OULUN AMMATTIKORKEAKOULU

Niko Karjalainen

ICT-omaisuudenhallintajärjestelmän toteutus

ICT-omaisuudenhallintajärjestelmän toteutus

Niko Karjalainen
Opinnäytetyö
Syksy 2014
Tietojenkäsittelyn koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Järjestelmäasiantuntemus

Tekijä: Niko Karjalainen
Opinnäytetyön nimi: ICT-omaisuudenhallintajärjestelmän toteutus
Työn ohjaaja: Jukka Kaisto
Työn valmistumislukukausi- ja vuosi: Syksy 2014 Sivumäärä:43+7

Tämän opinnäytetyön aiheena oli ICT-omaisuudenhallintajärjestelmän toteutus. Toteutukseen kuului järjestelmän suunnittelu, kehitys ja käyttöönotto. Aihe opinnäytetyölle löytyi Meka Pro Oy:lla ammattiharjoittelujakson aikana kesällä 2014.

Tämä opinnäytetyö esittelee järjestelmän tekniset kehitysmenetelmät sekä teoriapohjan ja käytännöt joiden mukaan opinnäytetyössä tehty ICT-omaisuudenhallintajärjestelmä kehitettiin. Teoreettinen pohja perustuu useisiin alan kirjoituksiin ja oppaisiin.

Järjestelmän suunnittelu ja kehitys aloitettiin tarvekartoituksella, jossa käytiin järjestelmän vaatimukset läpi. Tämän jälkeen suunnittelin tietokannan, joka tukee järjestelmän tarvittavia vaatimuksia. Tietokantasuunnittelun jälkeen toteutin automaattisen tiedonkeruumenetelmän. Viimeisenä kehitysvaiheena oli raportoinnin ja graafisten toimintojen suunnittelu.

Opinnäytetyön tuloksena syntyi työn alussa määriteltyjen vaatimusten mukainen järjestelmä, johon kuuluu automaattinen tiedonkeruu työasemien ja asennettujen ohjelmistojen osalta, tietokanta, sekä graafinen käyttöliittymä ja raportointi. Työn tulosten avulla saatiin automatisoitua ja tehostettua ohjelmistolisenssien ja ICT-omaisuuskannan hallintaa.

Asiasanat: ohjelmointi, tietokanta, ohjelmistokehitys

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Business information Systems
Option of Computer Systems Expertise

Author: Niko Karjalainen
Title of Bachelor´s thesis: IT Asset Management software implementation
Supervisor: Jukka Kaisto
Term and year of completion: Autumn 2014 Number of pages: 43+7

The purpose of this thesis was to implement an IT asset management software. The implementation of the software included design, development and deployment. The topic for the thesis was identified during the professional training at Meka Pro Oy in summer 2014.

This thesis introduces the technical methods used in the development process of the software. It also presents theoretical background and best practices that the developed IT asset management software is based on. The theoretical background of this thesis involves multiple blogs and books in the field of information technology.

The software development and design began with needs assessment. After that I designed a database, which supports the needs of the software. When the database design was finished, I developed the automatized data gathering method for the software. The last part of the development process was to design reporting and graphical functions.

The final result of this thesis was a software that meets the requirements defined at beginning of this thesis. The software includes automatic data gathering concerning the company's workstations, software installations, database and graphical interface. With the help of the results, it was possible to manage software licenses and asset registry more efficiently.

Keywords: programming, database, software development

SISÄLLYS

1 JOHDANTO

ICT-omaisuudenhallinta on tärkeässä roolissa yrityksissä. Se on strategisesti tärkeä prosessi tietohallinnolle, sekä laskutuksen, että tukipalveluiden hallinnan kannalta. Yrityksissä tapahtuu jatkuvasti muutoksia ohjelmisto- ja laitepuolella. Näiden hallinta on merkittävästi helpompaa ICT-omaisuudenhallintajärjestelmillä.

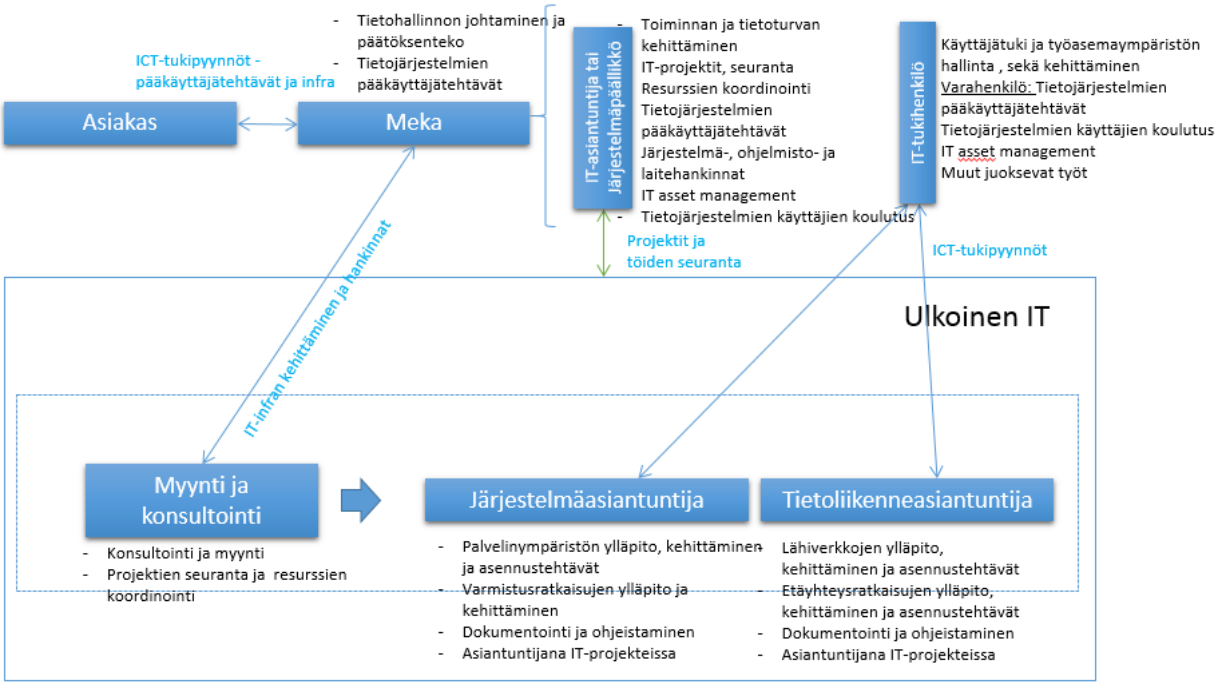
Hyvin suunniteltu ICT-omaisuudenhallintajärjestelmä mahdollistaa yrityksien ICT-omaisuuden ajan tasalla pysymisen muun toiminnan kanssa. Se toimii tukena koko organisaation toiminnalle, ja parantaa työn laatua ja vähentää työtunteja.

Tämän opinnäytetyön tarkoituksena on suunnitella ja toteuttaa ICT-omaisuudenhallintajärjestelmä Meka Pro Oy:lle. Järjestelmän kehitys rajattiin työasemien ja ohjelmistolisenssien automatisoituun tiedonkeruuseen ja -hallintaan.

Opinnäytetyön aihe sai alkunsa työharjoittelussa kesällä 2014 yrityksen järjestelmäasiantuntija kanssa käydyissä IT-palavereissa. Palavereissa suunniteltiin ICT-omaisuudenhallintajärjestelmän hankkimista ulkopuoliselta taholta. Sopivaa järjestelmää ei kuitenkaan löydetty, ja opinnäytetyön tekijällä riitti paljon omaa mielenkiintoa ja osaamista aiheeseen. Näiden johdosta yritys päätti toteuttaa ICT-omaisuudenhallintajärjestelmän omatoimisesti.

Meka Pro Oy on yksi Pohjoismaiden johtavia kaapeliteiden valmistajia. Tuotteisiin kuuluu kuuma-sinkityt teräksiset tikashyllyt, levyhyllyt, valaisinkiskot sekä alumiinirakenteiset johtokanavat ja pistorasiapylväät. Meka Pro Oy:n henkilöstömäärä on noin 100 henkilöä. (http://www.meka.eu/meka-pro-oy.html, hakupäivä 14.10.2014.)

Meka Pro Oy:n sisäiseen IT-organisaatioon kuuluu kolme henkilöä, tietohallintopäällikkö, järjestelmäasiantuntija ja IT-tukihenkilö. Erityisprojekteissa käytetään apuna ulkopuolisia konsultteja. IT-organisaation henkilöiden tarkemmat vastuualueet ovat kuvattuna kuviossa 1.



KUVIO 1. Meka Pro Oy:n IT-organisaatio.

Opinnäytetyön raportti esittelee järjestelmän tekniset kehitysmenetelmät sekä teoriapohjan ja käytännöt joiden mukaan Meka Pro Oy:lle tehty ICT-omaisuudenhallintajärjestelmä kehitettiin. Kappaleessa neljä kerrotaan järjestelmän suunnittelun, kehityksen, testauksen ja käyttöönoton vaiheista.

2 ITIL-VIITEMALLI

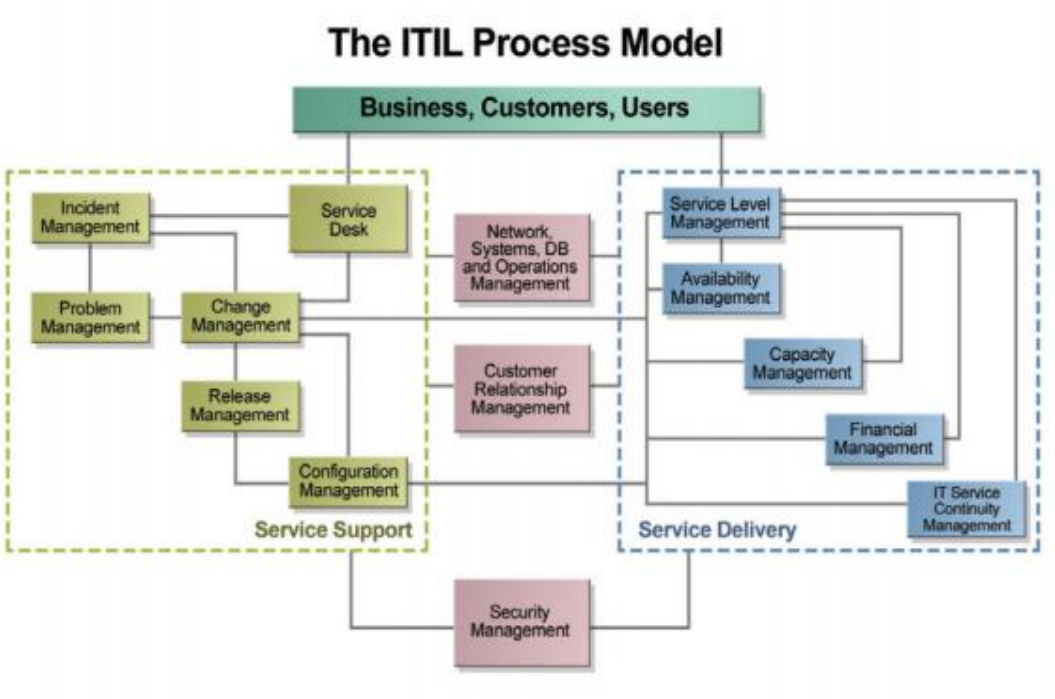
Information Technology Infrastructure Library (ITIL) on kattava parhaita IT-palveluiden käytäntöjä hyödyntävä viitemalli, ja sen kehitys aloitettiin Iso-Britannian hallituksen toimesta jo vuonna 1989. (Braden, S. 2005, 2.) ITIL on rekisteröity tuotemerkki, joka kuuluu Iso-Britannian hallitukselle (Hobbs 2011, 1).

ITIL-viitemallin käytännöt rakentuvat prosessien ympärille. Hartvaaran (hakupäivä 8.11.2014) mukaan prosessit ovat työtapahtumien sarjoja, jotka virtaavat tehtäväorganisaation läpi. ITIL-viitemallissa prosessi kuvataan sarjana toisiinsa liittyviä aktiviteetteja, joilla pyritään saavuttamaan tietty päämäärä. Prosessille määritetään tietyt syötetiedot ja tuloksena saadaan resursseja, joilla saadaan aikaan parempaa hallintaa, pienemmät kustannukset ja parempaa laatua. (Knowledge Transfer, hakupäivä 8.11.2014.)

2.1 ITIL-viitemallin historia

ITIL-viitemallista on kehitetty kolme versiota. Ensimmäinen versio ITIL-viitemallista sisälsi lukuisan määrän kirjoja ja ajatusmalleja parhaista käytännöistä. Vuonna 2000 julkaistiin ITIL V2, joka keskittyi tuki ja toimitusprosesseihin (katso kuvio 2). Se tiivistettiin kahdeksaan eri osa-alueeseen. Ne olivat palveluiden toimitus, palveluiden tuki, IT-infrastruktuurin hallinta, tietoturvan hallinta, liiketalouden näkökulma, applikaatioiden hallinta, ohjelmistojen hallinta ja palveluiden hallinnan toteutuksen suunnittelu. (McGunigle, hakupäivä 20.9.2014.)

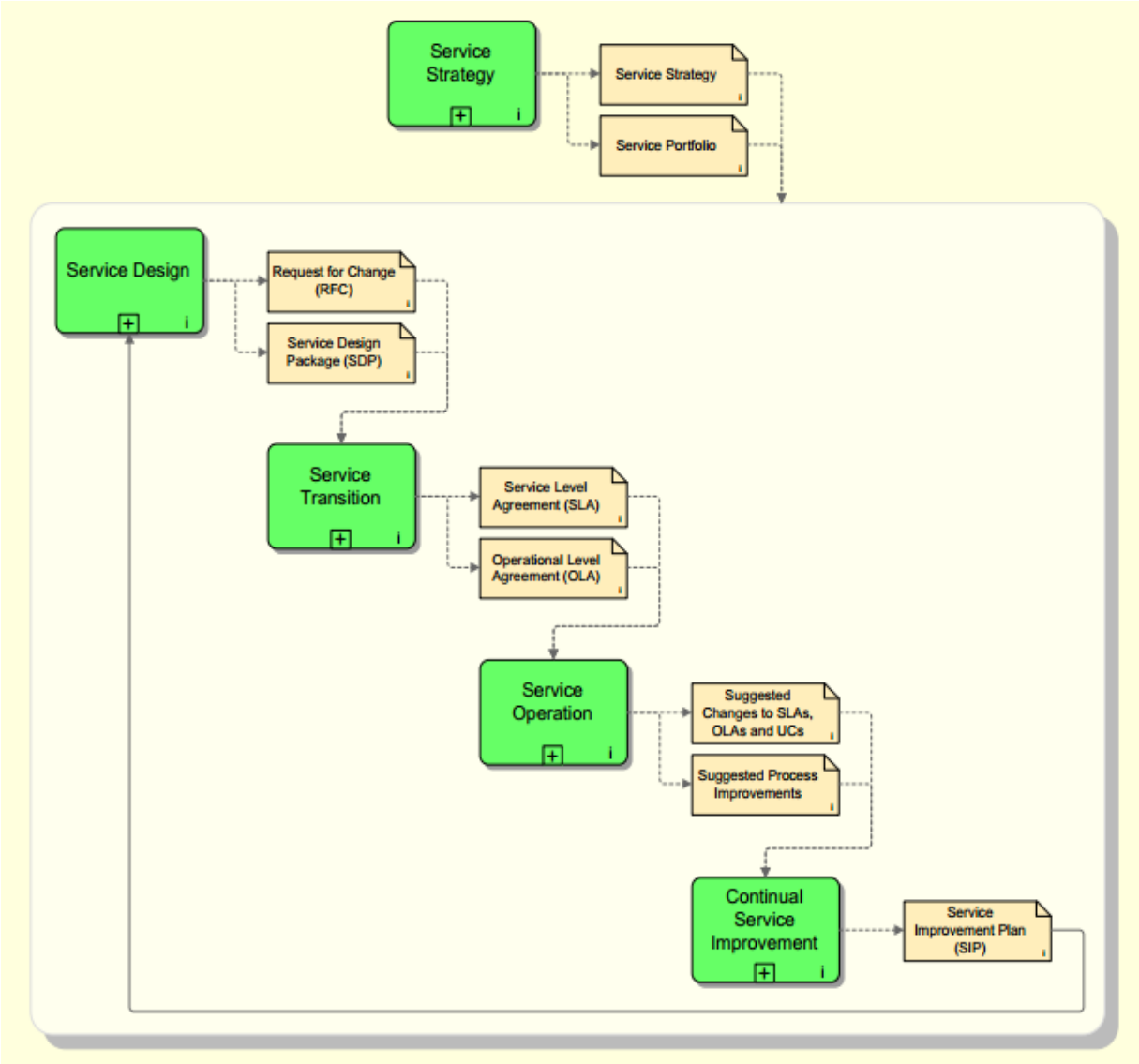
Kuvio 2 esittää yhteydet ITIL V2 -viitemallin avainprosessien välillä. Kaikki nämä prosessit toimivat yhteen keskenään, ja muutos yhdessä paikassa vaikuttaa kaikkialle. ITIL-viitemalli keskittyykin yhdistämään nämä vaiheet toisiinsa parhaiden käytäntöjen mukaisesti, jolloin organisaation IT-palveluita pystytään hallitsemaan tehokkaasti. Organisaatiossa täytyy tehdä paljon töitä, jotta nämä vaiheet saadaan toimimaan tehokkaasti keskenään. Prosessit etenevät johdonmukaisesti, ja jokainen prosessi saa syötetiedon ja antaa tuloksen seuraavalle prosessille aktiviteettien suorituksen jälkeen (Braden 2005, 3.)



KUVIO 2. ITIL V2 -prosessikuvio (Braden 2005, 3).

Viimeisin versio, ITIL V3 julkaistiin vuonna 2007 viiteen prosessiin tiivistettynä. Ne ovat palveluiden strategiat, palveluiden suunnittelu, palveluiden siirtyminen, palveluiden toiminta ja jatkuva palveluiden kehittäminen. (McGunigle, hakupäivä 20.9.2014.) ITIL V3 -viitemallissa ei ole omaa moduulia jokaiselle prosessille kuten ITIL V2 -viitemallissa. Tämän ansiosta pystytään hahmottamaan palveluiden täysi elinkaari strategisemmin (Turbitt, Strategic Thinking Over a Lifecycle of Service, hakupäivä 14.10.2014).

Kuvio 3 esittää ITIL V3 -prosessien elinkaaren. Prosessien elinkaaressa palveluita hallitaan niiden luomisesta päättymiseen saakka. Jokainen avainprosessi keskittyy tiettyyn vaiheeseen palveluiden elinkaaressa. Palveluiden strategiassa määritetään minkä tyyppisiä palveluita tarjotaan asiakkaille. Palveluiden suunnittelussa tunnistetaan palveluiden vaatimukset, keksitään uutta palvelutarjontaa, sekä kehitetään jo olemassa olevia palveluita. Palveluiden siirtymisessä rakennetaan ja otetaan käyttöön uudet tai muutetut palvelut. Palveluiden toiminnassa suoritetaan toiminnallisia tehtäviä. Jatkuvassa palvelun kehittämisessä pyritään oppimaan aiemmista onnistumisista ja virheistä, sekä kehitetään jatkuvasti palveluiden ja prosessien tehokkuutta. (IT Process Maps, hakupäivä 8.11.2014.)



KUVIO 3. ITIL V3 -viitemallin elinkaari (IT Process Maps, hakupäivä 8.11.2014).

ITIL V3 -viitemallissa pyritään luomaan yhteinen kommunikaatiokieli tietohallinnon ja muun liiketoiminnan väliin. Tällä toimintatavalla päästään tarkempiin ja tehokkaampiin päätöksiin koko liiketoiminnan kannalta. (Turbitt, A Quantum Leap in I.T.-Business Communication, hakupäivä 14.10.2014.) ITIL V3 -viitemallin prosesseilla mitataan selkeästi mitkä toimenpiteet, muutokset ja sijoitukset tuottavat lisäarvoa organisaatiolle. (Turbitt, Evolution not Revolution, hakupäivä 14.10.2014.)

2.2 ITIL V2 -viitemallin avainprosessit

Suurin osa IT-osastoista perustaa toimintansa edelleen ITIL V2 -viitemallin avainprosesseihin, vaikka uudet toimintatavat ovat jo saatavilla ITIL V3 -viitemallissa. Näillä avainprosesseilla saadaan hahmotettua miten toivutaan häiriöistä, miten ratkokaan ongelmat sekä miten muutokset implementoidaan organisaation toimintaan. (Hobbs 2011, luku 1, Operational Chaos.)

Häiriönhallinta

Hobbsin (2011, luku 1, Operational Chaos) mukaan häiriönhallinnassa pyritään hallitsemaan toipumista niihin järjestelmiin liittyen, joihin loppukäyttäjä luottaa jokapäiväisessä työssään. Tämä palvelu ei jostain syystä ole enää saatavilla, tai ei enää vastaa loppukäyttäjän odottamalla tavalla.

Häiriönhallinnan prosesseissa on tarkoituksena kirjata tämä ongelma ylös. Kirjauksen jälkeen suoritetaan sellaiset toimenpiteet, joilla järjestelmä saadaan taas mahdollisimman nopeasti siihen toiminnalliseen pisteeseen, että loppukäyttäjä voi taas jatkaa työntekoaan. Häiriöiden kirjauksessa kiinnitetään huomiota siihen miten loppukäyttäjä toi ongelman ilmi käyttäjätukeen. Kirjauksessa huomioidaan myös kuinka paljon meni aikaa järjestelmän toipumiseen ja mitä korjaavia toimenpiteitä tehtiin. (Hobbs 2011, luku 1, Operational Chaos.)

Ongelmanhallinta

Ongelmanhallinnassa tutkitaan mitkä asiat aiheuttavat ne häiriöt, joita kirjattiin häiriönhallintajärjestelmään, sekä kaikki muutkin organisaation ympäristössä tapahtuvat häiriöt. Ongelmanhallinnassa pyritään tuomaan kaikki mahdollinen ongelmiin liittyvä informaatio yhteen, jonka avulla sitten pyritään tuottamaan mahdollisesti pysyvät ratkaisut ongelmiin. Tällä tavoin pystytään ennaltaehkäisemään vakavammat ongelmat tulevaisuuden varalta, tai ainakin vähentämään niiden määrää organisaation sisällä. (Hobbs 2011, luku 1, Operational Chaos.)

Muutoksenhallinta

Muutoksenhallinnassa pyritään toteuttamaan muutokset organisaation ympäristöön mahdollisimman hallitulla tavalla. Muutoksien käyttöönoton vaiheet, kuten testaus ja mahdollisesti toipuminen dokumentoidaan. Muodostaessa mahdollisimman kokonaisvaltaista näkymää kaikista mahdolli-

sista muutoksista ja yksityiskohdista, pystytään muutokset tuomaan infrastruktuuriin juurikaan häiritsemättä organisaation toimintaa. (Hobbs 2011, luku 1, Operational Chaos.)

Toipuminen on tärkeä osa muutoksenhallintaa, sillä kaikki ei kuitenkaan mene aina virheittä, vaikka asia olisi suunniteltu kuinka perusteellisesti. Tällöin otetaan dokumentoitu toipumissuunnitelma käyttöön, ja jatketaan toimintaa vanhalla mallilla. (Hobbs 2011, luku 1, Operational Chaos.)

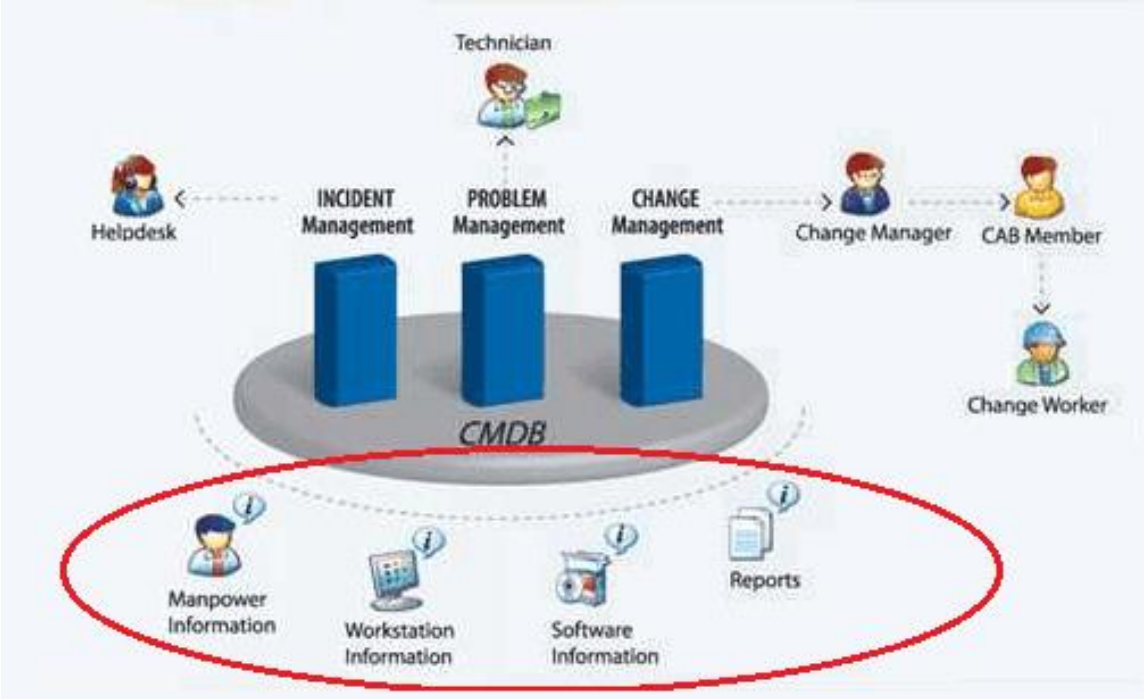
Konfiguraationhallinta

Konfiguraationhallinnassa kerätään kaikki IT-prosesseihin kuuluva tieto yhteen tietokantaan. Siten saadaan kaikkien IT-prosessien väliset yhteydet helpommin selvitettyä. Konfiguraatiotietokannan rakennus aloitetaan aina ICT-laitteistotiedon keräämisestä sinne. (Hobbs 2011, luku 1, Operational Chaos.) Asioiden yhteyksien hahmottaminen on konfiguraatiotietokannan tärkein tehtävä. Yhteyksien avulla pystytään palvelemaan toimintaa, ei pelkästään hahmoteta mitä omistetaan ja mitä dokumentteja organisaatiolla on olemassa.

Konfiguraatiohallinnalla ylläpidetään tietoa. Kannassa oleva tieto tulee poistaa tai päivittää vastaamaan todellista tilannetta. (Thejendra 2014, luku 9, Service Asset and Configuration Management.) Esimerkiksi jos organisaatioon hankitaan uusi ohjelmistoversio vanhan tilalle, niin tulee vanhan ohjelmiston merkinnät poistaa kannasta, ja uudet päivittää tilalle.

3 ICT-OMAISUUDENHALLINTA

ICT-omaisuudenhallinnan tarkoituksena on kerätä kaikki tieto niistä laitteista ja ohjelmistolisenseistä, joita organisaatio omistaa. Omaisuudesta hahmotetaan niihin kuuluvat yksityiskohdat ja käyttäjät. ICT-omaisuudenhallinta tuo paljon rahallisia ja teknisiä etuja. Vaikkakin hyvän omaisuudenhallinnan saavuttaminen vaatii paljon resursseja. (Hobbs 2011, luku 2, Basic Logistics and Inventory.) ICT-omaisuudenhallinta on yksi monista aktiviteeteistä ITIL-viitemallin konfiguraatiohallintaprosessissa (katso kuvio 4).

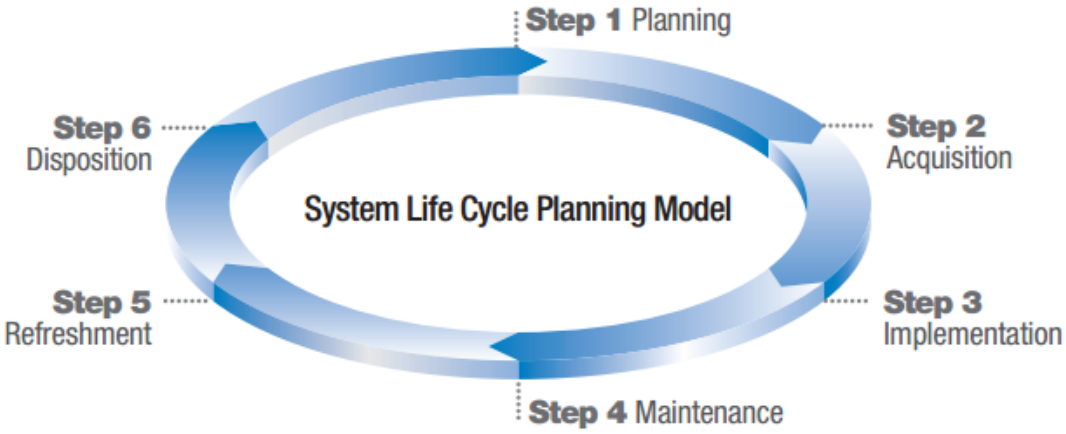


KUVIO 4. ICT-omaisuudenhallinta ITIL-viitemallin konfiguraatiohallintaprosessin aktiviteettina (ManageEngine, hakupäivä 8.1.2014).

3.1 Laitteiden elinkaari

Laitteiden elinkaaren hallinta on suuressa roolissa ICT-omaisuudenhallinnassa. (Hobbs 2011, luku 3, Keeping It Fresh). Elinkaareen kuuluu hankinnan suunnittelu, hankinta, käyttöönotto, huol-

lot, päivitykset ja hävitys (katso kuvio 5). Kaikkia näitä vaiheita pystytään hallitsemaan ICT-omaisuudenhallinnalla. *(QFIX, hakupäivä 8.1.2014.)*



KUVIO 5. Laitteiden elinkaaren suunnittelumalli (QFIX, hakupäivä 8.1.2014).

Laitteiston hankinta oikeaan aikaan säästää rahaa ja pitää loppukäyttäjät tyytyväisenä. Pitää selvittää riittävätkö työasemien tehot ajamaan tämän hetkisiä liiketoimintasovelluksia. Vanhan laitteen pitäminen toiminnassa saattaa vain lisätä kustannuksia. Loppukäyttäjä ei välttämättä ole tyytyväinen vanhaan laitteeseensa. Tämä lisää tukipyyntöjä ja heikentää työmoraalia, jolloin myös IT-osaston työtuntien määrä lisääntyy. Näiden ongelmien yhteyksien seuraamiseen on hyvä käyttää automaattisia raportteja, jotka saadaan tehtyä laitteistokannasta. Esimerkkinä raportti tietokoneista joissa on alle kolme gigatavua keskusmuistia ja tietty ohjelmisto. Saattaa kuitenkin olla työtehtäviä, joiden suorittamiseen riittää viisi vuotta vanhakin työasema. Näissä tilanteissa tulee miettiä yksityiskohtaisesti jatketaanko vanhojen laitteiden huoltosopimuksia, vai tuleeko halvemmaksi hankkia uusi laite. Mitä pitempään samoja laitteita pidetään käytössä, niin sitä vähemmän tarvitsee tehdä esimerkiksi valmiita asennusimageja tietokoneita varten. (Hobbs 2011, luku 3, Keeping It Fresh.)

Uusien laitteiden hankkimista varten tulee tehdä suunnitelma. Koneiden hankkiminen tietyn prosessin mukaan tulee halvemmaksi kuin ostaa koneita aina suuri määrä kerrallaan suunnittelematta hankintaa sen suuremmin. Laitteistokannasta voisi tehdä esimerkiksi raportin koneista, joista on takuu umpeutumassa kuuden kuukauden sisällä. Kun uusia koneita otetaan käyttöön asteittain, niin toiminnan häirintä pysyy mahdollisimman pienenä yksittäisillä liiketoiminnan alueilla.

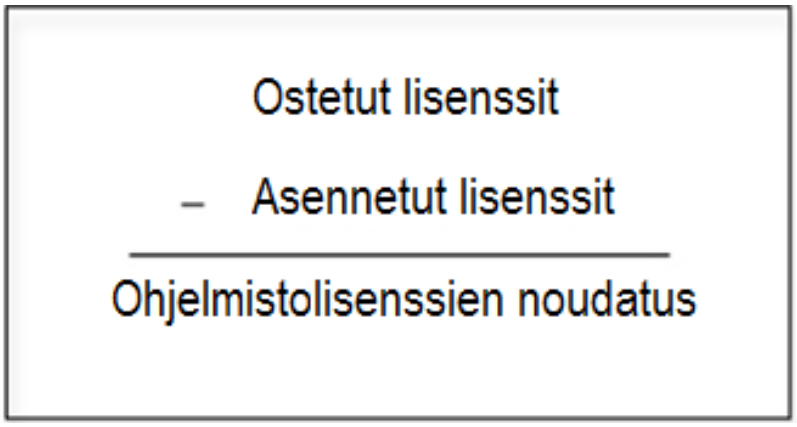
Keskitetty hankinta on laitteiden selvin reitti organisaatioon. Tulee kuitenkin huomioida että ihan kaikki laitteet eivät välttämättä tule IT-osaston kautta organisaatioon. Tätä varten kannattaa tehdä tarkat ohjeet. Laitteiden hankinnassa kannattaa huomioida että tuen tarjoaminen laitteille on mahdollista. (Hobbs 2011, luku 3, Keeping It Fresh.)

Laitteiden poistossa pätevät samat asiat kuin hankkimisessa. Poiston pitää tulla laitteistokannan hallitsijan kautta. (Hobbs 2011, luku 3, Keeping It Fresh.) Kun laitteistonkannan hallitsija poistaa laitteen, niin poisto tapahtuu tietoturvallisesti. Kovalevy tyhjennetään ja tuhotaan fyysisesti. Laitteista irrotetaan yritykseen liittyvät tarrat ja muut tunnistetiedot. Tämän jälkeen laitteet kierrätetään ympäristöystävällisesti.

Laitteistonhallinta on onnistunut silloin, kun kannan ajantasaisuus on kohdillaan. Laitteistokannasta ei ole juuri hyötyä, jos sieltä puuttuu uusia laitteita tai jos siellä on jo poistuneita laitteita. Laitteiston käyttöönottoa ja poistoa varten kannattaa siis luoda tietyt prosessit ja toimintatavat. Tätä varten kannattaa myös ehkä hyödyntää jotain automatisoitua sovellusta. (Hobbs 2011, luku 3, Keeping It Fresh.)

3.2 Lisenssien hallinta

Lisenssien hallinta yritystoiminnassa on äärimmäisen tärkeää. Sääntöjen mukainen toiminta ilman kunnollista lisenssien hallintaa on hankalaa. Jokaiselle asennetulle ohjelmistolle täytyy olla laillinen käyttöoikeus. Tavoitteen saavuttamiseksi tarvitaan täysi lista asennetuista ohjelmista jokaiselle yksittäiselle koneelle ja lista lisenssien määrästä tiettyä ohjelmistoa kohden. Ohjelmistojen ja lisenssien määrän on oltava yhdenmukaisia, asennettujen ohjelmistojen määrä ei saa ylittää ostettujen lisenssien määrää. Näiden kahden välistä tasapainoa kutsutaan ohjelmistonlisenssin noudattamiseksi (katso kuvio 6). (Hobbs 2011, luku 4, Licensed To Bill.)



KUVIO 6. Ohjelmistolisenssien noudatus yhtälönä (Hobbs 2011, luku 4, Licensed To Bill).

Asennettujen ohjelmistojen seuraamista varten tulee olla kokonaisvaltainen tieto fyysisistä laitteista. Fyysisiltä laitteilta pystytään tutkimaan asennettujen ohjelmistojen määrä. Myös ostettujen ohjelmistolisenssien tulee olla tallennettuna lisenssirekisteriin, jotta asennettuja ja ostettuja ohjelmistoja pystytään tarkkailemaan. Hyvänä linjauksena pidetään, että viimeisen viiden vuoden lisenssiostot pidetään kirjattuna. Ohjelmistoista kirjataan valmistaja, ohjelmiston nimi, versionumero, toimittaja sekä osto-aika. Myös ohjelmistojen tukisopimukset on hyödyllistä kirjata ylös. (Hobbs 2011, luku 4, Licensed To Bill.)

Edellä olevien asioiden täytyttyä voidaan tarkkailemaan ovatko kaikki asennetut ohjelmistot lisenssin alaisia. Voidaan tutkia, että paljon lisenssejä on jäljellä tai pitääkö joitain lisenssejä hankkia. Näin toimittaessa on helppo ryhtyä toimenpiteisiin tarvittaessa. Löydettäessä ohjelmistoasennuksia ilman riittävää määrää lisenssejä, vaihtoehtona on ostaa tarvittava määrä lisenssejä, tai tarkastella löytyykö työasemilta tarpeettomia asennuksia. (Hobbs 2011, luku 4, Licensed To Bill.)

Ohjelmistoillakin on elinkaari, joten useasti kun ollaan siinä tilanteessa että lisenssit alkavat olla lopussa, niin tulee tarkastella että olisiko jo syytä päivittää ohjelmisto uudempaan versioon. Lisenssejä vanhaan versioon ei aina välttämättä ole saatavissa. Tällöin kuitenkin voidaan keskustella erikseen ohjelmiston valmistajan tai jälleenmyyjän kanssa, jos ei esimerkiksi ole varoja päivittää kaikkia lisenssejä uudempaan versioon. (Hobbs 2011, luku 4, Licensed To Bill.)

3.3 Loppukäyttäjien rooli

Uusien loppukäyttäjiä koskevien sääntöjen luonti on melko todennäköistä omaisuudenhallintajärjestelmää käyttöönotettaessa. Yleensä nämä säännöt ovat oikeuksia ja toiminnollisuuksia rajoittavia. Hyvin perusteltujen sääntöjen kanssa selvitään kuitenkin isommilta selkkauksilta. (Hobbs 2011, luku 6, Dealing With The Locals.)

IT-tuen käytäntöjen ohjeistus

IT-tuen käytäntöjen määrittelyä varten tulee tunnistaa kaikki laitteet ja ohjelmistot, jotka ovat IT-osaston tuen ulkopuolella. Tämän voi selventää hyvin kertomalla loppukäyttäjälle, että kyseinen laite tai ohjelmisto ei lisää arvoa organisaation toiminnalle. Tällöin sitä voidaan pitää ongelmana organisaatiolle. (Hobbs 2011, luku 6, Dealing With The Locals.) Tuetut laitemallit ja ohjelmistot voidaan tunnistaa tutkimalla kokonaisvaltaista ICT-omaisuuskantaa, tarkistamalla esimerkiksi että mitä malleja on hankittu aiemmin. Näin voidaan keskittää tukea pienempään alueeseen jolloin IT-osaston työtunnit vähenevät ja käyttäjäkokemukset paranevat.

Hankintakäytännön ohjeistus

Hankintakäytännöistä tulee ilmoittaa kaikille kustannuksista vastaaville henkilöille. Kaikki IT-hankinnat tulee tehdä IT-osaston budjetista, jotta saadaan tarkka käsitys menoista, ja saadaan ylläpidettyä omaisuuskanta. Hankintoja varten on myös hyödyllistä luoda jonkinlainen kommunikointikanava, jonka kautta hankintapyyntöjä voidaan tehdä. (Hobbs 2011, luku 6, Dealing With The Locals.) Esimerkiksi tukipyyntöjärjestelmään voidaan helposti luoda oma osio hankintojakin varten.

Laitekatalogi

Loppukäyttäjille tulee tehdä lista tuetuista laitteista. Listan teko mahdollistuu ohjelmistokannan perusteella, sillä kaikki liiketoimintaohjelmistot eivät välttämättä tue kaikkia laitteita. Esimerkiksi mobiililaitteiden ohjelmistotuet ovat hyvin usein ristiriitaisia. Tulee huomioida, että organisaation eri osastoille laitelistat saattavat olla erilaisia, koska eri osastoilla on yleensä käytössä eri ohjelmistot. (Hobbs 2011, luku 6, Dealing With The Locals.)

IT-tuen rajojen määrittely

IT-tuen rajat ja vastuut on hyvä määritellä omassa dokumentissaan, joka on kaikkien loppukäyttäjien saatavilla. Dokumentissa tulee selvitä miksi ja miten valvontaa lisätään, esimerkiksi kertomalla IT-palveluiden tason nousevan ja standardoituvan kyseisillä toimenpiteillä. On tärkeää selventää että, että ne laitteet joita ei ole määritelty laitekatalogiin, eivät saa tukea. Nämä toimenpiteet parantavat toimintaa koko organisaation tasolla, ja koko henkilökunta hyötyy loppujen lopuksi. (Hobbs 2011, luku 6, Dealing With The Locals.)

IT-tuen rajat käyttäjien omien laitteiden käytön osalta on myös hyvä tuoda ilmi. Rajoitetaan mitä ohjelmistoja käyttäjät voivat asentaa omille laitteilleen. Käyttäjien omien laitteiden kohdallakaan ei kannata tehdä poikkeusta laitetuen puolesta. Tuen antaminen satunnaisille laitteille kuluttaa enemmän työaikaa kuin pienellä määrälle laitemalleja, joiden vianhausta löytyy jo kokemusta.

3.4 Hyödyt

Kunnollisella omaisuudenhallintajärjestelmällä pystytään harjoittamaan liiketoimintaa ilman isompia yllätyksiä. Nykypäivän IT on niin laaja käsite, että vaikka säästöt ovat pieniä yhdellä osa-alueella, niin kokonaisuudessaan säästöt kasvavat kuitenkin merkittäväksi koko liiketoiminnan kannalta. Omaisuudenhallintaprojekteilla saadaan kunnollisesti käyttöönotettuna koko organisaation henkilökunta ajattelemaan asiat uudelta kannalta. Juuri ihmisethän ovatkin kaikista tehokkaimpia työkaluja ja lähteitä asioiden hoitamiseen. (Hobbs 2011, luku 7, Avoiding Expensive Mistakes.)

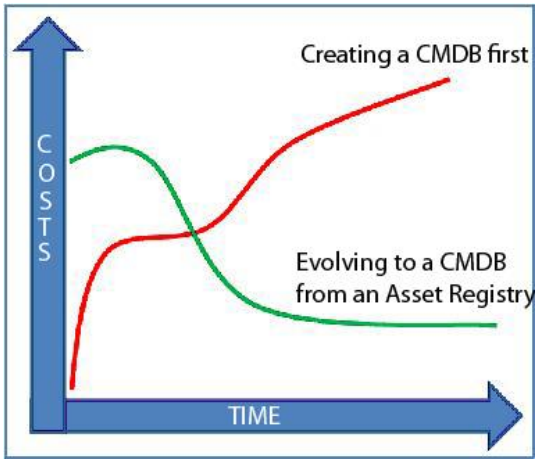
ICT-omaisuudenhallinta on myös tekniseltä näkökulmalta mukava työkalu päivittäiseen työskentelyyn. Tietokoneisiin ja ohjelmistoihin liittyvät työkulut ja dokumenttien käsittelyt helpottuvat, sillä asioiden väliin voidaan luoda yhteyksiä. Esimerkiksi dokumentille voidaan määrittää dokumenttienhallintajärjestelmään kenttä tietokone, ja tähän kenttään valitaan dokumenttiin kuuluva tietokone. Tämän jälkeen voidaan mennä tähän kyseiseen tietokone-objektiin järjestelmässä, ja tarkistaa mitä dokumentteja tietokoneesta on.

4 ICT-OMAISUUDENHALLINTAJÄRJESTELMÄN KEHITYS

Yrityksen IT-osastolla on suunnitelmissa ottaa ITIL-viitemalli käyttöön. Tämä on pitkä prosessi ja mielessä ei ole tehdä heti jotain valmista, vaan saada aikaan jatkuvaa kehitystä aikaan käyttäen ITIL-viitemallin prosesseja ja ajatusmallia.

Edellä mainitun myötä toimeksiantajan kanssa aloitettiin konfiguraationhallintatietokannan (CMDB) suunnittelu ja kehitys. Suunnitteluvaiheessa mietitään mitä karsitaan pois ja mikä on tärkeää. Kaikkea tietoa ei kannata tallentaa, ainoastaan ne asiat jotka tuovat arvoa yritykselle. Tietosisällön hahmotuksen jälkeen suunniteltiin tarkemmin tarvitaanko uusia järjestelmiä tukemaan konfiguraationhallintatietokantaa, toteutetaanko itse, hankitaanko vai pystytäänkö hyödyntämään jo olemassa olevia järjestelmiä (Thompson, hakupäivä 25.9.2014).

Opinnäytetyö keskittyy konfiguraationhallintatietokanta-projektissa omaisuushallintajärjestelmän kehitykseen työasemien ja ohjelmistolisenssien osalta. Omaisuushallinnan avulla seurataan laitteiden elinkaarta, jonka ansiosta saadaan tehtyä laite- ja lisenssihankinnat optimaaliseen aikaan. (Hyvönen, hakupäivä 25.9.2014.) Ilman tietoa organisaation omaisuudesta, ei konfiguraationhallintatietokantaa voida lähteä rakentamaan, koska olemassa olevat asiat pitää tunnistaa, ennen kuin niiden väliin voidaan luoda yhteyksiä (katso kuvio 7). (Thompson, hakupäivä 25.9.2014).



KUVIO 7. CMDB kehitys ilman rekisteriä omaisuudesta (Thompson, hakupäivä 25.9.2014).

4.1 Tarvekartoitus

Tarvekartoituksessa kerättiin ne tiedot, mitkä vaikuttivat olennaiselta yrityksen toiminnan kannalta. Ne laite-, ohjelma- ja käyttäjätiedot kirjattiin, joita omaisuustietokantaan halutaan tallentaa. Suunniteltiin tarkkaan mistä tiedoista voi olla hyötyä, turhaa tallentaa semmoista tietoa mitä ei tulla koskaan hyödyntämään mihinkään. Tarvittavia tietoja varten selattiin vanhoja dokumentteja, ja ajateltiin tekniseltä näkökulmalta mistä tiedoista voi olla hyötyä tukipyyntöjen ratkaisemisessa. Kerättävien tietojen tulee myös antaa tarvittava tietopohja kannasta luotaville raporteille. Järjestelmä tulee olemaan tukena käyttäjätuelle, tietohallintopäällikölle hankintoja varten sekä asioiden ja tietojen yhteyksien hahmottamiseen.

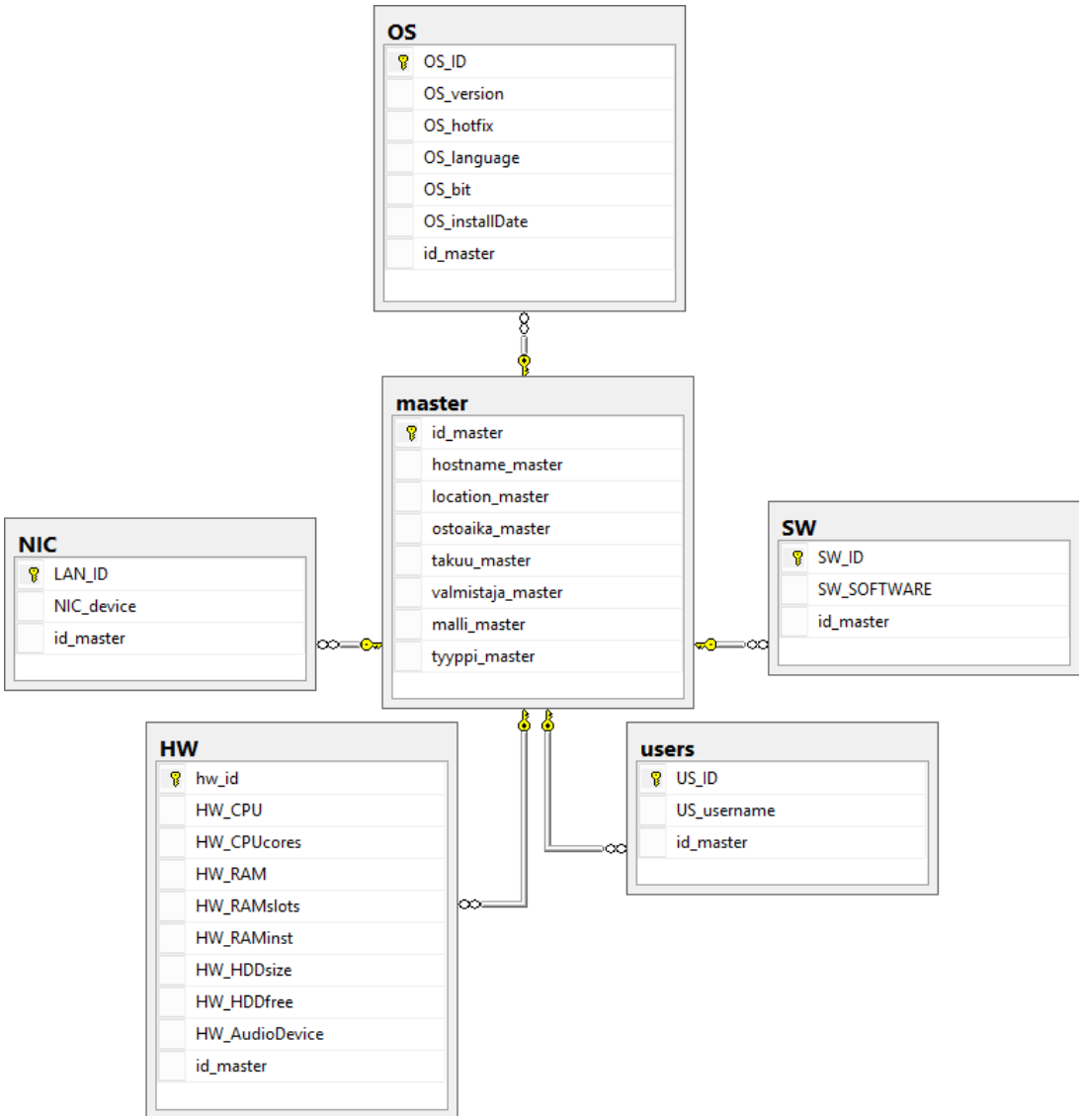
Tarvittavien kriteerien suunnittelun jälkeen alkoi miettiminen sen suhteen, että hankitaanko järjestelmä ulkopuoliselta taholta, vai kehitetäänkö se itse. Ulkoiset järjestelmät tuntuivat olevan erittäin laajoja yrityksen vaatimuksiin. Vaatimuksena oli myös tietokantaan tallentamisen mahdollisuus, koska sieltä tiedot saadaan hyödynnettyä yrityksen omistamiin järjestelmiin.

Lyhyiden testauksien ja päätelmien jälkeen organisaation sisäisen osaamisen todettiin riittävän tarvittavan järjestelmän kehitykseen. Seuraavana vaiheena tarvekartoituksen tekemisen jälkeen oli tietokannan suunnittelun aloitus.

4.2 Tietokannan suunnittelu

ICT-omaisuustietokantaversioon luodaan kuusi taulua, käyttöjärjestelmät (OS), ohjelmistot (SW), komponentit (HW), tietokoneet (master), käyttäjät (users) ja verkkokortit (NIC). Taulut ovat tietokannan peruselementtejä, ja yksi taulu kuvastaa aina yhtä kohdetta. Taulut sisältävät tietueita ja kenttiä. Sarake on tietokannan pienin käsite, ja sarakkeet sisältävät tietokannassa sijaitsevan tiedon. Tietue muodostuu yhdestä tai useammasta sarakkeesta ja tietue edustaa aina taulussa tiettyä, uniikkia esiintymää. (Hernandez 2013, luku 1, kappale 3. Terminology.) Näihin tauluihin tallennetaan tietokoneiden tunnistenumerot, käyttäjät, komponentit, valmistajat, mallit sekä asennettujen ohjelmistojen tiedot. Kannan laajentaminen aloitetaan pikkuhiljaa kun projekti on edennyt siihen vaiheeseen, että kaikki muut tekniset vaiheet on testattu toimivaksi.

Kuviossa 8 kuvataan taulujen väliset yhteydet relaatiokaaviolla. Jokaiselle taululle on määritelty perusavain. Perusavain identifioi yhden tietyn tietueen koko tietokannassa (Hernandez 2013, luku 1, kappale 3. Terminology). Kuviossa 8 taulujen perusavaimeksi valitun kentän tunnistaa sarakkeen edessä olevasta pienestä avaimesta. Kaikki taulut ovat yhteyksissä toisiinsa viiteavaimella nimeltä id_master. id_master kuvastaa tietokannassa laitteen emolevyn sarjanumeroa. Master taulua käytetään tietokannassa yksittäisen laitteiden identifiointiin, ja viiteavainten avulla taulussa sijaitseviin laitteisiin voidaan liittää siihen liittyviä ominaisuuksia.

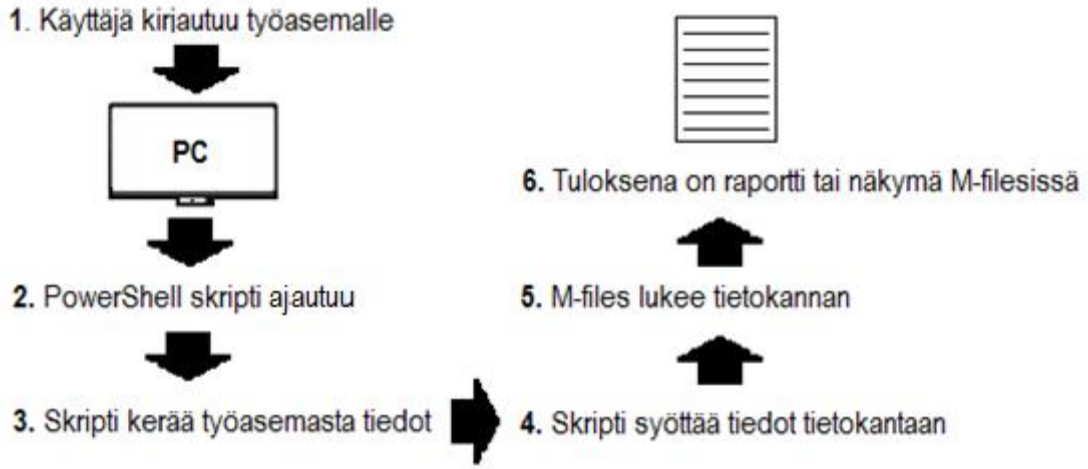


KUVIO 8. CMDB tietokannan relaatiokaavio.

4.3 Tietojen keräys

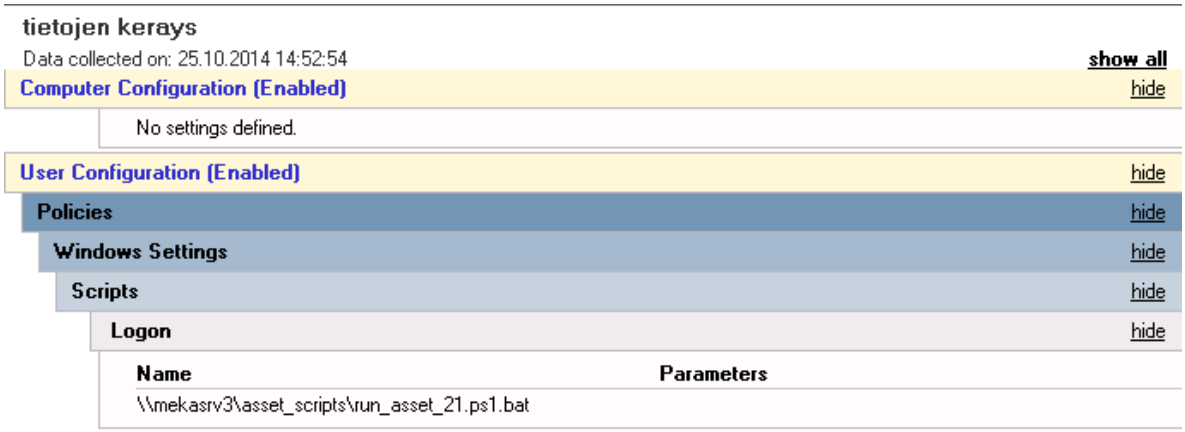
Omaisuuskanta sisältää tiedot tulostimista, näytöistä, puhelimista, navigaattoreista, videotykeistä, palvelimista, työasemista, projektoreista, tableteista, kameroista, palveluista ja ohjelmistolisenssistä. Laitteista kirjataan ylös laitteen sijainti, malli, valmistaja, sarjanumero, käyttäjä ja lisätiedot. Ohjelmistoista kerätään ohjelmiston nimi, valmistaja sekä lisenssien määrä, jos kyseessä on lisensoitu tuote. Kaikki fyysiset laitteet nimetään laitteen tyypin mukaan. Nimeämiseen käytetään tarratulostimella tehtyjä tarroja, ja esimerkiksi työasemissa tarra on helposti nähtävässä paikassa, jotta käyttäjä voi ongelmatilanteen sattuessa kertoa tietokoneensa nimen, ja tämän jälkeen voidaan tarkistaa käyttäjän laitteen tiedot omaisuuskannasta.

Tietojen kerääminen omaisuuskantaan aloitetaan työasemista ja asennetuista ohjelmistoista. Näiden tietojen kerääminen hoidetaan automatisoidusti. Tiedon keräys koostuu kolmesta isommasta kokonaisuudesta, PowerShell-skriptistä joka kerää automaattisesti tiedot laitteista ja ohjelmista, tietokannasta joka sijaitsee Microsoft SQL -palvelimella ja M-Files dokumenttienhallintajärjestelmään tehdyistä näkymistä ja hälytyksistä. Tämä prosessi on tarkemmin kuvattuna kuviossa 9.



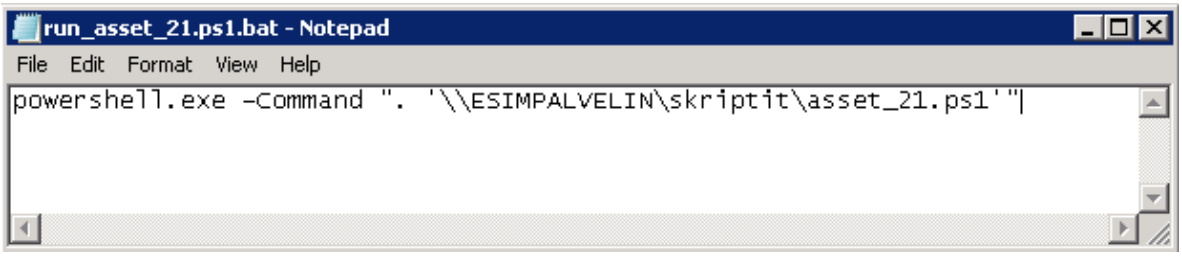
KUVIO 9. ICT-omaisuudenhallintajärjestelmän toimintaprosessi.

Työasemien ja ohjelmistojen tietojen keräys -skriptin ajo suoritetaan keskitetysti Group Policy -objektilla. Kuviossa 10. tarkempi kuvaus Group Policy -objektista. Skriptin ajoa varten on luotu myös Group Policy -objekti, jolla muutetaan PowerShell-ympäristön Execution Policy -asetus Remote Signed muotoon.



KUVIO 10. Tiedonkeräys-skriptin suoritus Group Policy -objektilla.

Kuten kuviosta 10 käy ilmi, valitussa Logon skriptissä ei ajeta suoraan PowerShell-skriptiä, vaan ensiksi ajetaan komentojonotiedosto, jossa kutsutaan PowerShell-komentotulkkia. Kuviossa 11 Batch-tiedoston tarkemmat parametrit. PowerShell Logon -skriptien suora tuki tulee Windows palvelimiin versiossa Windows Server 2008 R2 (Thomas a, hakupäivä 25.10.2014).



KUVIO 11. PowerShell-skriptin suoritus Batch-tiedoston avulla.

Skripti alkaa latautumaan käyttäjän kirjauduttua koneelle, ja käyttäjä saa ilmoituksen tietokoneen skannauksesta. Heti skriptin alussa tutkitaan omaisuuskannasta SQL SELECT -lausekkeella ovatko tiedot jo kannassa. Jos tietokoneen sarjanumero löytyy jo, niin suoritetaan vain pieni osa skriptistä, ja päivitetään tai lisätään muutamia asioita. Jos sarjanumeroa ei löytynyt tietokannasta, niin suoritetaan kaikki komennot, ja suoritetaan lopuksi SQL INSERT -lausekkeet. PowerShell-

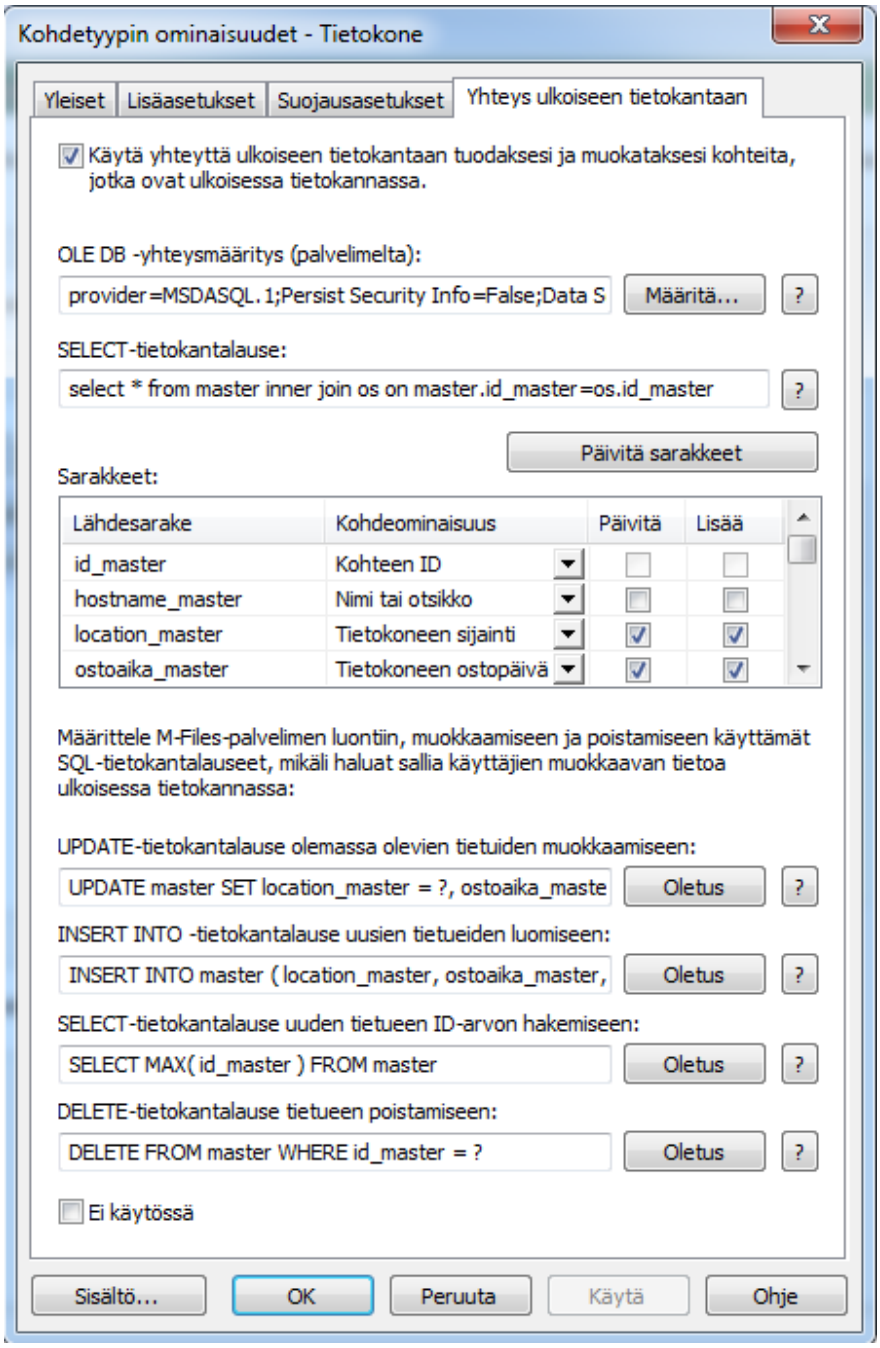
skriptin tarkka lähdekoodi kommentteineen löytyy liitteestä 1. Lähdekoodista on tietoturvasyistä jouduttu poistamaan muutamia rivejä.

4.4 M-Files toiminnollisuudet

Vaatimuksien mukaiset ominaisuudet oli helppo luoda M-Filesiin. Nämä asiat tulivat ilmi jo tarvekartoituksen yhteydessä. M-Files oli myös siksi luonnollisen oloinen ratkaisu, sillä tietokoneisiin ja lisensseihin liittyvät dokumentit löytyvät jo organisaation M-Files dokumenttivarastosta.

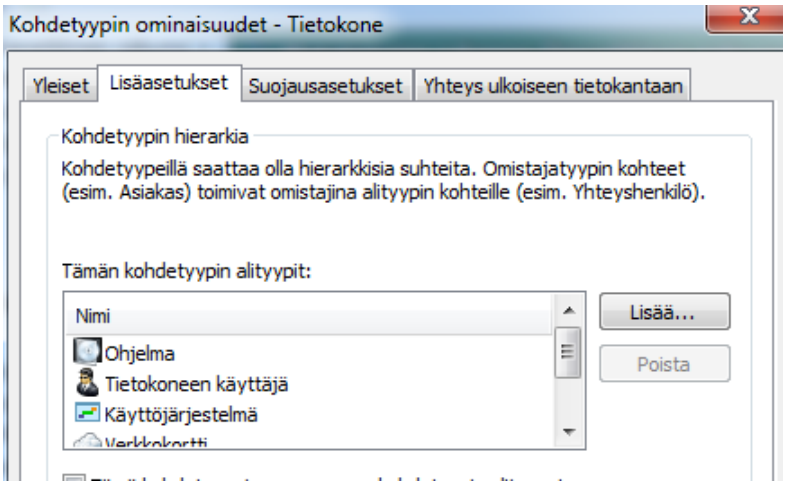
M-Filesissä on mahdollista yhdistää erilaisilla tekniikalla toimivat tietokannat, joten pystyttiin yhdistämään Microsoft SQL -tietokannalla olevat omaisuushallintajärjestelmän tiedot sekä M-Filesin Firebird SQL -tietokannalla sijaitsevat tietokone- ja lisensointidokumentit. Yhteys omaisuushallintajärjestelmän tietokantaan on luotu Microsoft OLE DB -yhteysmäärityksellä.

Kuviossa 12 esimerkki *tietokone* -kohdetyypistä, jossa tiedot haetaan M-Filesiin omaisuushallintajärjestelmän tietokannasta OLE DB -yhteysmäärityksellä. Kuviosta 12 selviää myös että kohdetyyppiin tulevat sarakkeet haetaan tavanomaisella SQL SELECT -lausekkeella. Myös päivitykset, poistot ja muokkaukset suoritetaan tavallisilla tietokantalausekkeilla. Tietokannasta haetuille sarakkeille tulee määrittää M-Filesissä näkyvät kohdetyypit. Esimerkiksi *location_master* sarakkeelle on määritetty kohdeominaisuudeksi *Tietokoneen sijainti*.



KUVIO 12. Tietokonekohdetyypin määritykset ulkoiseen tietokantayhteyteen.

Jokaiselle omaisuushallintakannassa sijaitsevalle taululle on luotu M-Filesin metatietorakenteeseen omat kohdetyypit. Taulut OS, SW, HW, users ja NIC ovat liitettynä *Tietokone* -kohdetyypin alityypeiksi. Kuviossa 13 selviää että tauluille on annettu selkeäkielisemmät nimet, ja miten liitokset ovat tehty.

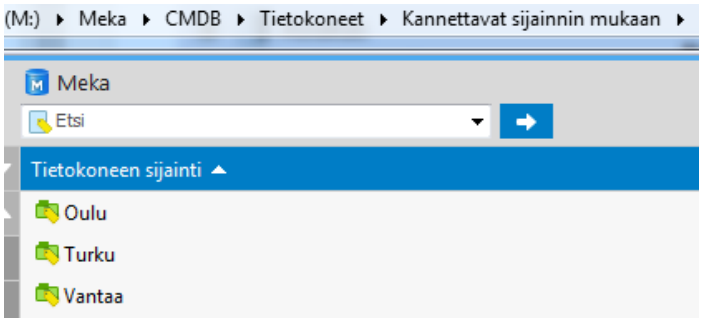


KUVIO 13. Tietokone-kohdetyypin alityypit.

Kuviossa 14 on esillä yksi esimerkki tietokonekohdetyypistä haetusta työasema-objektista. Työaseman alinäkymät ovat esille aukaistuna, ja sille määritetyt alityypit näkyvät listattuna sen alla. Alityypit saavat yhteyden tietokone-kohdetyypissä sijaitseviin objekteihin id_master-viiteavaimella. Kuviossa 14 on kuitenkin yksi poikkeus tähän, nimittäin erilaisia *Dokumentit* -luokkia oli jo olemassa M-Files yrityksen tietovarastossa ennen omaisuushallintakannan kehitystä. Näille tietyille dokumenttiluokille on nyt lisätty sarake *Tietokone*, eli jos jokin tietty dokumentti liittyy tietokoneeseen, voidaan sarakkeeseen valita haluttu tietokone.

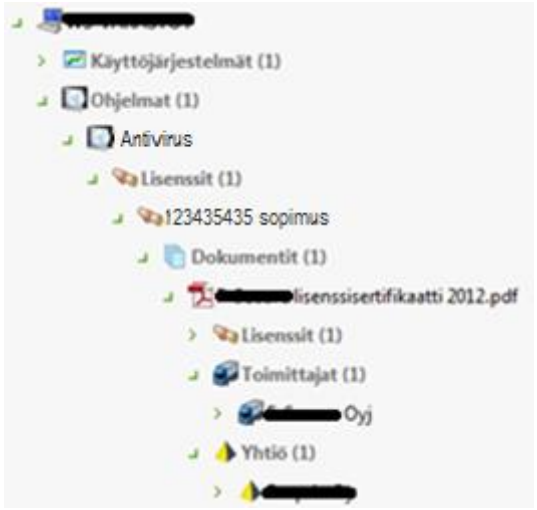
KUVIO 14. Esimerkki tietokoneobjektista M-Filesissä.

Tietokoneita varten on luotu monenlaisia näkymiä M-Filesiin: Laitteet sijainnin mukaan, laitteet takuun päättymisajan mukaan ja laitteet ostopäivämäärän mukaan. Kuviossa 15 on esillä näkymä *Kannettavat tietokoneen sijainnin mukaan.* Näkymään on määritelty seuraavat parametrit: Kohdetyyppinä tietokone, tietokoneen tyyppinä kannettava ja järjestys tietokoneen sijainnin mukaan.



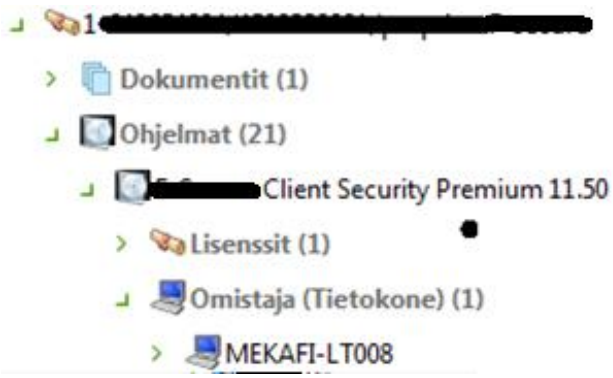
KUVIO 15. Kannettavat tietokoneen sijainnin mukaan -näkymä *M-Filesissä.*

Lisenssien hallinta M-Filesiin on luotu lisäämällä *Ohjelmat*-kohdetyyppiin sarake *lisenssi*. Lisenssi sarakkeeseen haetaan ohjelmistoon kuuluva lisensointisopimus. Lisensointisopimukset ja avaimet syötetään erikseen M-Filesiin. Kuvio 16 selventää asioiden yhteyden M-Filesissä. Kuviosta huomaa asioilla olevat lukuisat yhteydet. Näiden kaikkien hahmottaminen käsin on erittäin vaikeaa. Rakenne lähtee siten, että ylimpänä on tietokone, sitten näkyy ohjelmisto joka on asennettu tietokoneelle, tällä ohjelmistolla on sitten lisenssi, lisenssillä taas on sitten dokumentit ja toimittaja.



KUVIO 16. Lisenssien liittäminen asennettuihin ohjelmistoihin.

Saman asian yhteydet voidaan hahmottaa M-Filesissä myös toisinpäin. Jos halutaan esimerkiksi selvittää, että mitkä tietokoneet ovat jonkin vanhenevan lisensointisopimuksen alaisia, niin siirrytään Lisensointi -näkymään, joka on luotu M-Filesiin manuaalisesti (katso kuvio 17).



KUVIO 17. Lisenssinäkymä M-Filesissä.

4.5 Testaus ja käyttöönotto

Järjestelmän testaus suoritettiin siirtymällä pikkuhiljaa isompaan ympäristöön. Tietojen keräys -skriptiä testattiin kehitysvaiheessa yhdellä virtuaalisella työasemalla sekä yhdellä testikäytössä olevalla työasemalla. Se suoritettiin toiminnan testauksen jälkeen Group Policy -objektin kautta muutamalla työasemalla. Tästä ilmoitettiin käyttäjille etukäteen, ja käskettiin ilmoittaa jos koneen kirjautumisessa ilmenee hitautta tai muita ongelmia. Testit skriptin osalta menivät ongelmitta.

M-Filesiin tehty ympäristökin rakennettiin ensin erilliseen tietovarastoon. Tarvittavien testien jälkeen ympäristö rakennettiin tuotantoympäristöön. M-Files ympäristön rakentaminen sujui kohtuullisen vaivatta. Muutamiin teknisiin ongelmiin liittyen kysyttiin apua M-Filesin teknisiltä konsulteilta. Myös yrityksen järjestelmäasiantuntija oli useissa M-Filesiin liittyvissä toteutuksissa mukana.

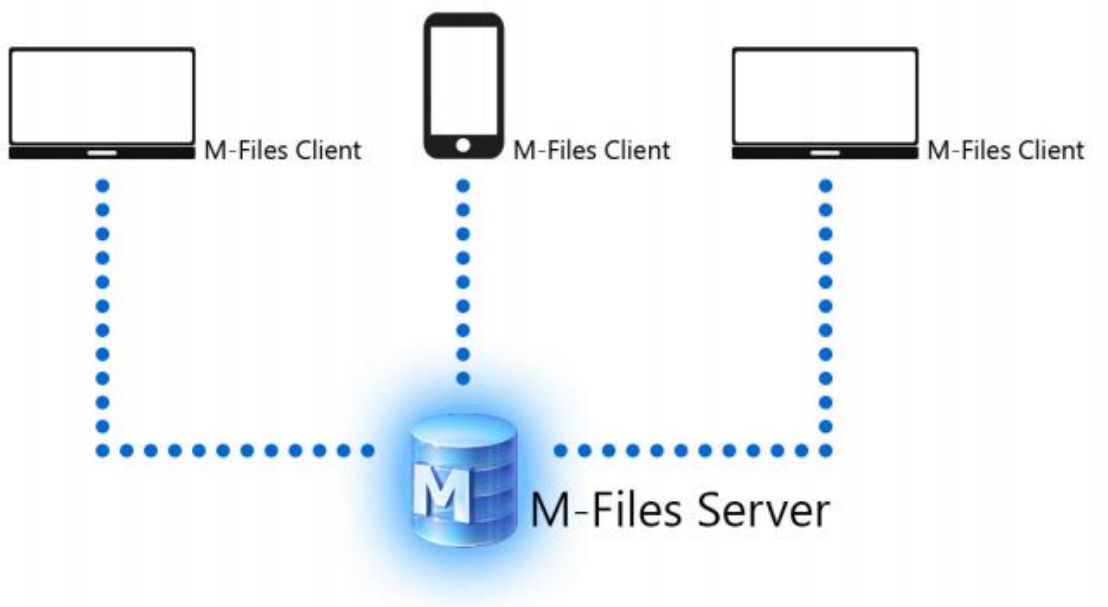
Ennen kuin automatisoitu tiedonkeruu otettiin koko organisaatiotasolla käyttöön, halutiin olla varmoja tietokannan sarakkeista ja yhteyksistä. Pohdittiin uusia kehitysmahdollisuuksia ja mahdollisia ongelmatilanteita. Tässä vaiheessa projektia löytyi vielä muutamia lisensseihin liittyviä asioita ja tehtiin vielä niihin liittyvät korjaukset.

Viimeinkin kun järjestelmä oli todettu helposti päivitettäväksi ja muokattavaksi, otettiin laitteiden ja ohjelmistojen tiedonkeruu -skripti massakäyttöön. Käyttöönotossa ei ilmentynyt ongelmia, ja skripti suoriutui kaikilla työasemille. Käyttäjätkään eivät huomanneet haittoja liittyen skannaukseen.

5 M-FILES

M-Files on dokumenttienhallintaohjelmisto, joka auttaa järjestelemään tiedot tehokkaammin kuin esimerkiksi Windowsin tavanomainen resurssienhallinta. Tiedon järjestely perustuu tiedon käyttöyhteyksiin sekä sisältöön, ei tallennussijaintiin. Kaikki tieto on helposti löydettävissä laajojen hakutoimintojen ja dynaamisten näkymien kautta. (M-Files a, 9.)

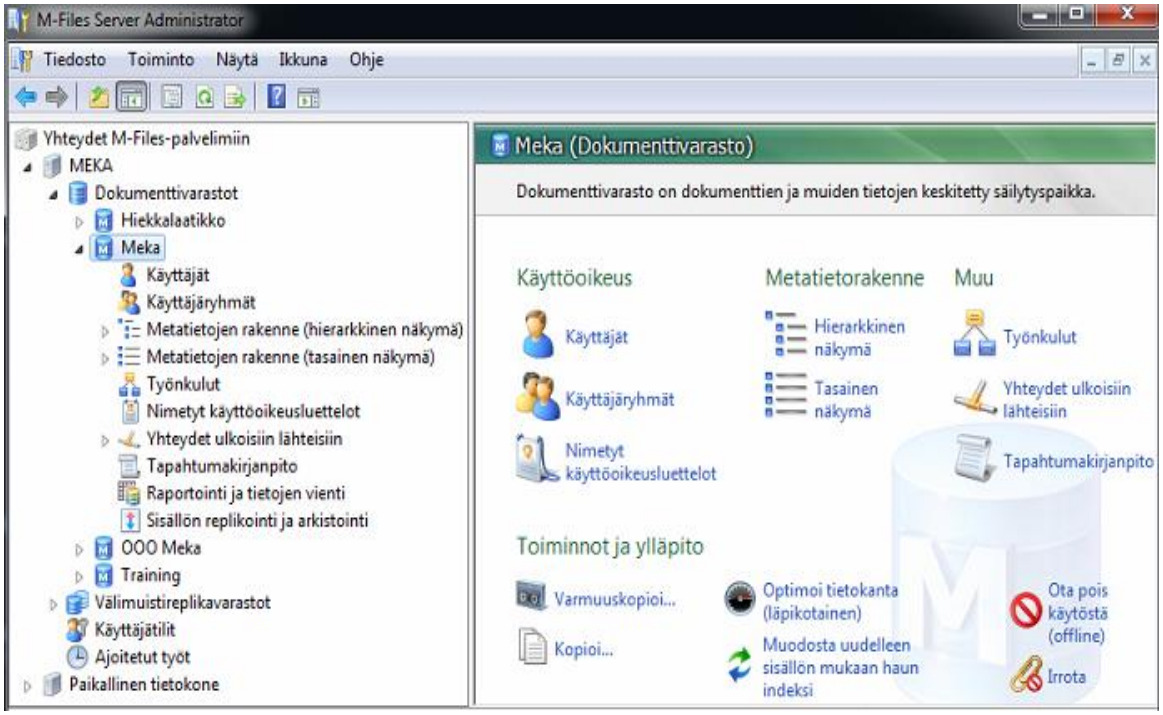
M-filesin dokumenttienhallintaohjelmisto rakentuu seuraavista ohjelmisto-osista: M-Files Asiakasohjelmisto, M-files-palvelimen hallinta, M-Files-asiakasohjelmiston asetukset, Näytä tilatietoja ja M-files palvelin. Kuviossa 18 näkyy perinteinen M-Files käyttöönottotilanne organisaatiossa. (M-Files a, 33.) Yleensä M-files-tietokantapalvelut toimivat M-Files-palvelimella suoraan, mutta isommissa tai raskaammissa ympäristöissä M-Files vaatii ulkopuolisen Microsoft SQL -tietokantapalvelimen.



KUVIO 18. M-Files-ohjelmiston käyttöönotto (M-Files a, 30).

5.1 M-Files-palvelinohjelmisto

M-Files-palvelinohjelmisto asennetaan erilliselle palvelinkoneelle. Dokumenttivarasto sijaitsee palvelinkoneella. Palvelimelle tallennetaan kaikki kohteet kuten, asiakkaat, työntekijät sekä dokumentit. Palvelimella hallitaan myös kaikkia käyttäjäoikeuksia. Asiakaskoneilta otetaan yhteys kyseiselle palvelinkoneelle, ja asiakaskoneet hakevat dokumenttivaraston sisällön, muutokset, oikeudet ja versiohistoriat palvelimelta. (M-Files a, 160.) Kuviossa 19 kuvankaappaus M-Files-palvelinohjelmistosta.



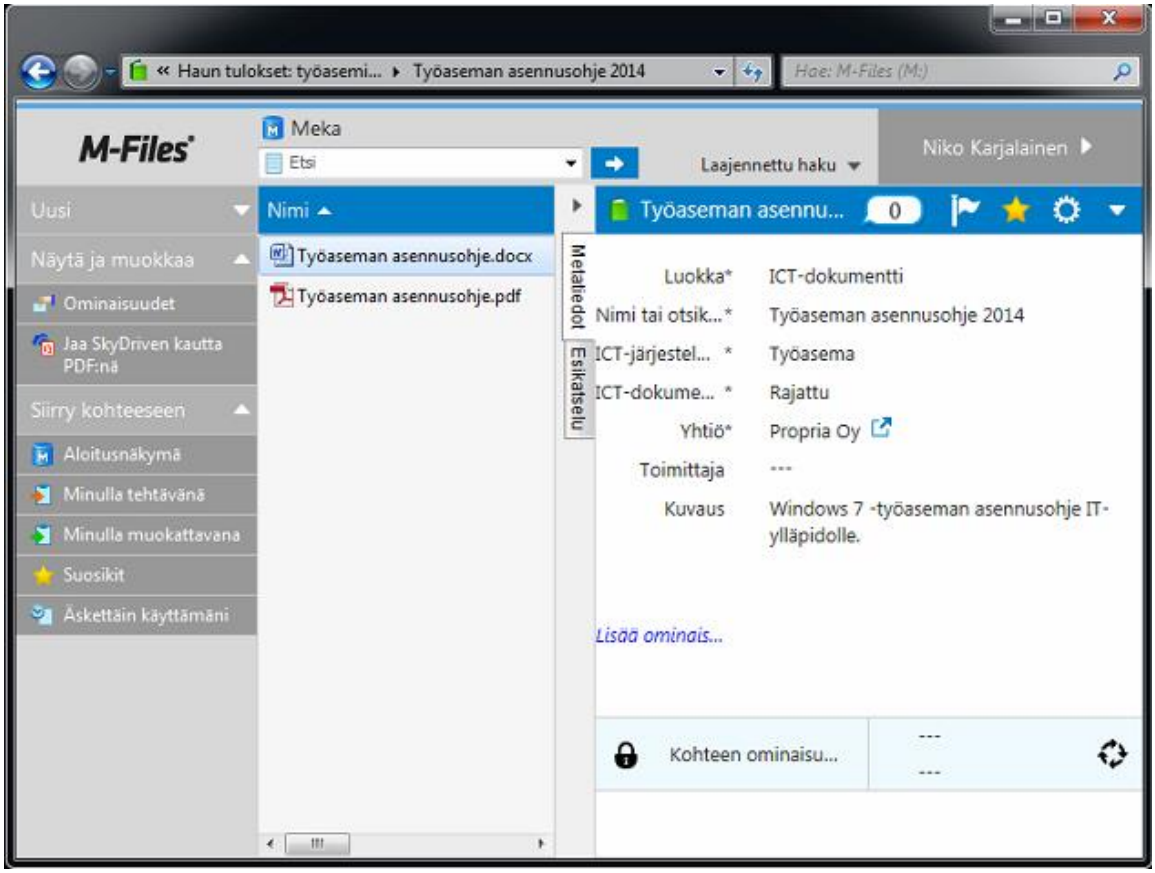
KUVIO 19. M-Files-palvelinohjelmiston aloitusnäkymä

M-Files-palvelinohjelmistoa voidaan ajaa myös täysin pilvipohjaisena ratkaisuna, tai pilvihybridinä. Pilviratkaisuissa M-Files-palvelimen dokumenttivarasto tai koko palvelinohjelmisto voidaan sijoittaa Windows Azure -pilvialustalle. M-Files-palvelimen dokumenttivarasto voidaan integroida helposti olemassa oleviin asiakkuudenhallinta- tai toiminnanohjausjärjestelmiin. (M-Files b, hakupäivä 26.10.2014.)

M-Files-dokumenttivarastot tukevat tietokantamoottorina joko Firebird SQL -palvelinta tai Microsoft SQL -palvelinta. Firebird on M-Filesiin sulautettu SQL-tietokantamoottori, ja sitä suositellaan käyttämään pienissä tai keskikokoisissa ympäristöissä. Jos Firebird-tietokantamoottorin suorituskyky alkaa loppumaan käytössä olevassa ympäristössä, on siirtyminen Microsoftin SQL -palvelimeen helppoa. (M-Files a, 182.)

5.2 M-Files-asiakasohjelmisto

M-Files-asiakasohjelmiston virallinen kutsumanimi on *Selaa M-Filesia*. Asiakasohjelmiston käyttöliittymä koostuu neljästä osa-alueesta: listausnäkymä, oikea sivupaneeli, hakutoiminnot sekä tehtäväalue (katso kuvio 20). (M-Files a, 32.)



KUVIO 20. Kuvankaappaus M-Files-asiakasohjelmistosta.

Listausnäkymässä näkyvät tietylle käyttäjälle sallitut tiedostot ja näkymät. Näkymä toimii samoin kuten Windowsin Resurssienhallinta (M-Files a, 32). Esimerkkinä kuvio 20, jossa listausnäkymässä näkyy tiedostot nimeltä *Työaseman asennusohje.docx* ja *Työasemanasennusohje.pdf*.

Käyttöliittymän vasemmalla reunalla on tehtäväalue. Oletuksena tehtäväalueelta löytyvät valinnat uusien kohteiden luontiin (Uusi) ja usein käytettyihin näkymiin siirtymiseen (Siirry kohteeseen) (M-Files a, 32). Kuviossa 20 näkyy esimerkkinä, *Suosikit*, *Äskettäin käyttämäni* sekä *aloitusnäkymä*.

Hakunäkymä on käyttöliittymän ylälaidassa. Oletuksena esillä on vain pikahaku, ja laajennettu haku on painikkeen *Laajennettu haku* takana. Pikahaku toimii suurimmissa osissa hauista, ja laajennettua hakua käytetään silloin, jos hakukriteerejä pitää olla useampia määriteltynä. (M-Files a, 116–117.)

Oikealla sivupaneelilla on monta eri funktiota. Kun mitään tiedostoa tai näkymää ei ole valittuna, niin sivupaneelista voi luoda uuden kohteen M-Files-tietovarastoon, tai valita jonkin tutustu M-Filesiin -osion linkeistä (katso kuvio 21). (M-Files a, 32.)

KUVIO 21. M-Files-asiakasohjelmiston oikea sivupaneeli

Kun jokin kohde on valittuna, niin oikea sivupaneeli näyttää joko metatietokortin tai esikatselun, riippuen valitusta kohteesta (M-Files a, 32). Kuviossa 20 oikealla sivupaneelilla on näkyvissä tiedoston *Työaseman asennusohje.docx* metatietokortti. Nämä metatiedot ovat dokumenttiin kuuluvaa ominaisuustietoa, jonka avulla dokumentteja voidaan muun muassa organisoida ja hakea (M-Files a, 33).

Käyttöliittymän oikeassa yläkulmassa näkyy tämän hetkisen kirjautuneen käyttäjän nimi. Nimestä napsauttamalla käyttäjä saa henkilökohtaisen valikon auki, jonka alta pystytään muun muassa muuttamaan ilmoituksiin, kieliin ja sijaiskäyttäjiin liittyviä asetuksia. Valikon alta vois myös kirjautua ulos, tai valita käyttävänsä M-Filesiä offline-tilassa. (M-Files a, 33.)

6 POWERSHELL

Windows PowerShell on laajentuva komentotulkki ja skriptausympäristö, jolla pystytään hallitsemaan Windowsin palvelinkäyttöjärjestelmiä, Exchangea, Sharepointtia ja työasemia. Laajentuvalla komentotulkilla tarkoitetaan mahdollisuutta lisätä omia työkaluja PowerShell-ympäristöön. PowerShell perustuu C#-ohjelmointikieleen ja .NET Frameworkiin. (Katerberg, hakupäivä 19.10.2014.) PowerShell kehitettiin alun perin vuonna 2002 koodinimellä *Monad.* Vuonna 2006 Microsoft julkisti alustan ensimmäisen version nimellä Windows PowerShell. PowerShell tulee esiasennettuna kaikkiin Windows työasemiin ja -palvelimiin, joten niiden hallinta onnistuu helposti PowerShell-skripteillä. (Finke 2012, 1–2.)

Alkuperäisessä toiminnollisuudessaan PowerShellin versio v3 tarjoaa 2300 komentoa. (Finke 2012, 2.) Näitä komentoja yhdistämällä voidaan luoda skriptitiedosto yhden helposti muistettavissa olevan komennon taakse. Toistuvien tehtävien tekeminen on varmempaa, koska kun toistuvia tehtäviä ajetaan komentotulkilta yksinkertaisella skriptillä, niin mahdolliset ihmisen aiheuttamat virheet vähenevät. (Katerberg, hakupäivä 19.10.2014.)

6.1 Tietoturva

PowerShell käyttää skriptien suorittamiseen Execution Policy -asetusta. Execution Policy -asetus päättää, saako PowerShell ladata konfiguraatiotiedostoja käyttöönsä, ja mitä skriptejä komentotulkilla voidaan suorittaa. Oletuksena PowerShellillä ei voida suorittaa mitään skriptejä. PowerShellissä on olemassa kuusi erilaista Execution Policy -sääntöä, nämä ovat Restricted, AllSigned, RemoteSigned, Unrestricted, Bypass ja Undefined. Taulukossa 1 kuvataan näiden asetusten eroavaisuudet. (Finke 2012, 7–9.)

TAULUKKO 1. Execution Policy -asetusten eroavaisuudet (Finke 2012, 8–9).

Asetus	Kuvaus
Restricted	Ei ladata mitään konfiguraatiotiedostoja tai skriptejä. Restricted on PowerShellin oletusasetus.
AllSigned	Ladataan ainoastaan sellaiset skriptit ja konfiguraatiotiedostot jotka ovat allekirjoitettuja luotettavan julkaisijan toimesta, mukaan lukien omalla koneella kirjoitetut skriptit.
RemoteSigned tiotie-	Ladataan ainoastaan sellaiset Internetistä ladatut skriptit ja konfiguraa- dostot, jotka ovat allekirjoitettuja luotettavan julkaisijan toimesta. Paikalliset skriptit ovat sallittuja.
Unrestricted	Ladataan kaikki konfiguraatiotiedostot ja skriptit. Jos suoritetaan Internetistä ladattu allekirjoittamaton skripti, niin PowerShell kysyy käyttäjältä luvan suorittamiseen.
Bypass tä.	Ladataan kaikki konfiguraatiotiedostot ja skriptit ilman minkäänlaisia estei-
Undefined	Poistaa tällä hetkellä voimassa olevan Execution Policy -asetuksen. Tämä asetus ei kuitenkaan muuta Group Policyyn määritettyä asetusta.

Execution Policy -asetuksista Restricted on turvallisin, mutta se ei sovi millään tavalla kehittäjille. Restricted-asetuksen ollessa päällä komentokehotteelta voidaan suorittaa ainoastaan PowerShellin omia komentoja, eikä räätälöintiä voida harrastaa ollenkaan. Yleisesti sanottuna RemoteSigned on hyvä ratkaisu aloitteleville kehittäjille, koska ei voida suorittaa Internetistä ladattuja tuntemattomia skriptejä, ja näin ollen voidaan luottaa että ei saada tuhoa aikaan. (Finke 2012, 8.)

6.2 Skriptaaminen PowerShell-ympäristössä

PowerShell asennuksen mukana tulee oletuksena ohjelma nimeltä PowerShell ISE (Integrated Scripting Enviroment). ISE on graafinen ohjelma joka on luotu PowerShell skriptien kirjoitusta varten. ISE:llä voidaan suorittaa, kirjoittaa, muokata, testata ja debugata skriptejä ympäristössä. ISE näyttää koodin syntaksit väreissä ja tukee Unicode-merkistöä. (Finke 2012, 9–10.)

Putkitus

Putkituksessa liitetään kaksi lauseketta yhteen siten, että ensimmäisen lausekkeen tulosteesta tulee toisen lausekkeen syöte (Thomas b, hakupäivä 19.10.2014). Kuvion 23 esimerkissä on haettu tietokoneen suorittimen tiedot käyttämällä *Get-Wmiobject* -komentoa. Ensimmäisessä komennossa ei ole putkitusta käytetty, ja tulostuksesta huomaa että tietoa tuli enemmän. Toisessa komennossa on käytetty *select* objektia putkituksen yhteydessä, ja valittua luokasta win32_processor pelkästään *name* attribuutti.

```
PS C:\Users\niko> get-wmiobject win32_processor


Caption            : Intel64 Family 6 Model 42 Stepping 7
DeviceID           : CPU0
Manufacturer       : GenuineIntel
MaxClockSpeed      : 3300
Name               : Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz
SocketDesignation  : LGA1155



PS C:\Users\niko> get-wmiobject win32_processor | select name

name
----
Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz
```

KUVIO 23. Putkituksen käyttö PowerShellissä.

Putkitus muokkaa ensimmäisen lausekkeen tulostetta. Putkituksella voidaan järjestellä, muokata ja hakea tiettyjä arvoja tulosteesta. Putkituksella saadaan järjestettyä ja siistittyä ohjelmakoodia. Putkitus vähentää myös työskentelyn vaivaa ja määrää. (Finke 2012, 15–16.)

Ohjelmointi PowerShell-skripteissä

PowerShell on dynaaminen skriptauskieli, joten se tukee loogisia lausekkeita ja silmukoita tavalliseen tapaan. Skriptejä pystytään ajamaan perustuen ehtoihin jotka esiintyvät tai käyvät toteen. Skripti voidaan esimerkiksi ajastaa suoriutumaan, kun jokin tietty prosessi tai ohjelmisto sulkeutuu. Logiikalla saadaan tehtyä päätöksiin perustuvia älykkäitä skriptejä. (Hamrick a, hakupäivä 19.10.2014.)

Silmukat ovat myös tarvittava lisä skripteihin. Niiden avulla saadaan tehtyä toistuvia toimenpiteitä. Yksi esimerkki PowerShell-skripteissä käytetyistä silmukoista *For Each* -lauseke jossa skripti suoritetaan jokaiselle taulussa tai kokoelmassa sijaitsevalle objektille. Käytännössä siis suoritetaan esimerkiksi tietyt toimenpiteet jokaiselle tietokoneelle, jotka sijaitsevat tietyssä kokoelmassa. (Hamrick b, hakupäivä 19.10.2014.)

Windows Management Instrumentation

Windows Management Instrumentation (WMI) on kokoelma objekteista, jotka mahdollistavat pääsyn käyttöjärjestelmän tietoihin. Jokaisella näistä objekteista on omat ominaisuudet, menetelmät ja tapahtumat. Kaikki näistä objekteista on määritetty *Manage Object Format* -tiedostoissa, jotka sijaitsevat käyttöjärjestelmän tiedostoissa. Nämä tiedostot latautuvat *Provider* isännillä, ja kun *Provider* isäntä on rekisteröity, se lataa objektien määritteet tämän hetkiseen WMI-nimiavaruuteen. Nämä nimiavaruudet näkyvät tiedostorakenteena, jonka alla löytyvät kyseiseen nimiavaruuteen kuuluvat luokat. Käyttöjärjestelmä ja siihen asennetut ohjelmistot jakavat tiedot näihin luokkiin. (Perez, hakupäivä 19.10.2014.) Kuviossa 24 on haettu Windows 8.1 käyttöjärjestelmällä käytössä olevat nimiavaruudet PowerShellin avulla.

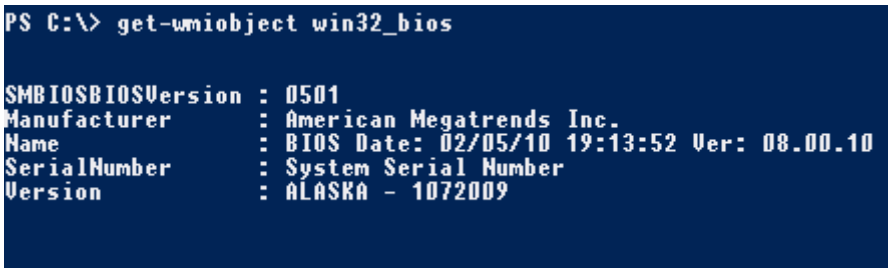
```
PS C:\> gwmi -namespace "root" -class "__Namespace" | Select Name

Name
----
subscription
DEFAULT
CIMV2
msdtc
Cli
nap
SECURITY
SecurityCenter2
RSOP
StandardCimv2
WMI
directory
Policy
virtualization
Interop
Hardware
ServiceModel
SecurityCenter
MSAPPS12
Microsoft
```

KUVIO 24. Windows 8.1 käyttöjärjestelmän nimiavaruudet listattuna PowerShellissä.

PowerShellissä tiedon haku nimiavaruuksista onnistuu komennolla *Get-WmiObject.* Komentoa voidaan ajaa myös etätietokoneille, joka tekee PowerShellistä todella vahvan hallintatyökalun.

Oletuksena *Get-WmiObject*-komento käyttää *root\cimv2*-nimiavaruutta, joka tekee minkä tahansa luokan tiedon hakemisesta erittäin helppoa. Esimerkkinä kuvio 25, jossa haetaan tietoa tietokoneen biosista. (Microsoft TechNet, hakupäivä 19.10.2014.)



KUVIO 25. Tietokoneen BIOS-tietojen haku PowerShellin Get-WmiObject -komennolla.

7 POHDINTA

Työn tavoitteena oli suunnitella ICT-omaisuudenhallintajärjestelmä Meka Pro Oy:lle. Tein suuren osan järjestelmän suunnittelusta teoriaosuuden kirjoituksen jälkeen. Tämän totesin oikeaksi ratkaisuksi, sillä sain lukemastani kirjallisuudesta paljon uusia näkemyksiä työasemien ja ohjelmistolisenssien hallinnasta. Opiskelun myötä osasin myös rajata käytännön työn sopivaksi kokonaisuudeksi, enkä pyrkinyt tekemään kerralla kaikkea. Tämän myötä järjestelmästä rakentui hyvin skaalautuva, ja jatkokehityssuunnitelmat ovat jo mielessä. Esimerkiksi jonkinlaista integraatiota yrityksen ERP-järjestelmän kanssa on tiedossa.

Työn aihe sai alkunsa jo aiemmin ammattiharjoittelussa kesällä 2014 Meka Pro Oy:n järjestelmäasiantuntijan kanssa käydyistä IT-palavereista. Palavereissa totesimme uuden ICT-omaisuudenhallintajärjestelmän tarpeelliseksi, ja emme löytäneet sopivaa järjestelmää ulkopuolisilta tarjoajilta. Totesimme oman osaaminen ja mielenkiinnon riittävän järjestelmän toteutukseen, ja päätimme suunnitella järjestelmän itse.

Työn toteuttaminen meni aikataulutuksen puolesta jopa paremmin kuin alkuperäisissä suunnitelmissa. Käytännön työn osuudessa luodun järjestelmän käyttöönotto meni erinomaisesti, ja opinnäytetyön tilaaja oli työhön tyytyväinen. Olin myös itse tyytyväinen työn tuloksiin ja järjestelmä on ollut jo jokapäiväisessä työssä apuna.

Opinnäytetyön tekeminen lisäsi huomattavasti ymmärrystäni IT-ympäristöjen yleisistä hallintaprosesseista ja käytännöistä. Opin myös paljon uutta skriptauksen mahdollistamista automatisoiduista tehtävistä, sekä tietokannan suunnittelusta. Suunnittelun ja asioihin perehtymisen tärkeys korostui itselleni tämän työn myötä.

Lopuksi haluaisin kiittää kaikkia opinnäytetyön tekemisessä auttaneita tahoja. Erityiset kiitokset Marko Matelalle, joka ehti omien työkiireidensä keskeltä auttamaan opinnäytetyön käytännön osuuden kanssa.

LÄHTEET

Braden, S. 2005. IT Asset Management: A Pocket Survival Guide. Cambridgeshire, Iso-Britannia: IT Governance Publishing.

Finke, D. 2012. Windows Powershell for Developers. Sebastopol, Kalifornia: O'Reilly Media, Inc.

Hamrick, J. a Conditional Logic. Hakupäivä 19.10.2014, http://www.powershellpro.com/powershell-tutorial-introduction/powershell-tutorial-conditional-logic/

Hamrick, J. b Conditional Logic Using Loops. Hakupäivä 19.10.2014, http://www.powershellpro.com/powershell-tutorial-introduction/powershell-tutorial-conditional-logic/

Hartvaara, M. 2008. Miten ja miksi prosesseja mallinnetaan?. Hakupäivä 8.11.2014, http://www.lpt.fi/tykes/news_doc/prosessit_mea-hartvaara.pdf

Hernandez M, J. 2013. Database Design for Mere Mortals®: A Hands-on Guide to Relational Database Design. Ann Arbor, Michigan: Addison-Wesley Professional. Kolmas julkaisu.

Hobbs, M. 2011. The ITIL Foundation Exam Study Guide. Richardson, Texas. Hakupäivä 20.9.2014, http://www.pc-freak.net/files/ITIL_v3_books/ITIL%20Foundations%20EN.pdf.

Hyvönen, T. Missä järjestyksessä CMDB rakentuu parhaiten. Hakupäivä 25.9.2014, http://www.itil.fi/2009/02/missa-jarjestyksessa-cmdb-rakentuu.html

IT Process Maps. 2009. Introduction: ITIL Version 3 and the ITIL Process Map V3. Hakupäivä 8.11.2014, http://ftp.psu.ac.th/pub/itil/introduction_itil_process_map_v3.pdf

Katerberg, R. 2012. What is PowerShell and why should I use it?. Hakupäivä 19.10.2014, http://www.sharepointnoob.com/2012/07/11/what-is-powershell/

Knowledge Transfer. Process, Hakupäivä 8.11.2014, http://www.knowledgetransfer.net/dictionary/ITIL/en/Process.htm

Mackie, K. 2013. Microsoft Releases Windows 8.1, Windows Server 2012 R2 and System Center 2012 R2. Hakupäivä 18.5.2014, http://redmondmag.com/articles/2013/10/18/windows-8.1-and-wave-r2-release.aspx.

Meka Pro Oy. Historia. Hakupäivä 14.10.2014, http://www.meka.eu/Meka-Pro-Oy/Historia.html.

ManageEngine. ITIL Features. Hakupäivä 8.11.2014, http://www.manageengine.com/products/service-desk/itil-features.html

McGunigle, K. 2014. ITIL History. Hakupäivä 20.9.2014, https://purplegriffon.com/blog/it-service-management/itil-history.

Microsoft TechNet. 2014. Using the Get-WMiObject Cmdlet. Hakupäivä 19.10.2014, http://technet.microsoft.com/en-us/library/ee176860.aspx

Mistry, R & Misner, S. 2014. Introducing Microsoft SQL Server 2014. Redmond, Washington: Microsoft Press

M-Files. Ohjekirja Käyttäjälle. Hakupäivä 5.10.2014, www.m-files.com/Content/documents/fi/res/M-FilesUsersGuide.pdf.

M-Files. Pilvi & on-premise. Hakupäivä 26.10.2014, http://www.m-files.com/fi/cloud-on-premise-hybrid

Perez, C. 2013. Introduction to WMI Basics with PowerShell Part 1 (What it is and exploring it with a GUI). Hakupäivä 19.10.2014. http://www.darkoperator.com/blog/2013/1/31/introduction-to-wmi-basics-with-powershell-part-1-what-it-is.html

QFIX. IT Laitteiden koko elinkaari hallitusti. Hakupäivä 8.11.2014, http://www.qfix.fi/palvelut/laitteet-elinkaaripalvelut/

Thejendra, B.S. 2014. Practical IT Service Management. Cambridgeshire, Iso-Britannia: IT Governance Publishing. Toinen julkaisu.

Thomas, G b. PowerShell Scripting - Pipeline Symbol. Hakupäivä 19.10.2014, http://www.computerperformance.co.uk/powershell/powershell_pipeline.htm

Thomas, G a. PowerShell Logon Scripts. Hakupäivä 25.10.2014 25.10.2014,http://www.computerperformance.co.uk/powershell/powershell_logon_script.htm

Thompson, M. ITAM: A Prerequisite for Building a CMDB (Part 3/6) – "4 Reasons Why ITAM is a Prerequisite for a CMDB". Hakupäivä 25.9.2014, http://www.itassetmanagement.net/2009/11/27/itam-a-prerequisite-for-building-a-cmdb-part-36-%E2%80%93-%E2%80%9C4-reasons-why-itam-is-a-prerequisite-for-a-cmdb%E2%80%9D/

Turbitt, K. What's New in ITIL V3? Understanding the changes in the latest ITIL release. BMC Software. Hakupäivä 14.10.2014, http://www.ts.avnet.com/clientsolutions/itilv3_whats_new

POWERSHELL TIEDONHAKU-SKRIPTIN LÄHDEKOODI LIITE 1

```
 [void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")

#Määritellään ilmoitusmuuttujat
$objNotify = New-Object System.Windows.Forms.NotifyIcon
$objNotify.Icon = [Drawing.Icon]::ExtractAssociatedIcon((Get-Command powershell).Path)

$CurrentUser=[Environment]::UserName
$ID = (Gwmi Win32_Bios).Serialnumber

  #### Määritellään arvot yhteyttä varten:
   $ConnectionString = “Server=MEKASRV;Database=CMDB;Integrated Security=true;”
   $Connection = New-Object System.Data.SqlClient.SqlConnection
   $Connection.ConnectionString = $ConnectionString

  #### Aukaistaan yhteys tietokantaan
   $Connection.Open()

  # User kanta tutkitaan erikseen, koska näin voidaan tutkia esim tuotannon ja varaston koneilta, että ketkä käyttäjät käyttävä konetta.
  # Ja jos jokin kone on hukassa, niin nähdään että kuka käyttäjä konetta käyttää tällä hetkellä.

  $queryIsUserAdded="SELECT * FROM users WHERE us_username='"+$CurrentUser+"' AND id_master='"+$ID+"'"
  $command = $connection.CreateCommand()
  $command.CommandText =$queryIsUserAdded
  $FinalUserSelect = $command.ExecuteReader()
  $queryUserCheck=new-object "System.Data.DataTable"
  $queryUserCheck.Load($FinalUserSelect)
  $queryUserCheckCount=($queryUserCheck | measure-object).count

  if($queryUserCheckCount -eq 0) { $USER_Query = “INSERT INTO users(US_username,id_master) VALUES('$CurrentUser','$ID')" }
  else { $USER_Query=$null <# tehdään tyhjä insert lause jos arvot olivat jo kannassa #> }

  $FinalUserSelect.close()

#Master kannan tutkinta

#### Tarkistetaan, että joko tietyt arvot ovat olemassa, duplikaatteja ei haluta
#### (Osa arvoista syötetään auto incremental primary keyllä, joten duplikaattiarvoja voisi syntyä)

$queryIsMaster="SELECT * FROM master WHERE id_master='"+$ID+"'"
$command = $connection.CreateCommand()
$command.CommandText = $queryIsMaster
$FinalMasterSelect = $command.ExecuteReader()
$queryMasterCheck=new-object "System.Data.DataTable"
```

```
$queryMasterCheck.Load($FinalMasterSelect)
$queryMasterCheckCount=($queryMasterCheck | measure-object).count

$FinalMasterSelect.close()

if($queryMasterCheckCount -eq 0) {

 # Tulostetaan käyttäjälle ilmoitus että järjestelmän skannausta suoritetaan

  $objNotify.BalloontipIcon = "Info "
  $objNotify.BalloonTipText = "Suoritetaan järjestelmän skannausta"
  $objNotify.BalloonTipTitle = "IT-tuki"

  $objNotify.Visible = $true
  $objNotify.ShowBalloonTip(15)

 ##### Variables to be inserted in CMDB #######

   # Serialnumber For the Master database
   #
    $ID = (Gwmi Win32_Bios).Serialnumber
   #

 #### Computer Model & manufacturer ##########

    $PC_Manufacturer = (Gwmi Win32_computersystem).manufacturer
    $PC_Model = (Gwmi Win32_computersystem).model

 ##### Operating system variables ##########
   #
    # Hostname will be saved in Master Database!
    $OS_Hostname = Hostname
    $OS_SP = (gwmi Win32_OperatingSystem).ServicePackMajorVersion
    $OS_Version = (gwmi Win32_OperatingSystem).Caption +"SP "+$OS_SP
    $OS_Hotfix = (Get-Hotfix | Measure-Object).count
    $OS_Language = (Get-Culture).Name
    $OS_Bit = if ([System.IntPtr]::Size -eq 4) { "32-bit" } Else { "64-bit" }
    $OS_InstallDate                =                ([WMI]").ConvertToDateTime((Get-WmiObject
Win32_OperatingSystem).InstallDate)
   #
 ##### Hardware variables #################
   #

    $hwCPU = (gwmi Win32_Processor).Name
    $hwCPUCores = (gwmi Win32_Processor).NumberOfCores

    # Ram määrän hakeminen on hieman monimutkaisempaa, luku tulee bitteinä, joten
    # se muutetaan gigatavuiksi. Tämän jälkeen ei tule vielä tasalukua, joten luku täytyy pyöristää.
    $hwRAM = ((Get-WmiObject -Class Win32_ComputerSystem).TotalPhysicalMemory)/1gb
```

```
#Ram luvun pyöristys:
$hwRAM = "{0:N0}" -f $hwRAM

 # Tutkitaan onko vapaita muistipaikkoja, ensin slot-määrä
 $hwRAMslot = (Get-WmiObject -Class "Win32_PhysicalMemoryarray" -namespace "root\cimv2").memorydevices

 # Montako kampaa asennettu

 $hwRAMs = Get-WmiObject -Class "win32_PhysicalMemory" -namespace "root\CIMV2"
 $hwRAMinst = ForEach ($tempRAM in $hwRAMs) { $tempRAM.DeviceLocator }
 $hwRAMinst = $hwRAMinst.Count

# haetaan verkkokortit, fyysiset verkkokortit haetaan valmistajan nimien mukaan.

 # 1. tallennetaan verkkokorttien tiedot väliaikaiseen tiedostoon $env:temp\templan.txt

 Get-WmiObject Win32_NetworkAdapterConfiguration | Select Description > $env:temp\templan.txt

 # 2. Karsitaan tekstitiedostosta ylimääräiset vaihtoehdot pois ja tallennetaan uuteen tekstitiedostoon.

 Select-String -Pattern intel,sierra,marvell,broadcom,asus -Path "$env:temp\templan.txt" `
 | FOREACH { $_.Line } > $env:temp\templan2.txt

 # 3.Tehdään laskuri tekstitiedoston verkkokorttien määrän mukaan

 $hwLANlaskuri=(Select-String -Pattern intel,sierra,marvell,broadcom,asus `
 -Path "$env:temp\templan.txt" | FOREACH { $_.Line }).Count

 # 4. kirjataan verkkokortit tauluun, taulun koko määräytyy verkkokorttien määrän mukaan
 # Tämä operaatio käydään for lausekkeella läpi, täytetään taulua niin pitkään kuin verkkokortteja on.

 $hwLANdevice = @()
 For($i=0; $i -le $hwLANlaskuri; $i++){ $hwLANdevice+= (Get-Content $env:temp\templan2.txt)[$i] }
 # hwLANlines taulusta syötetään verkkokortit omina riveinään sitten tietokantaan.
 # Syöttäminen tehdään for lausekkellee.

 # Poistetaan väliaikainen tiedosto johon verkkolaitteet tallennettiin
 Remove-Item $env:temp\templan.txt
 Remove-Item $env:temp\templan2.txt

# Kovalevyjen tutkinta. Kovalevyista katsotaan ainoastaan C: aseman tiedot.
# C:asemasta otetaan ylös koko ja vapaa tila.

 $hwHDD = Get-WmiObject Win32_LogicalDisk -Filter "DeviceID='C:'"
```

```
# Haetaan koko muuttuja win32_logicaldiskistä ja muutetaan gigatavuiksi
 $hwHDDsize=($hwHDD.Size)/1gb

# Kovalevytilan pyöristys nollaan desimaaliin:
 $hwHDDsize = "{0:N0}" -f $hwHDDsize

# Haetaan vapaa tila muuttuja win32_logicaldiskistä ja muutetaan gigatavuiksi
 $hwHDDfree=($hwHDD.FreeSpace)/1gb

# Vapaan kovalevytilan pyöristys nollaan desimaaliin:
 $hwHDDfree = "{0:N0}" -f $hwHDDfree

# Haetaan äänikortit valmistajien nimien mukaan.

      #1. tallennetaan äänikorttien tiedot väliaikaiseen tiedostoon $env:temp\tempaudio.txt

    Get-WmiObject Win32_SoundDevice | select name > $env:temp\tempaudio.txt

   # 2. Karsitaan tekstitiedostosta ylimääräiset vaihtoehdot pois ja tallennetaan uuteen teksti-
tiedostoon.
         # Pattern kentän jälkeen tulevat halutut äänikorttivalmistajat!

    $hwAUDIOdevice=Select-String -Pattern conex-
ant,realtek,sigmatel,maestro,soundmax,idt,legacy,intel `
          -Path "$env:temp\tempaudio.txt" | FOREACH { $_.Line }

   # 3. poistetaan väliaikainen tiedosto johon verkkolaitteet tallennettiin
    remove-item $env:temp\tempAUDIO.txt

##### Installed Software ##############
 #
       # Haetaan asennetut ohjelmat nimen mukaan, siirretään väliaikaisesti tiedostoon
%temp%/TempProduct.txt
      Get-WmiObject Win32_Product | Select Name > $env:temp\TempProduct.txt

       # Valitaan tiedostosta halutut ohjelmat, ja tehdään uusi väliaikainen tiedosto.
       # Nämä ohjelmat ovat niitä joita halutaan seurata.
       # Listaan lisättävät ohjelmat tulee lisätä komennon "Get-WmiObject Win32_Product |
Select Name" kanssa
          Select-String -Pattern `
                "Microsoft Office Professional Plus 2010", `
                "Microsoft Office Visio S", `
                "Microsoft Office Project", `
                "Adobe Acrobat", `
                "Antivirus",`
                "7-Zip", `
                "Google Chrome", `
                "M-Files 1", `
                "Microsoft Lync 2010", `
      "Adobe Reader", `
```

```
            " Encryption Client", `
        -Path "$env:temp\TempProduct.txt" | FOREACH { $_.Line } > $env:temp\TempProduct2.txt

        # Lasketaan että monta ohjelmaa löytyi, jotta ohjelmat voidaan syöttää tietokantaan järkevästi taulun Array:n avulla
    # Tässä kohdassa pitää olla aina samat ohjelmat kuin ylläolevassa Select-String lausekkeessa!
        $AppCalculator = (Select-String -Pattern `
            "Microsoft Office Professional Plus 2010", `
            "Microsoft Office Visio S", `
            "Microsoft Office Project", `
            "Adobe Acrobat", `
            "Antivirus",`
            "7-Zip", `
            "Google Chrome", `
            "M-Files 1", `
            "Microsoft Lync 2010", `
            "Adobe Reader", `
            "Encryption Client", `
        -Path "$env:temp\TempProduct.txt" | FOREACH { $_.Line } ).count

        # tehdään uusi Array, ja syötetään For lausekkeella ohjelma siihen.
        # Luodaan IF-lausekkeella vikasietoisuutta, ohjelma kaatuu jos yhtään yllä olevaa ohjelmaa ei ollut asennettuna.
    # IF kiertää tämän ongelman, ja suorittaa For -lausekkeen vain jos ohjelmia on enemmän kuin 0

    if($AppCalculator -ne 0) {
        $InstalledApps = @()
        For($i=0; $i -le $AppCalculator; $i++){ $InstalledApps+= (Get-Content $env:temp\TempProduct2.txt)[$i] }
    }
        # Visman L7 asennus ei ilmesty Windowsin rekisteriin, joten pitää tutki eritavalla onko sitä asennettu.
        # Elikkä katsotaan, että löytyykö C: aseman juuresta kansiota Visma\L7

        $visma = Get-ChildItem -Force C:\visma -ErrorAction SilentlyContinue | `
        Where-Object { ($_.PSIsContainer -eq $true) -And  ( $_.Name -Like "*L7*") } `
        | Select-Object Name

        # Käydään IF-lausekkeella läpi, että löytyikö kansiota
        # ja jos löytyi, niin lisätään $InstalledApps muuttuujaan Visma L7.

        if($visma) { $InstalledApps+="Visma L7" } else { $error.Clear() }

        # Poistetaan väliaikaiset tiedostot
    Remove-Item $env:temp\TempProduct.txt
    Remove-Item $env:temp\TempProduct2.txt
```

```
  ##### ALL VARIABLES FOR CMDB COLLECTED, TIME TO SAVE THEM TO DATABASE #######

    # 1. Muodostetaan insert lausekkeet:

    $MASTER_Query = “INSERT INTO master VALUES ('$ID','$OS_Hostname','$NULL','$NULL','$null','$PC_Manufacturer','$PC_Model','$NULL')"

    $OS_Query = “INSERT INTO OS(OS_version,OS_hotfix,OS_language,OS_bit,OS_InstallDate,id_master) `
            VAL-
UES('$OS_Version','$OS_HotFix','$OS_Language','$OS_Bit','$OS_InstallDate','$ID')"

    $HW_Query = “INSERT INTO HW(HW_CPU,HW_CPUcores,HW_RAM,HW_RAMslots,HW_RAMinst,HW_HDDsize,HW_HDDfr
ee,HW_AudioDevice,id_master)`
            VAL-
UES('$hwCPU','$hwCPUCores','$hwRAM','$hwRAMslot','$hwRAMinst','$hwHDDsize','$hwHDDfr
ee','$hwAUDIOdevice','$ID')"

    For($l=0; $l -le $AppCalculator-1; $l++){ $SW_Query += "INSERT INTO SW(SW_SOFTWARE,id_master) values('" + $InstalledApps[$l] +"','"+$ID+"');" }

    For($j=0; $j -le $hwLANlaskuri -1; $j++){ $NIC_Query += "INSERT INTO NIC(NIC_DEVICE,id_master) values('" + $hwLANdevice[$j] +"','"+$ID+"');" }

 # Notify viestiä varten tehty, jos $ilmoitus=1 tai 0, niin tulostetaan eri viestit
 $ilmoitus="0"
}
else {
      <# tehdään tyhjät insert lausekkeet jos arvot olivat jo kannassa #>
      $MASTER_Query=$null
      $OS_Query=$null
      $HW_Query=$null
      $SW_Query=$null
      $nic_Query=$null

      # Notify viestiä varten tehty, jos $ilmoitus=1 ta 0, niin tulostetaan eri viestit
      $ilmoitus="1"
}

    #2. Suoritetaan insert lausekkeet (Jos tiedot olivat jo kannassa, ajetaan tyhjät queryt)
     $command = $connection.CreateCommand()
     #$command.CommandText = $SW_Query
     $command.CommandText = $MASTER_Query +";"+ $OS_Query +";"+ $HW_Query +";"+ $USER_Query +";"+ $SW_Query +";"+ $NIC_Query
     $FinalQuery = $command.ExecuteReader()

   $Connection.Close() ## Suljetaan yhteys tietokantaan.
```

```
# Tulostetaan käyttäjälle ilmoitus skannauksen tulostuksesta. Tämä perustuu $error muuttujaan.

  if (!$error.count -And $ilmoitus -eq '0') {
        $objNotify.BalloonTipText = "Järjestelmän skannaus laitekantaan suoritettu onnistuneesti"
        $objNotify.BalloonTipTitle = "IT-tuki"
        $objNotify.Visible = $true
        $objNotify.ShowBalloonTip(48)
  } elseif (!$error.count -And $ilmoitus -eq '1') {
        $objNotify.BalloonTipText = "Järjestelmän päivitys laitekantaan onnistui"
        $objNotify.BalloonTipTitle = "IT-tuki"
        $objNotify.Visible = $false
        $objNotify.ShowBalloonTip(32)
  } else {
        $objNotify.BalloonTipText = "Järjestelmän skannauksessa ilmeni virhe, oletko yhteydessä IT-tukeen, kiitos!"
        $objNotify.BalloonTipTitle = "IT-tuki"
        $objNotify.Visible = $true
        $objNotify.ShowBalloonTip(66)
  }

$error.clear()
```