää myös vaihtoehtoisia käyttömalleja, joten molemmille sovelluksille kehitettiin myös vaihtoehtoinen ratkaisumenetelmä.

6.1.1 Prima

Prima on työasemaan asennettava henkilöstöhallinnon sovellus, joka käynnistyessään tarkistaa päivitykset ohjelmalle osoitetulta palvelimelta. Mikäli uusi päivitys on julkaistu, käyttäjältä kysytään kirjautumisen jälkeen lupaa päivityksen asentamiseen. Ongelmatilanne syntyy, kun peruskäyttäjän oikeuksilla kirjautunut henkilö hyväksyy päivityksen asentamisen. Koska uusien päivityksien julkaisua ei pystytä ennakoimaan, asentamisen tulee onnistua peruskäyttäjältä.

Sovelluksen toimivuus testattiin UAC virtualisointia hyödyntämällä ja kopioimalla asennustiedostot c:\programfiles -hakemistosta profiilikohtaiseen virtualstore hakemistoon. Tämän lisäksi muutettiin polkumäärittelyt kahteen ohjelman käyttämään ini-tiedostoon. Muutosten jälkeen päivitysten asennus onnistui myös rajoitetuilla käyttöoikeuksilla koska työkansiona toimi käyttäjän profiilikohtainen hakemisto.

Toimivuusongelman vaihtoehtoinen ratkaisu toteutettiin muokkaamalla hakemiston käyttöoikeuksia. Microsoft suosittelee tätä tapaa yksinkertaiseksi ratkaisuksi mm. tukisivullaan (Tuotetuki n.d.). Koska peruskäyttäjä ei voi hyödyntää käyttöjärjestelmän sisäänrakennettua ominaisuutta, suorita järjestelmänvalvojana, tälle annettiin kirjoitusoikeudet ohjelman asennushakemistoon. Ominaisuuden käyttöä ei ole estetty teknisten ratkaisujen avulla, vaan peruskäyttäjän ongelmana on, ettei tämä tiedä järjestelmänvalvojan salasanaa, eivätkä pääkäyttäjät voi auttaa laajassa ympäristössä jokaista apua tarvitsevaa.

Kahdesta edellä esitetystä korjausehdotuksesta suositeltavaksi valittiin jälkimmäinen. Profiilikohtaisen virtualstore hakemiston käyttö todettiin sovelluksen asennusvaiheessa huomattavan monimutkaiseksi, koska asennuksen aikana ei ollut mahdollisuutta muuttaa kohdekansiota. Käytetty asennuspaketti ei myöskään mahdollistanut korjattujen ini-tiedostojen lisäämistä. Tämä aiheutti vielä yhden asennuksen jälkeisen työvaiheen. Toisaalta peruskäyttäjän käyttöoikeuksia muokatessa oli tarvetta lisätä vain kirjoitusoikeus yksittäiseen kansioon asennushakemistossa, eikä asennuspaketin muokkaamiselle ollut tarvetta

6.1.2 ZOC

ZOC on käytöstä poistuva pääteohjelma, jonka avulla työasemalta kirjaudutaan keskuskoneeseen. Sovellus tulee väistymään uuden järjestelmän tieltä, mutta on käytössä niin kauan kun vanhaan järjestelmään kirjatut asiat ovat ratkaisematta. Vaikka kyseessä on elinkaarensa loppupuolella oleva sovellus, uuteen käyttöjärjestelmään siirtyminen tapahtuu nopeammalla aikavälillä kuin ZOC:n korvaavan sovelluksen käyttöönotto. Tästä johtuen ohjelman yhteensopivuusongelmat otettiin tarkasteluun.

Sovellus käyttää asennuskansiossa sijaitsevaa ini-tiedostoa eräiden asetusten, kuten työhakemistojen määrittelyyn. Ini-tiedosto on muokattu toimivaksi XP ympäristössä ja näin ollen vanhat polkumäärittelyt eivät toimi 7:ssa. Microsoftin muuttamien suositusten mukaan myös profiilikohtaiset

asetukset sekä asennushakemiston kirjoitusoikeudet aiheuttivat ristiriitoja sovelluksen toimivuudelle. Edellä mainituista syistä alkuperäistä msi-asennuspakettia ei voitu käyttää uudessa ympäristössä.

Jotta ennalta määritetyt, uudessa käyttöjärjestelmässä toimimattomat asetukset saatiin kierrettyä, ZOC asennettiin testiympäristöön manuaalisesti exe-tiedostosta. Asennuksen aikana datakansion sijainniksi asetettiin oletuskäyttäjän profiilikohtainen hakemisto ja asennuksen jälkeen tehtiin muutokset ini-tiedostoon. Ensimmäisen käynnistyksen yhteydessä sovellus luo määrityksistä riippuen joko kone- tai käyttäjäkohtaisen datahakemiston. Jotta käyttökokemuksesta saatiin mahdollisimman yksinkertainen loppukäyttäjää ajatellen, prosessissa hyödynnettiin Windowsin sisäänrakennettua oletuskäyttäjän, default user profiilia. Default user käyttäjäprofiilin hakemistorakenne kohteessa C:\Users\Default\AppData kopioidaan uudelle käyttäjälle, kun tämä kirjautuu työasemaan ensimmäisen kerran. Automaattisesti tapahtuvan kopioinnin ansiosta loppukäyttäjän ei tarvitse määritellä datahakemiston sijaintia.

Kuten aiemmin käsitelty Prima, myös ZOC käyttää työhakemistonaan Program Files -kansiota. Koska käyttäjätilien valvonta estää käyttäjäkohtaisten muutosten tekemisen edellä mainittuun hakemistoon, asetusten muokkaaminen jälkeenpäin ei onnistu. Vaihtoehtoinen ratkaisu toimivuusongelmaan on asentaa sovellus oletusasetuksin ja lisätä peruskäyttäjälle muokkausoikeudet kansioon C:\Program Files\ZOC. Tämä toimintatapa on kuitenkin OTTK:n linjauksen vastainen, joten sitä ei oteta huomioon toimenpide-esitystä tehtäessä.

6.2 Valintaperusteet ja poisjääneet ratkaisut

Työssä tutkittiin kahden sovelluksen toimivuusongelmia, joihin pyrittiin etsimään useampi ratkaisuvaihtoehto. Näiden joukosta kummallekin sovellukselle suositeltiin parhaiten soveltuvaa vaihtoehtoa ottaen huomioon aiemmin, kuvassa kolme esitetyt muuttujat: ajankäyttö, laatu sekä kustannukset. Neljäs tekijä joka voidaan lukea laatu-kategoriaan kuuluvaksi, oli käyttömallin elinkaari. Lopullisiksi ratkaisuiksi valitut mallit eivät vaadi taustalla olevien järjestelmien ylläpitoa, eikä korjauksiin käytetty vanhentuneita tekniikoita, joten käyttömallin elinkaari ei jää kummankaan sovelluksen tapauksessa liian lyhyeksi.

Merkittävin ratkaisumallien valintaan vaikuttava tekijä oli havaittujen ongelmien laatu. Yksinkertaisen ongelman korjaaminen monimutkaisilla menetelmillä tai raskaalla työkalulla ei välttämättä tuo ratkaisulle lisäarvoa. Mikäli tätä näkökulmaa ei oteta huomioon, vaarana on resurssien tuhlautuminen ja pahimmassa tapauksessa lisäkustannusten syntyminen siirryttäessä ylläpitovaiheeseen.

Poisjääneiden ratkaisujen ongelmaksi koettiin suuret resurssivaatimukset. Yhden sovelluksen korjaamiseksi ei ollut perusteltua rakentaa tuotantoympäristöön raskasta virtualisointiratkaisua, joka olisi vaatinut niin laitteiston kuin sovelluksen ylläpitoa. Suurimmaksi esteeksi ei niinkään koitunut maksullisuus, sillä tarvittavat sovellukset olisivat olleet saatavilla

uuden käyttöjärjestelmän myötä ilman lisäkustannuksia. Mikäli hankintoihin olisi pitänyt ryhtyä, käytössä olevien työkalujen soveltuvuus olisi otettu tarkasteluun. Suurten keskitettyjen järjestelmien ongelmana oli myös vikasietoisuuden aiheuttamat kustannukset. Useiden uusien palvelinten rakentaminen ja vakaan palvelun tarjoaminen 10 000 käyttäjälle 400 eri toimipisteeseen valtakunnallisesti olisi johtanut huomattaviin kustannuksiin. Koska kummassakin tapauksessa korjaustoimenpiteet suoritettiin käyttöjärjestelmän sisäänrakennetuilla ominaisuuksilla, ulkopuolisten järjestelmien vikasietoisuuteen ei tarvinnut kiinnittää erityistä huomiota.

Selvitystyön tarkoituksena oli yhtäältä kyseenalaistaa yksinkertaisimman käyttömallin valinta ja tutkia, tarjoaisiko vaihtoehtoinen menetelmä lisäarvoa ratkaisulle. ZOC:n tapauksessa yksinkertaisin ratkaisuvaihtoehto jätettiin pois lopullisesta toimenpide-esityksestä, koska profiilikohtainen asennus tarjosi loppukäyttäjälle laajemmat muokkausvaihtoehdot sovelluksen asetuksiin sekä näkymään liittyen.

Sovelluksista tehdyt toimenpide-esitykset on osoitettu toimeksiantajalle erillisinä dokumentteina. Esityksissä on huomioitu korjaustoimenpiteisiin tarvittavat resurssit ja käyty yksityiskohtaisesti läpi, mitä muutoksia työympäristöön tulee tehdä. Dokumentissa on kuvailtu asennukseen liittyvät vaiheet, mutta lopullista asennuspakettia ei kummankaan sovelluksen kohdalla päästy tekemään resursseista tai teknisistä syistä johtuen.

Priman valmis asennuspaketti asetti rajoituksia, jotka kyseenalaistavat lopullisen käyttömallin valinnan. Ongelmaksi muodostui sovelluksen asennushakemisto, jonka vaihtaminen ei ollut mahdollista. Ohjelma ja sen päivitykset saatiin toimimaan testausvaiheessa yksinkertaisesti kopioimalla asennushakemisto peruskäyttäjän profiiliin, joka on Windows 7:ssa suositeltu hakemisto jälkeenpäin tapahtuvaa muokkausta vaativille sovelluksille. Koska asennushakemiston kopiointi ei ole tuettu tapa sovelluksen asentamiseksi, tätä toimintatapaa ei voitu käsitellä toimenpide-esityksessä mahdollisten epävakausongelmien takia. Mikäli asennus olisi onnistunut profiilikohtaisesti, tämä olisi ollut yksi esitetyistä käyttömalleista.

7 YHTEENVETO

Tämän opinnäytetyön tavoitteena oli tarkastella vaihtoehtoja Legacy-sovellusten yhteensopivuuden takaamiseksi oikeushallinnon tietotekniikkakeskuksen vaihtaessa käyttöjärjestelmän XP:stä Windows 7:ään. Työ oli osa meneillään olevaa projektia ja liittyi erityisesti sen testausvaiheeseen. Ennen varsinaisen projektin käynnistämistä toimeksiantaja oli teettänyt esiselvityksen, jossa tarkasteltiin alustavasti sovellusten yhteensopivuutta. Esiselvitysraportti asetti selkeät reunaehdot opinnäytetyössä käsiteltäville ympäristöille ja tekniikoille ja ensimmäinen testauskierros osoitti työssä käsiteltävät sovellukset.

Yhteensopivuustyökalujen valikoima on melko laaja ja ne sopivat niin pienten kuin suurten IT-infrastruktuurien tarpeisiin. Microsoft on panostanut merkittävästi sovellusyhteensopivuuteen julkaisemalla työkaluja ongelmatapausten korjaamiseksi. Yhteensopivuuden voidaankin todeta olevan hyvällä tasolla. Työssä tarkasteltujen sovellusten avulla on mahdollista rakentaa käyttömalleja alkaen hyvin pienistä korjauksista, päättyen suuriin, tuhansia henkilöitä palveleviin järjestelmiin. Ongelmallisille Legacy-sovelluksille pyrittiin etsimään vaihtoehtoisia ratkaisuja, eikä tarkoituksena ollut lopettaa ensimmäisen vaihtoehdon löydyttyä. Useampien vaihtoehtojen tarkastelulla pyrittiin takaamaan parhaan mahdollisen käyttömallin löytyminen. Tämä toteutuikin jälkimmäisen sovelluksen kanssa, tarjoten käyttäjälle mahdollisuuden muokata henkilökohtaisia asetuksiaan sekä käyttöliittymää.

Virtualisoinnin hyödyntäminen oli alkuperäisessä suunnitelmassa vahvasti esillä, mutta suurin syy sen poisjäämiselle oli toimivuusongelmien laatu ja Windows 7 kehittynyt sovellusyhteensopivuus. Käyttömalleja valittaessa ei nähty perustelluksi ratkaisuksi rakentaa monimutkaista testiympäristöä ja lähteä lopulta viemään tätä tuotantoon, koska sovellukset saatiin toimimaan huomattavasti pienemmillä resursseilla. Lopulliset käyttömallit on esitetty opinnäytetyön toimeksiantajalle erillisinä dokumentteina.

LÄHTEET

Application Compatibility Best Practices for Remote Desktop Services. 2009. Microsoft. Viitattu 7.12.2012. http://www.microsoft.com/en-us/download/details.aspx?id=18704

Beaubouef, G. 2009. Maximize Your Investment: 10 Key Strategies for Effective Packaged Software Implementations. Birmingham: Packt Publishing Ltd.

Esiselvitys. 2011. Oikeusministeriön hallinnonalan työasemien käyttöjärjestelmän uudistamisen esiselvitys.

Finn, A., Gibson, D. & Van Surksum, K. 2011. Mastering Windows 7 Deployment. Indianapolis: Wiley Publishing, Inc.

Infrastructure Planning and Design. Microsoft® Enterprise Desktop Virtualization Version 1.2. 2011. Viitattu 30.11.2012. http://download.microsoft.com/download/5/B/C/5BC966BC-47D8-41DF-95F2-FA9A2D816258/MED-V.zip

Inside Windows Vista User Account Control. 2007. Microsoft. Viitattu 11.9.2012. http://technet.microsoft.com/en-us/magazine/2007.06.uac.aspx

JUHTA - Julkisen hallinnon tietohallinnon neuvottelukunta 2009. JHS 173 ICT-palvelujen kehittäminen: Vaatimusmäärittely. Viitattu 25.9.2012. http://docs.jhs-suositukset.fi/jhs-suositukset/JHS173/JHS173.pdf

Myers, G., Badgett, T., Thomas, T. & Sandler, C. 2004. The Art of Software Testing. 2. osin uud. p. Hoboken: John Wiley & Sons, Inc.

Nurminen, M., Reijonen, P. & Vuorenheimo, J. 2002. Tietojärjestelmän organisatorinen käyttöönotto: kokemuksia ja suuntaviivoja. Turun kaupungin terveystoimen julkaisuja, Sarja A, Nro 1/2002. Turku.

Oikeushallinnon tietotekniikkakeskus. 2011. Oikeusministeriö. Viitattu 18.6.2012. http://www.om.fi/Etusivu/Ministerio/Hallinnonala/Oikeushallinnontietotekniikkakeskus

Panek, W. 2010. Microsoft Windows 7 Administration Instant Reference. Indiana: Wiley Publishing, Inc.

Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. Jyväskylä: Docendo.

Saleh, K. A. 2009. Software Engineering. Fort Lauderdale: J. Ross Publishing, Inc.

Thurrott, P. 2008. Windows Vista Secrets: SP1 Edition. Indianapolis: Wiley Publishing, Inc.

Understanding Application Compatibility. N.d. Windows. Viitattu 28.11.2012. http://technet.microsoft.com/en-us/library/ee449431

Tuotetuki: Common file and registry virtualization issues in Windows Vista or in Windows 7. N.d. Microsoft. Viitattu 5.11.2012. http://support.microsoft.com/kb/927387

Watkins, J. 2001. Testing IT: An Off-the-Shelf Software Testing Process. Port Chester: Cambridge University Press.

What's New in User Account Control. 25.6.2009. Microsoft. Viitattu 5.11.2012. http://msdn.microsoft.com/en-us/library/dd446675

Windows 7, Windows 7 SP1, and Windows Server 2008 R2 Application Quality Cookbook. 21.10.2010. Microsoft. Viitattu 20.8.2012. http://archive.msdn.microsoft.com/Windows7AppQuality/Release/ProjectReleases.aspx?ReleaseId=5095

Windows lifecycle fact sheet. 2012. Microsoft. Viitattu 5.11.2012. http://windows.microsoft.com/en-US/windows/products/lifecycle

Liite 1

TESTAUSLOMAKE

Oikeushallinnon Tietotekniikkakeskus Hämeenlinna

Sovellustestauspöytäkirja

Windows 7 työaseman testauslomake
ver. 2.3

Testauksen lähtötiedot

Työaseman esitarkastus ennen testausta:

	Työasema on kytketty verkkoon ja testaustila- ja aika täyttävät tarvittavat kriteerit
	Käytettävät testaustunnukset ovat määritelty ja toimivat
	Tarvittavista menettelyistä on sovittu ulkopuolisen palveluntarjoajan kanssa (tarvittaessa)

Testaustiedot

Testauspaikka	
Testauspäivä ja -aika	
Testauksen suorittaja(t)	

Työasematiedot

Koneen merkki ja malli	
Työaseman nimi	
Sektori	

Käyttöjärjestelmä	
Bittisyys (32/64)	

Tarvittava testauspaketti on toimitettu:

	ZenWorks (Järjestelmänhallinta)
	System Center Configuration Manager (SCCM)) (Järjestelmänhallinta)
	Asennettu muusta lähteestä
	Asennettu manuaalisesti CD:ltä / DVD:ltä / muistitikulta / palvelimelta

Näyttötilat (resoluutio) millä testattu

1024*768	1280*768	1440*900	1920*1200	
Muu, mikä				

Tuplanäyttö tai muu ominaisuus

Kyllä	Ei

Muu laite tai ajuri

Kyllä	Ei

Testattavan ohjelman asennus

Työasema käynnistetty uudestaan asennuksen jälkeen (tarvittaessa)

Työaseman lokit

Testattava sovellus

VHL

Vati	Effica	Winpos	Aromi	*Joku muu, mikä*

OH

Sakari	Tuomas	Uljas	Rajsa	Edward
Romeo				

OM

Oskari	Eutori	Senaattori		*Joku muu, mikä*

Ohjelman käynnistys (kirjautumisikkunaan asti)

Onnistui	Ei onnistunut

Kirjautuminen sovellukseen

Onnistui	Epäonnistui	Ei vaadi kirjautumista

Sovellusliitännäisen tarkistus (Esim Adobe Acrobat, Java, Notes tms)

Onnistui	Epäonnistui	Ei vaadi

Hakutoimisto (esim diaarihaku)

Onnistui	Epäonnistui	Ei mahdollisuutta

Tulostus sovelluksesta

Onnistui	Epäonnistui	Ei mahdollisuutta tulostaa

Ohjelmien ikkunointi, näkymät

Normaalit	Poikkeavat	Testaajalla ei riittävää tietoa

Yhteiskäyttö

1. Leikepöytä (leikepöydän käyttö eri ohjelmissa)samanaikaisesti, kuin testattava sovellus auki. Merkitse, mitä ohjelmia ollut auki, ja vie leikepöydältä sisältöä toiseen ohjelmaan.

Open Office Writer	Open Office Calc	Open Office Impress
MS Office Word 2000	MS Office Execel 2000	MS Office PowerPoint 2000
MS Office Word 2003	MS Office Excel 2003	MS Office PowerPoint 2003
MS Office Word 2003 PRO	MS Office Excel 2003 PRO	MS Office PowerPoint 2003 PRO

Tiedostojen avautuminen (tarvittaessa)

3. Eri tiedostomuotojen avautuminen:

ods	odp	odt	doc	ppt /pps	xls	docx
pptx	xlsx	gif	jpeg/jpg	bmp	wav	mp3
mid	avi	swf	rtf	pdf		

Ohjelman sulkeminen

Onnistui	Ei onnistunut