Perinnejärjestelmän alasajo ja siihen tarvittavat toimenpiteet

Kiia Kassila

Opinnäytetyö

Informaatio- ja mediatekniikka

2013

OPINNÄYTE	
Arcada	
Koulutusohjelma:	Informaatio- ja mediatekniikka
Tunnistenumero:	4098
Tekijä:	Kiia Kassila
Työn nimi:	Perinnejärjestelmän alasajo ja siihen tarvittavat toimenpiteet
Työn ohjaaja (Arcada):	Hanne Karlsson
Toimeksiantaja:	Neste Oil Oyj

Tiivistelmä:

Tämä opinnäytetyö pohjautuu Neste Oil:n lähettämöjärjestelmä LÄJÄ:än. Opinnäytetyössä kerrotaan järjestelmän elinkaaren eri vaiheet alasajoon asti.

Tavoitteena oli ajaa alas yrityksen käyttämä lähettämöjärjestelmä, joka on hyvä esimerkki niin sanotusta perinnejärjestelmästä. Tehtävänä oli ottaa selvää mitä tietoa pitää säilyttää, kuinka kauan ja missä se säilytetään. Tiedonsiirtoa varten piti tehdä sopiva työkalu, sekä pohtia minkälainen työkalu sekä ohjelmointikieli sopii tähän käyttöön.

Työssä käytettiin ASP-ohjelmointikieltä arkistointityökalun kehittämiseen. ASP osoittautui hyväksi valinnaksi, sillä se on helposti muokattavissa ja siitä saa tehtyä monipuolisia työkaluja. Työn teknisessä osassa kerrotaan työkalun tekemisestä, sekä siitä millä tavalla se toimii. Teoriaosuuden tarkoituksen on selventää itse aihe ja siihen liittyvät käsitteet. Tämä helpottaa ymmärtämään työn kokonaisuuden.

Työ toteutettiin ensin opiskelemalla ja tutkimalla aihetta. Tämän jälkeen päästiin itse työntekoon, kun kaikki tarvittava tieto oli saatu selville.

Tarkoituksena oli myös dokumentoida alasajo sekä arkistointi, jotta työn tilaaja pystyy tulevaisuudessa hyödyntämään tämän alasajon dokumentaatiota muissa vastaavissa projekteissa.

Lopputuloksena oli onnistunut alasajo, tarvittavat tiedot saatiin arkistoitua talteen sekä laitteelle tilattiin kuljetus metalliromuttamolle.

Avainsanat:	Neste Oil Oyj, perinnejärjestelmä, alasajo, arkistointi, ASP, lähettämöjärjestelmä
Sivumäärä	42
Kieli:	Suomi
Hyväksymispäivämäärä:	2.5.2013

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations – och medieteknik
Identifikationsnummer:	4098
Författare:	Kiia Kassila
Arbetets namn:	Nedkörning av ett legacy-system samt relaterade åtgärder
Handledare (Arcada):	Hanne Karlsson
Uppdragsgivare:	Neste Oil Oyj

Sammandrag:

Detta examensarbete baserar sig på Neste Oil:s försändelsesystem LÄJÄ. I examensarbetet introduceras olika faser av systemets livscycel ända till nedkörning.

Syftet var att köra ner företagets försändelsesystem, som är ett bra exempel på ett så kallat legacy-system. Uppgiften var att ta reda på vilken information som bör arkiveras samt hur länge och var. För att nå det bästa slutresultatet, var man tvungen att skapa ett lämpligt verktyg, och att överväga vilken typ av verktyg samt programmeringsspråk som lämpar sig för detta ändamål.

I arbetet användes ASP-programmeringsspråk vid framtagning av arkiveringsverktyget. ASP visade sig vara ett bra val, eftersom det är lätt att modifiera och man kan göra mångsidiga verktyg. I arbetets tekniska del beskrivs verktyget samt på vilket sätt det fungerar. Teoridelen är för att klargöra syftet med temat och relaterade ämnen. Detta gör det lättare att förstå hela arbetet.

Arbetet utfördes först genom att studera och undersöka ämnet. Efter det började man med själva arbetet, då man samlat in all information.

En annan målsättning var att dokumentera nedkörningen och arkiveringen, så att företaget kan dra nytta av dokumentationen från nedkörningen och tillämpa den i liknande projekt.

Slutresultatet blev en lyckad nedkörning, all nödvändig information arkiverades, och att hårdvaran efter detta kunde återvinnas i form av metallskrot.

Nyckelord:	Neste Oil Oyj, legacy-system, nedkörning, arkivering, ASP, försändelsesystem
Sidantal:	42
Språk:	Finska
Datum för godkännande:	2.5.2013

DEGREE THESIS	
Arcada	
Degree Programme:	Information and Media technology
Identification number:	4098
Author:	Kiia Kassila
Title:	The shutdown process of a Legacy system
Supervisor (Arcada):	Hanne Karlsson
Commissioned by:	Neste Oil Oyj

Abstract:

This thesis is based on Neste Oil's Dispatch system LÄJÄ and explains the different stages of a system life cycle through to the final shutdown stage.

The objective was to shut down the company's dispatch system, which is a good example of a Legacy system, and in the process decide what information should be stored and archived, for how long and where it should be stored. For the best results a suitable tool had to be developed taking into consideration the tool's function and appropriate programming language.

ASP-programming language was used for coding the archiving tool. ASP turned out to be a good choice because it is easily customized and it can be used to make a wide range of tools. The technical section describes the development of the tool and how it operates. The theory is to clarify the purpose of the theme itself and related topics. This makes it easier to understand the entire work.

The work was accomplished first by studying and examining the topic. After necessary information had been collected data was analyzed and the tool was developed based on the results.

Yet another objective was to document the shutdown and archiving process, so that in the future the commissioner is able to take advantage of the documentation in similar projects.

The end result was a successful shutdown. The necessary information was archived, as well as a metal scrap transport was ordered for the server.

Keywords:	Neste Oil Oyj, legacy system, system shutdown, archiving, ASP, Dispatch system
Number of pages:	42
Language:	Finnish
Date of acceptance:	2.5.2013

SISÄLTÖ

Kuvat

SANASTO

ASP

Active Server Pages on ohjelmointikieli

COBOL

Commone Business Oriented Language on ohjelmointikieli

CSV

Comma-separated values on tiedostomuoto, joka muuntaa taulukkotiedostot tekstitiedostoksi

DLL

Dynamic-link Library

HTML

HyperText Markup Language

IIS

Internet Information Server

ISAPI

Internet Server Application Programming Interface

LÄJÄ

Lähettämöjärjestelmä

ESIPUHE

Haluan kiittää Neste Oil Oyj:tä tästä opinnäytetyömahdollisuudesta. Etenkin työnohjaajaa Kari Nummenpaloa, joka teki tästä projektista mahdollisen.
Haluan myös kiittää Hanne Karlssonia, työnohjaajaa Arcadassa, joka on neuvonut ja auttanut minua kirjoitusprosessissa.
Viimeiseksi haluan vielä kiittää kaikkia, jotka ovat tukeneet ja auttaneet minua koko opiskelujen ajan.

1 JOHDANTO

Tekniikan mennessä eteenpäin ja asiakasmäärien kasvaessa on järjestelmien pystyttävä palvelemaan uusia tarpeita ja toimimaan yrityksen muiden järjestelmien kanssa yhdessä. Vanhat järjestelmät vievät yritykseltä resursseja sekä niiden ylläpito on kallista. Opinnäytetyössä tutustutaan lähettämöjärjestelmään, joka on hyvä esimerkki niin sanotusta perinnejärjestelmästä, tiedon arkistointiin ja sen työkalun tekemiseen sekä alasajoon.

1.1 Tavoitteet ja tarkoitus

Tarkoituksena on ajaa alas Neste Oil:n käyttämä lähettämöjärjestelmä sekä selvittää mitä alasajolla ja perinnejärjestelmällä tarkoitetaan. Lisäksi pitää ottaa selvää mitä tietoa pitää säilyttää, kuinka kauan ja missä. Tiedonsiirtoa varten pitää kehittää sopiva työkalu, sekä pohtia minkälainen työkalu ja ohjelmointikieli sopii tähän käyttöön.
Tavoitteena on myös dokumentoida koko alasajoprosessi sekä arkistointi, jotta työn tilaaja pystyy tulevaisuudessa hyödyntämään tämän alasajon dokumentaatiota muissa vastaavissa projekteissa.

1.2 Menetelmät

Alasajo-projekti aloitetaan haastatteluilla sekä kirjallisuustutkimuksella. Eri käyttäjiltä saadaan tietoa siitä, mitkä tiedot pitää arkistoida, kuinka kauan sitä säilytetään sekä mitä tehdään lisenssien ja laitteiston kanssa. Työkalun tekeminen ja arkistointi onnistuu sen jälkeen kun kaikki tarvittavat tiedot on kerätty. Viimeisenä osana laitteisto ajetaan alas ja toimitetaan romuksi. Teoria käsittelee alasajon, perinnejärjestelmän, arkistoinnin sekä selvennyksen ohjelmointikieleen, jota työkalun tekemisessä käytetään.

1.3 Tilaajan esittely

Tilaajana toimii Neste Oil Oyj, suomalainen öljynjalostus - ja markkinointiyhtiö, joka keskittyy korkealaatuisiin puhtaamman liikenteen polttoaineisiin. Yhtiö valmistaa uusiutuvia polttoaineita ja kaikkia tärkeimpiä öljytuotteita. Toimintaa on 15 maassa, muun muassa Hollannissa, Singaporessa ja Kanadassa. Suomessa jalostamot ovat Naantalissa

ja Porvoossa. Porvoon Kilpilahdessa olevat tuotantolaitokset ovat Pohjoismaiden suurin kemian prosessiteollisuuden keskittymä. Neste Oil:lla työskentelee noin 5000 henkilöä, joista noin 1900 Porvoossa. (Neste Oil vuosikertomus, 2011). Tilaus tehdään tuotanto- ja logistiikka-osastolle, jonka tehtävänä on varmistaa, että yhtiön tuotteet valmistetaan ja kuljetetaan asiakkaalle mahdollisimman luotettavasti ja kustannustehokkaasti.

2 PERINNEJÄRJESTELMÄ

Perinnejärjestelmä tulee englannin kielen sanoista Legacy system. Perinnejärjestelmä on yritykselle usein taakka sen monimutkaisuuden vuoksi ja järjestelmän käyttäjät ja ylläpitäjät kärsivät siitä (Datpro, 2012). Vanhentuneet järjestelmät vievät usein turhaan yrityksen resursseja muun muassa lisenssien ja työlään ylläpidon takia.

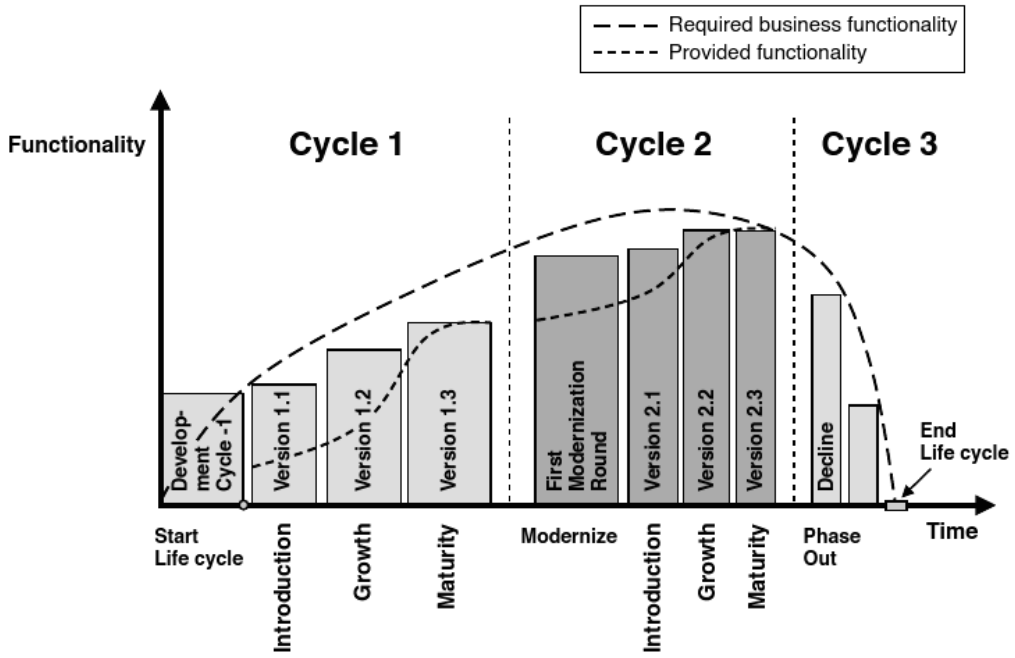
Perinnejärjestelmä koostuu laitteistoista ja järjestelmistä, jotka ovat suhteellisen vanhoja, suurtietokone-pohjaisia (mainframe) järjestelmiä ja ne on optimoitu mukautumaan muistin, tallennustilan, muun ohjelmistoalustan sekä käyttöjärjestelmän toiminnallisiin rajoitteisiin. Useat perinnejärjestelmät ovat yli 20 vuotta vanhoja ja ohjelmoitu COBOL:lla, PL/1:llä tai Assembly/370:llä. (van den Heuvel W. J. 03/2007, s.21)

Ohjelmistoista tulee perinnejärjestelmiä, kun ne alkavat vastustaa muutosta ja kehitystä. Perinnejärjestelmäksi lasketaan myös ne järjestelmät, joiden alkuperäinen suunnittelija ei ole enää tavoitettavissa. Jos alkuperäinen ohjelmoija on edelleen yrityksen palveluksessa, ei hän välttämättä enää uskalla muuttaa koodia. Jos koodia muutettaessa tulee ongelmia, ei hän välttämättä ymmärrä koodia, eikä osaa korjata virheitä. Perinnejärjestelmissä oleva tieto on edelleen merkittävä yrityksen voimavara. Olettaen että nämä järjestelmät tarjoavat edelleen merkittävää arvoa liiketoiminnalle, pitää ne joko nykyaikaistaa tai korvata. (Seacord, Robert C. 2003, s.5)

2.1 Järjestelmän elinkaari

Kuten monet uudistukset ja muutokset yrityksissä, syntyvät sovelluksetkin lupaavina järjestelminä ja täynnä mahdollisuuksia. Tästä huolimatta nämä järjestelmät vanhenevat ajan myötä ja niitä on vaikea saada toimimaan tehokkaasti muuttuvan liiketoiminnan

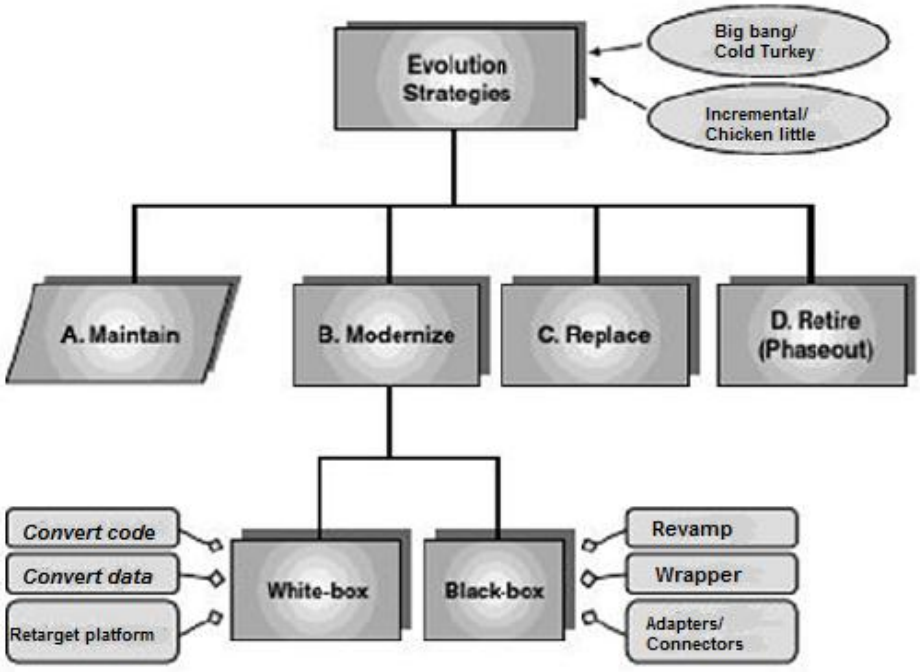
kanssa. Useissa tapauksissa järjestelmän arkkitehtuuri ei ole enää sopiva nykytilanteen kanssa, joten järjestelmä vaatisi uudistamista. Ilman uudistamista järjestelmän rakenne heikkenee ja siitä voi tulla yhteensopimaton yrityksen vaatimusten ja muiden järjestelmien kanssa.



Kuva 1. Järjestelmän elinkaari. Mukautettu Seacord, Plakosh and Lewis (2003, 8) (van den Heuvel W. J. 03/2007, s.24)

Yllä oleva kuva on jaettu kolmeen sykliin, joista jokainen kuvastaa yhtä ajanjaksoa (Kuva 1.). Monissa tapauksissa järjestelmän elinkaarella on taipumus seurata tiettyä kaavaa uudistamisyrityksineen. Kaksi ensimmäistä jaksoa noudattavat S-kirjaimen muotoa. Järjestelmän kehityksen ensimmäinen sykli on aloitusvaihe. Toisessa syklissä sovellusta uudistetaan ja sovelluksen sopivuutta laajennetaan liiketoiminnan kehittyessä. Viimeinen vaihe näyttää väärinpäin käännetyltä S-kirjaimelta, joka kuvastaa sitä, että sovellusta asteittain yritetään poistaa käytöstä.

2.2 Kehittymisstrategia



Kuva 2. Kehittymisstrategioita perinnejärjestelmän hallitsemiseen, (van den Heuvel W. J. 03/2007, s.25*)*

Järjestelmän elinkaaren vaiheissa voidaan käyttää erilaisia strategioita sen kehittämiseen (Kuva 2.). Järjestelmän elinkaaristrategian merkitys vaihtelee minimaalisesta merkittävään, järjestelmästä riippuen. Näistä strategioista vaikein on järjestelmästä luopuminen (Kuva 2. D.), jonka seurauksena perinnejärjestelmä ajetaan alas.

Ylläpito (Kuva 2. A.) on hyvä vaihtoehto silloin, kun perinnejärjestelmä toimii vielä suhteellisen hyvin. Kustannusten ja riskien näkökulmasta tämä on edelleen suosituin vaihtoehto. Ylläpidolla tarkoitetaan pieniä muutoksia järjestelmään, ilman että koodiin tehdään suuria muutoksia tai että sen pohjalla olevaa arkkitehtuuria rikotaan.

Nykyaikaistaminen (Kuva 2. B.) tulee kuvioihin siinä vaiheessa, kun ylläpitoa on tehty useita vuosia, mutta se on heikentänyt järjestelmän teknillistä laatua ja joustavuutta. Nykyaikaistamisen voi tehdä kahdella tavalla. Ensimmäinen vaihtoehto on sen kunnostaminen integroimalla se uusilla sovelluksilla tai komponenteilla. Tätä vaihtoehtoa kutsutaan nimellä ”black-box”. Toinen vaihtoehto on järjestelmän muutos. Tämä vaatii järjestelmän täydellistä ymmärtämistä, jotta sen koodia ja tietoja voidaan muuttaa. Tätä

kutsutaan nimellä ”white-box”. Näiden kahden vaihtoehdon suurin ero on siinä, että ensimmäisessä vaihtoehdossa ei järjestelmää tarvitse juurikaan ymmärtää, kun taas toisessa vaihtoehdossa se on välttämätöntä.
Väistyvän järjestelmän korvaaminen (Kuva 2. C.) vaatii joko uuden järjestelmän rakentamista tai valmiin järjestelmän ostamista ja räätälöimistä. Uusi järjestelmä saattaa näyttää hallinnolle erittäin lupaavalta sen nykyaikaisen teknologian ansiosta, mutta eräitä asioita on otettava tässäkin huomioon. Jotta perinnejärjestelmään tehtyjä viimeisimpiä investointeja voidaan hyödyntää, on tehtävä muutoksia arvokkaaseen ja monimutkaiseen ohjelmakoodiin sekä dataan. Uusi järjestelmä ei todennäköisesti täysin vastaa korvattavaa järjestelmää ja se saatetaan korvata jopa useammalla järjestelmällä. (van den Heuvel W. J. 03/2007, s.23-26)

2.3 Perinnejärjestelmän alasajo

Kehittymisstrategian viimeisin vaihe on järjestelmän alasajo, joka tarkoittaa sitä, että järjestelmä poistetaan käytöstä (Kuva 2. Retire). Poistettavasta järjestelmästä on tarvittavat tiedot siirrettävä joko migraation avulla uuteen järjestelmään tai arkistoitava pitkäaikaiseen, edullisempaan säilöön.

Perinnejärjestelmän alasajon tärkeys ja hyödyt jäävät usein uuden järjestelmän käyttöönottoprojektin jalkoihin. Hyödyt ovat usein merkittäviä, mutta vaikeasti mitattavia.

Monissa organisaatioissa uusien järjestelmien käyttöönottoprojekteissa projektipäälliköitä mitataan vain aikataulun ja budjetissa pysymisen mukaan. Perinnejärjestelmän alasajoa pidetään vain hyvänä tapana, eikä tärkeänä osana uuden järjestelmän käyttöönottoa. Sen vuoksi vanhan järjestelmän alasajo määritellään vain uuden järjestelmän käyttöönottoprojektin päätösvaiheeksi.
Alasajo saattaa olla yritykselle kallis, mutta pitkällä tähtäimellä kannattava vaihtoehto. Sen hyödyt tulevat yritykselle vanhan järjestelmän kulujen poistuessa. Tässä ei vielä ole otettu huomioon yhden järjestelmän vaikutusta yrityksen tai organisaation järjestelmien kokonaisuuteen. Järjestelmien määrä yrityksessä saattaa olla todella suuri ja yhden järjestelmän poistaminen tuntuu mitättömältä. Jos taas katsoo kaikkien järjestelmien kokonaisuutta, niin pienikin yksinkertaistaminen tuo helpotusta.

Yksi tapa arvioida alasajon hyötyjä on listata asiat, jotka yksinkertaistuvat poistamalla vanha järjestelmä:

- Kahden järjestelmän aiheuttama päällekkäinen työ
- Pienentää operatiivisia kustannuksia
- Uusien järjestelmien käyttöönotto on helpompi perustella, kun vanhasta järjestelmästä voidaan luopua

Ongelmana on että arvokkain noista kolmesta hyödystä on viimeinen. On hankalaa arvioida milloin alasajosta hyötyvä järjestelmä saadaan käyttöön. Vielä huonomminkin voi käydä, jos hankkeiden käynnistämiseen menee liian kauan, saattaa järjestelmän käytöstä poisto olla liian myöhäistä. Tästä syystä kannattaa ottaa käyttöön taloudellinen mallinnus kaikkien järjestelmien kokonaisuuteen ja niiden olemassaoloon, kuin ryhtyä todistelemaan säästöjä luettelemalla listaa jo käynnistetyistä yksittäisistä projekteista. (Deloitte. 2012)

2.4 Alasajon valmistelut

Seuraavia tehtäviä voidaan hyödyntää alasajon valmistelussa:

- Suunnittele alasajo, roolit ja vastuut alasajossa
- Laadi ja toteuta riskienhallinta alasajossa
- Dokumentoi riskienhallinnan lähestymistapa. Mitkä tiedot on säilytettävä ja siirrettävä uuteen järjestelmään
- Mitkä tiedot arkistoidaan ja kuinka pitkäksi ajaksi
- Määritä, kuinka tiedot arkistoidaan, missä muodossa ja minne
- Listaa toimet sisäisen ja ulkoisen tuen lopettamiseen
- Määritä tarvittava muutoksenhallinta poistuvalle järjestelmälle
- Määritä tiedottamiskeinot poistuvan järjestelmän vaikutuksista
  (USDM. 2013)

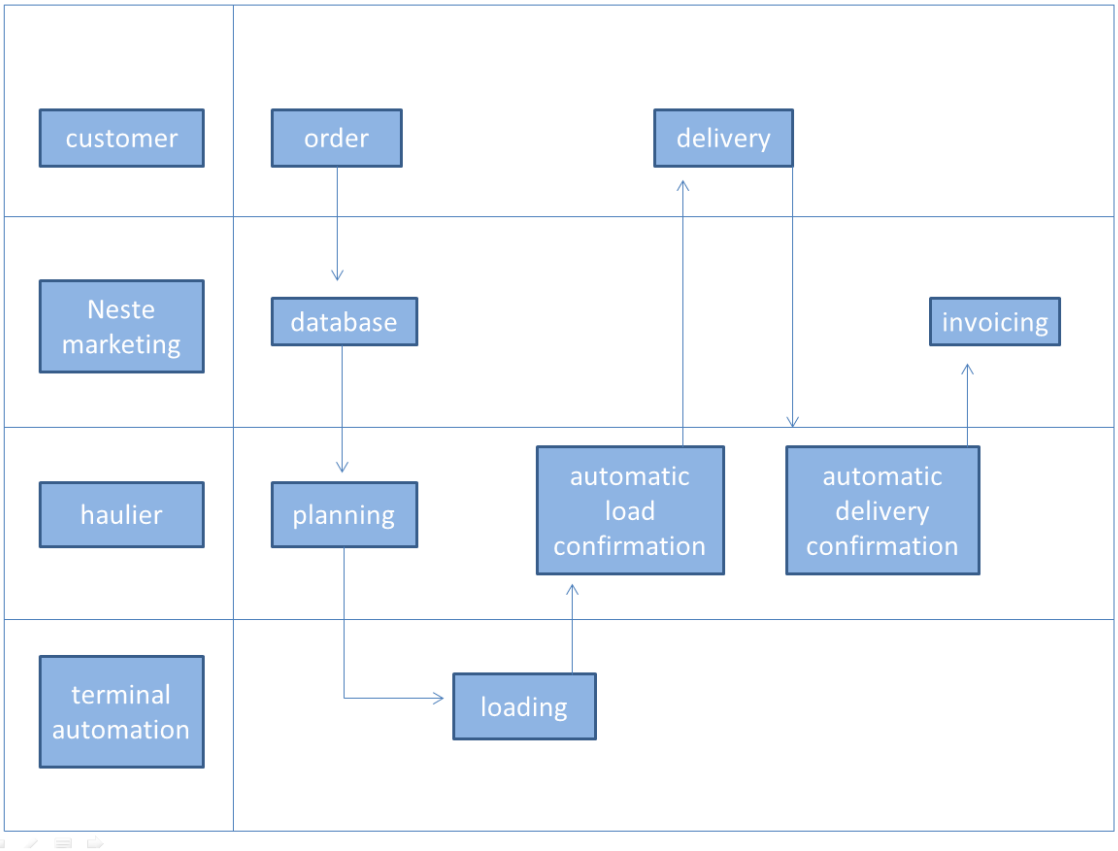
3 LÄHETTÄMÖJÄRJESTELMÄ

Lähettämöjärjestelmä (LÄJÄ) on järjestelmä, joka on otettu käyttöön kesällä vuonna 1984. Järjestelmä on ohjelmoitu COBOL-ohjelmointikielellä. Tällä hetkellä yrityksessä on vain yksi henkilö, joka ymmärtää ohjelmakoodin ja muun muassa sen vuoksi LÄJÄ lasketaan perinnejärjestelmäksi. LÄJÄ on säiliöautojen tapahtumajärjestelmä, jolla hoidetaan kuljetusten suunnittelu. Tilaukset suunnitellaan erillisiin toimituskuormiin. Järjestelmässä Suomi on jaettu maantieteellisiin alueisiin postinumeroittain ja kuljetusliikkeet, jotka hoitavat tuotteiden kuljetukset, hoitavat omien alueittensa kuljetusten suunnittelun.

Kuormansuunnittelussa syntyy lastaus- ja toimitussuunnitelma automaatiolle. Lastausselvittelyssä kohdistetaan lastaussuunnitelmalta ja automaatiolta saadut tiedot, eli tarkistetaan onko lastattu sitä, mitä on suunniteltu. Toimitusselvitykseen tallennetaan jokaisen yksittäisen asiakastapahtuman tieto, kuten esimerkiksi se kenelle kuljetettiin, mitä tuotetta, milloin ja kuinka paljon.

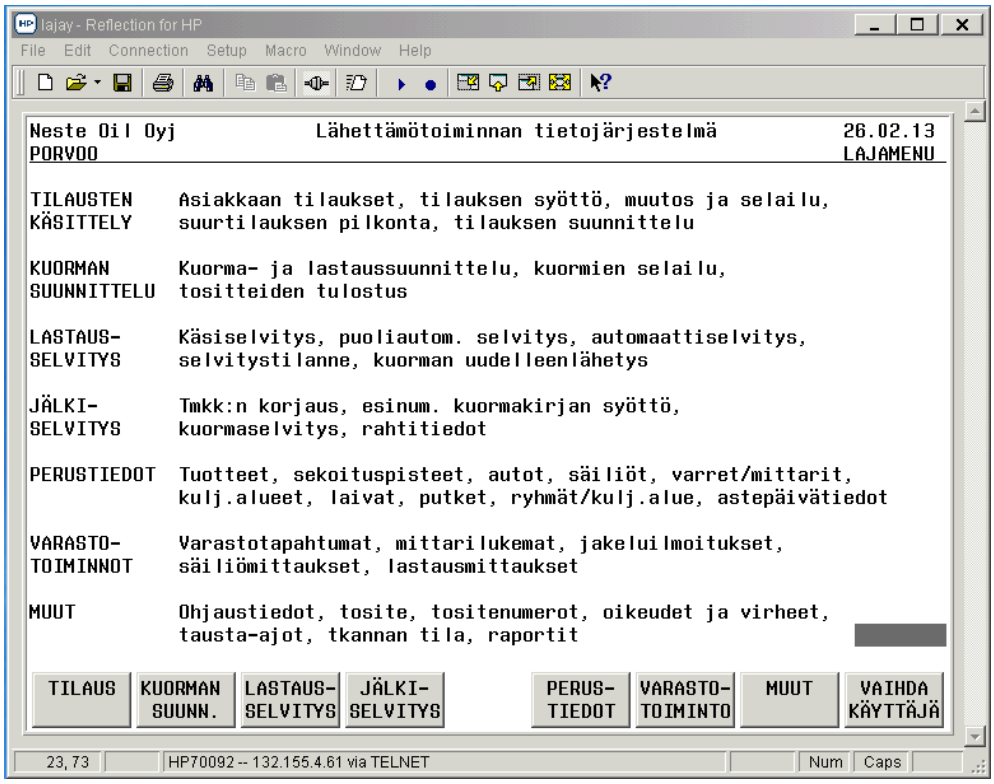
Vuoteen 1995 saakka tämä kaikki tehtiin manuaalisesti. Tositteet käytiin käsin läpi, kirjattiin LÄJÄ:ään, ja sieltä tiedot siirrettiin automaattisesti suurtietokone-pohjaiseen laskutusjärjestelmään.

Järjestelmää kehitettiin jatkuvasti ja loppujen lopuksi se oli niin automatisoitu, että kaikki tapahtumat, tilauksesta toimitukseen, hoituivat automaattisesti. Tilauksen kirjaamisesta järjestelmään, aina toimitukseen saakka, hoitivat kuljetusliikkeet. (Kuva 3.) He hoitivat 95 % kuljetuksiin liittyvistä kirjauksista ilman Neste Oil:n henkilöstön työpanosta. Tähän liittyvät ongelmat tulivat esiin silloin, jos jotain poikkeavaa tapahtui. Henkilöitä, jotka osasivat korjata ongelman, ei aina ollut saatavilla.



Kuva 3. Uimarataprosessikaavio lähettämöjärjestelmästä

Kuva 4. esittää LÄJÄ:n valikkoa. Kuvan alareunassa on kuvaukset funktionäppäimistä, joilla siirryttiin eteenpäin halutulle sivulle. Tämä valikko on esimerkkinä kuvastamaan sitä, miltä vanha järjestelmä nykypäivän järjestelmiin verrattuna näyttää. Sama valikkonäkymä on ollut alusta asti käytössä.



Kuva 4. Lähettämöjärjestelmän aloitusvalikko

3.1 Lähettämöjärjestelmän elinkaari

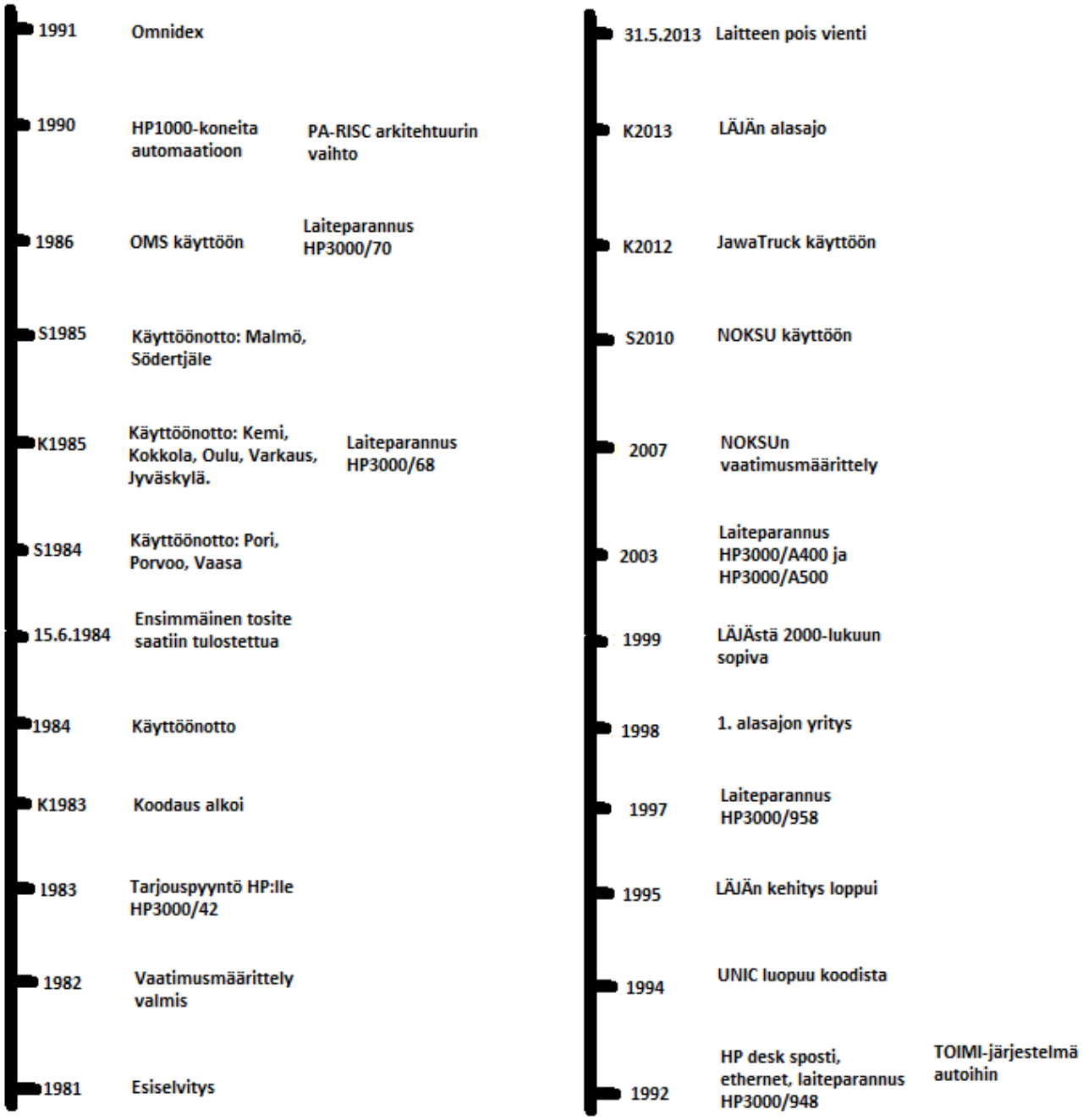
Suunnittelu ja ohjelmointi aloitettiin vuonna 1983. Seuraavana vuonna otettiin ensimmäinen lähettämö järjestelmän piiriin ja vuoden 1984 joulukuussa otettiin viimeinenkin lähettämö sen piiriin. Laitteiston hankintavaiheessa vertailtiin eri laitevalmistajien tuotteita, jotka kilpailutettiin siten, että paras mahdollinen laitteisto saataisiin käyttöön. Vertailuissa oli mukana muun muassa Digital sekä Hewlett Packard (HP), jonka toimittama HP3000-laitteisto valittiin järjestelmän alustaksi. Vertailussa sekä Digital että HP sai yhden kuukauden aikaa valmistaa oma ehdotuksensa, joista toinen valittaisiin. Järjestelmä on ohjelmoitu COBOL-ohjelmointikielellä ja ohjelmointi aloitettiin vuonna 1983. Ohjelman on tehnyt Unic, joka teki yhteistyötä HP:n kanssa. Seuraavana kesänä 1984 järjestelmä saatiin käyttöön ja tarpeellisia parannuksia tehtiin vuoden loppuun saakka.

Alun perin käytössä oli kolme tuotannon konetta, mitkä ajan myötä vähennettiin yhdeksi koneeksi. Alkuperäiset koneet oli HP3000/42, mutta parantelun myötä vaihdettiin

muun muassa HP3000/68 koneiksi (Kuva 2. Modernize). Viimeisimpänä käytössä olleet koneet olivat HP3000/A400 sekä HP3000/A500, toinen tuotannossa ja toinen testikoneena. LÄJÄ:ä on käytetty myös Ruotsissa Malmössä sekä Södertäljessä. Näitä varten ohjelma oli käännetty myös ruotsiksi. LÄJÄ:n ensimmäistä alasajoa suunniteltiin jo vuonna 1998 (Kuva 2. Replace). Tällöin korvaajana olisi ollut Oraclen tuote. Tarkoituksena oli ottaa käyttöön graafinen käyttöliittymä, mutta Oraclella ei ollut valmista kehityspolkua siihen. Uuden järjestelmän käyttöönottoyritys oli suuri, jonka takia Keminterminaalissa LÄJÄ korvattiin Oraclen tuotteilla, mutta vaihdettiin melkein heti takaisin LÄJÄ:ksi.

LÄJÄ:n korvaajat NOKSU (Neste Oil Kuljetuksen Suunnittelu) ja JawaTruck tuli kuvioihin vuonna 2010 ja 2012. Alla aikajana LÄJÄ:n tapahtumista. (Kuva 5.)

Ennen järjestelmää käytettiin päätteillä, jotka olivat päätehuoneissa. Ethernet-verkon ja henkilökohtaisten työasemien yleistyttyä, järjestelmää oli mahdollista käyttää, nykypäivään nähden normaalimmin, omalta työpisteeltään. Unic:n koodista luopuminen tarkoitti sitä, että yhteistyö heidän kanssaan päättyi ja Neste hoiti tarvittaessa koodauksen itse siitä eteenpäin (Kuva 2. Maintain).
Suurempia ongelmia ei järjestelmän kanssa ole tullut. Yksi suurehko muutos liittyi vuosituhannen vaihteen vuosilukujen käsittelyyn, sillä ilman muutosta vuodet olisivat siitä eteenpäin käsitelty väärin.



Kuva 5. Lähettämöjärjestelmän aikajana

3.2 LÄJÄ:n korvaava järjestelmä

LÄJÄ:n korvaa kaksi eri järjestelmää. Kuormansuunnittelu tehdään NOKSU:ssa ja lastaus-, toimitus- ja rahtausselvitys tehdään JawaTruck:ssa. Molemmat ovat Logican Neste Oil:lle tekemiä järjestelmiä.
Alun perin tavoitteena oli LÄJÄ:n korvaaminen yhdellä uudella järjestelmällä. Kuormansuunnittelu saatiin eriytettyä LÄJÄ:stä ja se hoidetaan pelkästään NOKSU:ssa. Vanhan ja uuden järjestelmän välillä ei tehty migraatiota. Alkutiedot on otettu vanhasta järjestelmästä uuteen liittymien kautta. Käyttöönotto tapahtui jakeluterminaaleittain. Koko käyttöönottoprojektiin meni aikaa kaksi vuotta. Uusien järjestelmien käyttöönotto aloitettiin vuonna 2010 Kemissä ja kaikki jakeluterminaalit olivat järjestelmien piirissä loppuvuonna 2012, jolloin uutta dataa ei enää viety LÄJÄ:ään. NOKSU sekä JawaTruck korvasivat sen täysin.

3.3 Tekniset tiedot

Laitteiston tekniset tiedot on esimerkki siitä, minkälainen laitteisto on ennen ollut käytössä. Vertailukohteena tämän päivän laitteistolle.

LÄJÄ:n tuotantopalvelinta ei voi klusteroida. Toimittajalla on käytössä järjestelmien kriittisyysluokittelu P1-P4, jonka mukaan LÄJÄ on luokkaa P2. P1 on kriittisin ja P4 on vähiten kriittinen.

Alla olevassa kuvassa on suuntaa-antava kuvaus järjestelmän kriittisyysluokituksesta P2. Taulukosta liiketoimintaan vaikuttavat tiedot–sarakkeen (Details of Business Impact) tiedot on jätetty pois, sillä ne tiedot pysyvät Neste Oil:lla salassa.

System down	Business Impact of Disruption				
	A	B	C	D	Details of Business Impact
<30min				X	
30..120min			X		
2h..4h		X			
4h..8h	X				
8h..12h	X				
12h..24h	X				
1...3day	X				

Kuva 6. Lähettämöjärjestelmän kriittisyysluokitus

Häiriöiden vaikutus liiketoimintaan (Kuva 6.):

A) Liiketoiminta uhattuna

B) Vakavaa vahinkoa (Serious damage)

C) Merkittävä vaikutus (Significant impact)

D) Vähäistä vahinkoa tai ei ollenkaan vaikutusta (Minor or no impact)

Liiketoimintaan vaikuttavat tiedot kuvaa minkälaista vahinkoa viivästyksistä seuraa. Kuvan ensimmäinen rivi kuvastaa sitä, että minkälaista vahinkoa siitä seuraa, jos järjestelmä on alhaalla alle 30 minuuttia. Tällöin vahinko on vähäistä. Liiketoiminta on uhattuna, jos alhaallaoloaika on yli neljä tuntia.

Alla oleva kuva on lista lähettämöjärjestelmän testikoneen laitetiedoista (Kuva 7.). Tuotantokoneena on HP3000/A500 kone. Testi- ja tuotantokone toimivat tarvittaessa toistensa sijaisina. Alun perin koneita oli kolme, Naantalissa, Porvoossa ja Keilaniemessä.

HP3000 on Hewlett-Packardin (HP) kehittämä ja valmistama tietokone. Kaikki HP3000 tietokoneet käyttävät MPE-käyttöjärjestelmää ja sen mukana tulee TurboIMAGE

tietokanta. Hewlett-Packardilta tuli 80-luvulla uusi Precision Architecture, Reduce Instruction Set Computing (PA-RISC) laitteistoarkkitehtuuri HP3000:lle.

Vaikka PA-RISC:ssa on täysin erilainen teknologia, sen käyttämä käyttöjärjestelmä MPE/iX pystyy ajamaan suurimman osan Classic HP3000 ohjelmista Compatibility Modessa. (Green, Bob)

Rivi	Laite	Kuvaus ja määrä
		HP 3000/A400 Server
1	A5136A	1 x Rack System/E41 1.96 meter cabinet
2	A7019C	1 x A500 1 way, 200MHz, 512MB, MPE
3	A7020A	1 x HP e3000 A500 200MHz CPU
4	A5840A	1 x 512MB High Density SyncDRAM Memory Mod
5	A6948A	1 x 36GB 15K HotPlug Ultra160 disk, rp24X0
6	A6828A	3 x PCI Ultra160 SCSI Adapter
7	A6395A	1 x HP e3000 A-Class Server rack mount kit
8	C1099A	1 x Terminal console for HP3000/9000 systems
9	C7508AZ	1 x HP Tape Array 5300 (factory-racked
10	C7498A	1 x HP DAT 24 Array Module (flint)
11	C7456A	1 x HP DLT 80 Array Module
12	A5218AZ	1 x Rear Door to RS/E41 1.96 meter cabinet
13	A5137AZ	1 x Modular Power Dist Unit for std racks
14	A5675AZ	2 x HP SureStore Disk System 2100 Fact. Rack
15	A6541A	4 x Add on 36 GB 15K RPM Ultra3 SCSI Drive
16	32651B	MPE-iX Operating system software
17	B3524A	Image/SQL software
18	B5152AA	TurboStore iX software

Kuva 7. Lähettämöjärjestelmän laite- ja laiteohjelmistotiedot

Käyttöjärjestelmä MPE/iX

MPE on lyhenne Multi-Programming Executive, joka on Hewlett-Packardin valmistama liiketoimintaan suunnattu minitietokonekäyttöjärjestelmä. (Multi-programming executive, 2013) Se toimii HP3000-tietokoneperheen koneissa. Alun perin käyttöjärjestelmää kutsuttiin nimellä MPE/XL. MPE:n alkuperäinen versio on ohjelmoitu System Programming Language:lla (SPL). Myöhemmin käyttöjärjestelmän nimi muutettiin osoittamaan Unix yhteensopivuutta lisäämällä Portable Operating System Interface (POSIX) yhteensopivuus, jolloin nimeksi tuli MPE/iX. (Robelle, a.)

Tietokanta turboIMAGE/XL (ALLBASE/SQL)

- 2-tasoinen verkkotietokanta
- Adaqer-tietokannanhallinta
- Minisoft odbc ajuri
- Omnidex-indeksointityökalu

IMAGE on yksinkertainen ja nopea tietokanta MPE:lle. Myöhemmin parannusten jälkeen se nimettiin turboIMAGE:ksi, jonka jälkeen vielä turboIMAGE/XL:ksi. Viimeisimmäksi se muuttui PA-RISC -migraation jälkeen. Tänä päivänä se tunnetaan nimellä IMAGE/SQL. (Robelle, b.)

ALLBASE/SQL tarjoaa toiminnallisesti rikkaan ja hyvin suorituskykyisen relaatiotietokannan hallintajärjestelmän HP3000 serverille. Tämä on suunniteltu vastaamaan liiketoiminnan tarpeita sovellusten kehittämisessä ja sen on tuettava liiketoimintakriittisten Structured Query Language (SQL) tapahtumien käsittelyä verkossa. (HP. 2009)
Minisoft Open Databse Connectivity (ODBC) ajuri UNIX:lle mahdollistaa suoran yhteyden TurboIMAGE tietokantaan. Se tukee linkitystä useiden tietokantojen välillä sekä MPE tiedostoja. (Minisoft. 2013)
Omnidex on hakumoottori, joka tuottaa nopeita tietokantahakuja käyttäen SQL-liitäntöjä. Omnidex:ä pidetään yhtenä tietokannan kerroksista, johon pääsee ODBC:n kautta. ODBC kommunikoi tietokantapalvelimen kanssa. Omnidex sen sijaan indeksoi ja päästää tarvittaessa tietokantaan käsiksi. (Omnidex. 2012)

Ohjelmointikieli COBOL

- Qedit editori ohjelmointia varten

4 ARKISTOINTI

Yksi tärkeä vaihe perinnejärjestelmän alasajossa on datan profilointi. Ennen datan arkistointia, se pitää analysoida. ”Datan profilointi” viittaa kertaluontoiseen datan laadun analyysiin. Tavoitteena profiloinnilla on selvittää, mikä on analysoitavan järjestelmän nykytilanne. Tyypillinen esimerkki profiloinnista on korvattavan perinnejärjestelmän datan analyysi ennen migraatiota uuteen järjestelmään. Profiloinnilla pienennetään varsinaisen migraation riskejä ja kustannuksia. (Datpro, 2012)

Tiedon arkistointi on passiivisen tiedon tunnistamista ja siirtämistä nykyisestä tuotannon järjestelmästä johonkin tiettyyn pidemmän ajan säilytyspaikkaan. Passiivisen tiedon siirtäminen pois tuotannonjärjestelmästä optimoi tallennukseen tarvittavia resursseja. Tiedot tallennetaan kustannustehokkaammin ja se tarjoaa myös mahdollisuuden päästä näihin tiedostoihin käsiksi vielä myöhemmin.
Tiedon arkistointi on olennaista yrityksissä, jotka keräävät uutta tietoa, mutta tarvitsevat edelleen vanhojakin tietoja. Tiedon sisällöstä riippuen, vanhoja tietoja saattaa tarvita myöhemmin esimerkiksi jossain asiakasta koskevassa epäselvässä vaiheessa, joka täytyy tarkistaa.
Organisaatio asettaa omat menettelytavat arkistoitavan tiedon valintaan. Nämä menettelytavat automatisoivat prosessin tiedon valintaan ja näin saadaan valittua tiedot, jotka arkistoidaan. Silloin kun tiedot ovat arkistossa, informaatio pysyy saatavilla. Tiedon siirtäminen ja säilyttäminen arkistoon vähentää kustannuksia. (EMC. 2013)

4.1 Arkistoitavan tiedon valintaan vaikuttavat seikat

Lähettämöjärjestelmästä löytyy kaikki kuljetuksiin liittyvät tiedot. Tiedoista löytyy myös verotiedot, joiden säilytysajan määrää laki. Jos kuljetuksissa sattuu virheitä, huomauttaa asiakas tästä viikon sisällä. Miksi sitten tietoja pitää säilyttää pitempään? Asi-

aan saattaa puuttua esimerkiksi verottaja, joka kyselee minkä takia osa kuormista on lähtenyt verottomana ja osa ei.

4.2 Arkistoitavan tiedon säilytysmuoto

Arkistoitavat tiedot haetaan tietokannasta. Tiedot tulevat taulukkomuodossa ja ne muutetaan tekstitiedostoiksi. Tässä käytetään tiedostomuotoa .csv. Valinta oli siinä mielessä yksinkertainen, sillä CSV-tiedostot on helppo avata Microsoft Excel:llä, joka on yrityksissä laajasti käytössä oleva ohjelmisto. Tämä tarkoittaa sitä, että arkistointiajan umpeutumisen lähestyessä tiedostot saadaan tarvittaessa vielä auki, sillä suurella todennäköisyydellä Excel on edelleen laajasti käytössä.

Comma-separated values (CSV)

Comma-separated values (CSV) on tiedostomuoto, jota käytetään silloin kun halutaan tallentaa taulukkomuotoista tietoa tekstitiedostoon. Tiedot taulukosta on muutettu tekstitiedostoksi ja sarakkeet on eroteltu yleisimmin joko pilkulla tai välilyönnillä. Tietokantaohjelma osaa tuoda tiedot CSV-muodossa. (Wikipedia. 2013a)

4.3 Arkistoinnin tietoturva

Arkistoon menevä tieto analysoitiin, jotta saatiin selville, minkälaista tietoa arkistoidaan. Arkistoitavat tiedot eivät sisällä arkaluontoista tietoa, kuten hintatietoja. Tämän takia arkistointipäätös ei tietoturvan kannalta ollut vaikea. Päätöksenä on, että tiedot siirretään tallennuslevylle, jossa niitä säilytetään kuusi vuotta. Tämä tiedon säilyttämisaika on lakisääteinen. Tiedot pidetään tallennuslevyllä, jonne vain osalla työntekijöistä on oikeudet. Oikeuksia myönnetään tarpeen tullen niitä tarvitseville.

5 ARKISTOINTITYÖKALU

Vertailussa käytettiin COBOL:ia (Commone Business Oriented Language) sekä ASP:tä (Active Server Pages). COBOL oli harkinnassa mukana, sillä itse järjestelmä on ohjelmoitu sillä kielellä.

5.1 COBOL

Commone Business Oriented Language (COBOL) on vanha ohjelmointikieli vuodelta 1959. Se on korkean tason ohjelmointikieli, ja näin ollen sen lähdekoodin pitää olla kirjoitettu sopimaan COBOL-kääntäjään. Tästä syystä COBOL-ohjelmat on kirjoitettu COBOL-ohjelmointiarkkiin, jotka käyttävät standardimuotoja.
Ohjelman kääntäjä, sen normaalin kääntämisen lisäksi, voi optimaalisesti tuottaa tulostetun kopion käännettävästä lähdekoodista. Kääntäjä jättää huomioimatta tunnistuskenttään kirjoitetun tekstin, mutta se tulee näkyviin lähdekoodiluettelossa (source code listing). (Roy & Dastidar. 1989, s.47-52) Esimerkkinä tästä on *-merkki ja kauttaviiva, joita käytetään kommentoimisessa. Kääntäjä jättää kommentit huomiotta.

COBOL on rakennettu vahvan idean pohjalta – avoin kieli, jota ei mikään organisaatio, henkilö eikä organisaatioryhmä omista. Se käyttää kielenään englantia keskustellakseen selkeästi ohjelmoijansa sekä seuraajien kanssa, ei keskustellakseen tietokoneen kanssa. (Coyle, Frank P. 2000)

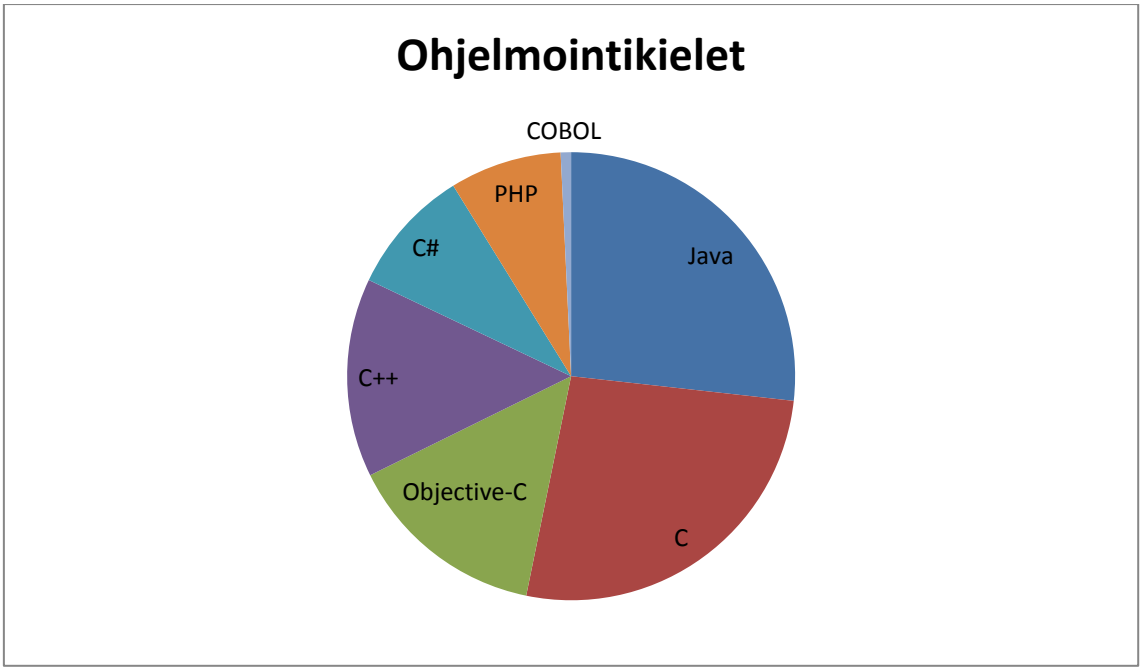
5.1.1 COBOL lauseet

Jokainen lause alkaa verbillä, kuten ”ADD”, joka osoittaa sen toiminnon, joka sen on määrä suorittaa lausetta ajettaessa. Usein lause lopetetaan pisteeseen, mutta se ei ole välttämätöntä. Kaksi tai useampi lause voidaan erottaa välilyönnillä tai pilkulla, jota seuraa välilyönti. COBOL suosii englanninkielistä ohjelmointia. (Roy & Dastidar. 2004) Esimerkkinä ”ADD COST TO TOTAL”, joka tarkoittaa että “COST:n” tämänhetkinen arvo lisätään “TOTAL:n” tämänhetkiseen arvoon.

5.1.2 Onko COBOL poistunut käytöstä kokonaan?

Kysymys ”Onko COBOL vielä olemassa?” saattaa kuulostaa harhautukselta. Todellisuudessa miljardeittain COBOL-koodia on vastuussa melkein kaikesta päivittäin käytetyistä asioista – laskuista koko yrityksen atk-pohjaiseen infrastruktuuriin saakka. (Coyle, Frank P. 2000) Suurelle osalle tämän päivän ohjelmoijista tätä kieltä ei ole edes olemassa. Harvoissa paikoissa opetetaan sitä ja esimerkiksi Java- ja C-ohjelmoijat jättävät sen huomioimatta.

Useimmat yritykset eivät enää käytä kyseistä kieltä uusien järjestelmien kehityksessä. COBOL-koodia muutetaan tai korvataan muilla ohjelmointikielillä kaikkialla. (Seacord, Robert C. 2003). Tänä päivänä käytetyistä ohjelmointikielistä, COBOL löytyy 30 käytetyimmän joukosta (TIOBE Software, 2013). Java on käytetyin ohjelmointikieli yli 18 % -osuudella, kun taas COBOL:n käyttöprosentti on 0,5 %. Alla oleva kuva esittää kuuden eniten käytetyn kielen sekä COBOL:n käyttöosuudet ohjelmistoissa vuonna 2013 (Kuva 8.).



Kuva 8. Ohjelmointikielien käyttö vuonna 2013

5.2 OHJELMOINTIKIELEN VALINTA

ASP:hen päädyttiin sen takia, että yrityksessä oli jo olemassa pohja samankaltaiselle työkalulle, ja sitä voitiin muokata tähän tarkoitukseen sopivaksi. Jotta kaikkea ei tarvitse tehdä uudestaan, työkalun rakentamiseen valittiin ASP. Myös se, että ASP-sovellusten yksi merkittävimmistä ominaisuuksista on tietokantapohjaiset sovellukset, kuten tietokantoihin tehtävät vaativat haut.

Lisäksi valintaan vaikutti ASP-sovelluksen hyödyllisyys jatkossa, sillä nyt alasajettava LÄJÄ on yksi harvoista yrityksen järjestelmistä, joka on ohjelmoitu COBOL:lla.

ASP-sovelluksia voidaan laajentaa myös palvelimen ulkopuolelle, joten laajat hajautetut sovellukset ovat mahdollisia, kuten esimerkiksi tietokantakyselyt satojen kilometrien päässä sijaitsevaan kantaan. ASP:llä pyritään saamaan nopeita perustietoja ja toimitusraportteja järjestelmistä. ASP-tiedostot on helposti muokattavissa ja niistä saa helposti uusia työkaluja. (Peltomäki, Juha. 2000, s.312-314)

Lisäksi ASP:n kanssa voi käyttää mitä tahansa kieltä, joka tukee ActiveX skriptiä. COBOL:sta yritetään, etenkin perinnejärjestelmissä, päästä eroon. COBOL-osaajia on vähän, joten sen käyttäminen työkalussa olisi jatkoa ajatellen vaikeaa. Jos tätä työkalua hyödynnettäisiin muissa alasajoissa, tulisi seuraavan tekijän opetella tämä vaikeahko kieli uudestaan. ASP on osoittautunut monipuolisemmaksi ja helpommaksi käyttää.

5.2.1 Active Server Pages

Active Server Pages (ASP) mahdollistaa tehokkaan web-sovellusten kehittämisen. Se on myös erittäin tehokas ActiveX- ja Java-komponenttien laajennettavuuksien kanssa. Tiedostoa ajettaessa selain pyytää ASP-tiedostoa, jolloin IIS (Internet Information Server) lähettää pyynnön ASP-moottorille. ASP-moottori lukee tiedoston rivi riviltä, jonka jälkeen se suorittaa tiedoston komennot. Lopuksi ASP-tiedosto palautetaan selaimelle puhtaana HTML:nä. (w3schools. 1993-2013)

ASP on palvelinpuolen ohjelmointiympäristö, jonka avulla voidaan luoda ja suorittaa dynaamisia, vuorovaikutteisia web-palvelin sovelluksia. Sen kanssa voi yhdistää HTML-sivuja, komentosarjoja ja COM-komponentteja luodakseen vuorovaikutteisia web-sivuja sekä web-pohjaisia sovelluksia, joita on helppo kehittää ja muokata. (Microsoft. 2013)

ASP on teknologia, jonka Microsoft alun perin kehitti ratkaisuna Common Gateway Interface (CGI) kanssa esiintyviin kehitysongelmiin. CGI tarjoaa mekanismin, jolla web-selain voi tehdä sovelluksen suorituspyynnön web-palvelimelle. Sovelluksen tulos on muunnettu selaimelle luettavaan muotoon (HTML) ja lähetetty pyynnön lähettäneelle selaimelle. Tämä mahdollistaa minkä tahansa skriptikielen käytön, VBSkriptistä (Visual Basic Scripting Edition) Pythoniin.(Weissinger, Keyton. 2000, s.xi)

ASP-tiedostot ovat skriptejä. Skriptit viittaavat tiedostoihin, jotka sisältävät sellaista ohjelmointikieltä, joka käännetään vasta sitten, kun niitä ajetaan. Tämä on erona toisiin ohjelmointikieliin, kuten Javaan ja C++:aan, joissa kone kääntää koodia erillisessä vaiheessa.
ASP-ohjelmat ovat ohjelmia, jotka ajetaan palvelimella selaimen pyynnön mukaisesti. Palvelimen ohjelmisto määrittelee, että tiedosto ei ole tavallinen HTML-tiedosto, vaan se on ASP-tiedosto. Tämä johtaa siihen, että palvelimen tukijärjestelmä tulkitsee tiedoston ASP-tiedostona. (Meyer, Jeanine. 06/2003)

5.2.2 ASP rakenne

ASP-skripti aloitetaan <% - merkillä ja lopetetaan %> - merkkiin.
Normaalisti ASP-tiedosto sisältää HTML-tagit, aivan kuten HTML-tiedostokin. ASP-tiedosto voi myös sisältää palvelin-skriptejä, jotka on ympäröity <% ja %> - merkeillä. Skriptit ajetaan palvelimella, ja ne voivat sisältää mitä tahansa määrityksiä (expressions, statements, procedures), jotka valittu skripti-kieli tunnistaa. (Peltomäki, Juha. 2000, s.312) ASP-skripti sisältää usein HTML-kieltä, jonka takia sivun perusrakenne perustuu HTML-sivun rakenteista. (Kuva 9.)

```
<html>
<body>
<%
response.write ("ASP osuus")
%>
</html>
</body>
```

Kuva 9. ASP-rakenne

ASP-osuus lisätään HTML-rakenteen sekaan aloittamalla ja lopettamalla ASP-lauseke merkeillä.

5.2.3 ASP teknologia

Tiedostot, jotka luodaan ASP:llä, saa päätteen .asp. ASP-tiedostolla on mahdollista aktivoida web-sivusto käyttäen mitä tahansa HTML-tiedostoja, skriptejä – kuten JavaScript ja Visual Basic – yhdistelmää ja komponentteja, jotka on kirjoitettu millä tahansa kielellä. Tämä tarkoittaa sitä, että ASP-tiedosto on yksinkertaisesti tiedosto, joka voi sisältää mitä tahansa HTML ja skripti yhdistelmiä. Kun ASP-tiedostoon tehdään muutoksia, sitä ei tarvitse kääntää, vaan riittää että muutokset tallennetaan. Tämä skripti käännetään automaattisesti seuraavan kerran kun web-sivu ladataan. Tämä on mahdollista sen takia, että ASP-teknologia on rakennettu suoraan Microsoft Web Servers:lle. ASP sisältää viisi standardi-objektia globaaliin käyttöön:

- Request – saadaan informaatiota käyttäjästä
- Response – lähetetään informaatiota käyttäjälle
- Server – internet-palvelimen hallinta
- Session – tallentaa tietoa ja muuttaa asetuksia käyttäjän sen hetkiseen web-palvelinistuntoon
- Application – jakaa sovellus-tason tietoja ja kontrolloi sovelluksen asetuksia sen eliniän ajan

ASP teknologia on koteloitu yhteen pieneen noin 300 KB Dynamic-link Libraryyn (DLL) nimeltä ASP.DLL. Tämä DLL on Internet Server Application Programming Interface (ISAPI) filtteri, joka on samassa muistissa kuin Internet Information Server (IIS). Heti kun käyttäjä pyytää tiedostoa, jonka tiedostopääte on .asp, tulee ASP ISAPI -filtteri käsittelemään tulkinnan. Tämän jälkeen ASP lataa tarvittavat DLL:t muistiin skriptin tulkitsemista varten, suorittaa minkä tahansa palvelin-puolen koodin, joka löytyy ASP:stä, ja siirtää saadun HTML-vastauksen web-palvelimelle. Tämä web-palvelin puolestaan lähettää HTML-vastauksen eteenpäin pyynnön esittäneelle selaimelle. (Weissinger, Keyton. 2000, s.6)

ASP-sivun käsittely voidaan jakaa kahdeksaan osaan:

- selain tekee pyynnön, jotta ASP-sivu ladattaisiin
- WWW-palvelin tunnistaa ASP-tarkenteen

- ASP-laajennus (ASP.DLL) käsittelee ASP-tiedoston
- mahdolliset lisädokumentit (SSI) lisätään tässä vaiheessa
- ohjelmat, jotka suoritetaan palvelinpäässä, käännetään määrätyllä skriptikielellä
- suoritetaan skripti
- luodaan HTML-muotoinen tiedosto
- toimitetaan luotu tiedosto selaimelle

5.2.4 Tietokantaan yhdistäminen ASP:llä

Tietokantaan yhdistäminen on hyvin yksinkertaista. Tietokantaan yhdistäminen voidaan tehdä erilliseksi omaksi tiedostoksi. ASP käyttää mekanismia ”include” osana kommenttia, ei lausetta, joka lisätään kaikkiin niihin tiedostoihin, jotka pitää yhdistää tietokantaan. Tiedosto, joka sisältää ”include” osan, on nimettävä ja sille annettaan pääte. Näiden tiedostojen päätteeksi voidaan laittaa .inc. Tämä pääte kuvaa sitä, että tiedosto ei ole käytettävissä sellaisenaan, vaan se on liitetty muihin tiedostoihin. Toinen vaihtoehto on laittaa tiedoston päätteeksi .asp. Tämä jälkimmäinen pääte estää sen, ettei tiedoston selaaja pääse tutkimaan lähdetiedostoa (Meyer, Jeanine. 06/2003). Tietoturvan kannalta tämä on hyvä asia, sillä kukaan ulkopuolinen ei saa käsiinsä mitään arkaluontoista tietoa.

5.3 TYÖKALUN TEKEMINEN JA TOIMINTA

Työkalua käytetään tiedon hakuun LÄJÄ:stä. Arkistoinnissa käytetään työkaluna käyttöliittymää, jossa on kaksi valintaa: jakeluterminaali ja vuosi. Vuosivalikosta löytyy vuodet 2005-2012 ja terminaalivalikossa kaikki jakeluterminaalit. Tarkoituksena on hakea arkistoon tositteet kahdeksan vuoden ajalta, jaoteltuna jakeluterminaaleittain sekä vuosittain. (Kuva 10.).

Kuva 10. Työkalun valikko

Työkalun tekeminen aloitettiin tarkastelemalla vanhoja ASP-tiedostoja, joita on käytetty muuhun tarkoitukseen. Apuna sieltä sai muun muassa tietokantaan yhdistämisen sekä jakeluterminaalivalikon. Jakeluterminaalivalikkoa apuna käyttäen saatiin vuosivalikolle vastaavanlainen alasvetovalikko. Ensimmäinen tiedosto sisältää koodin arkistointityökalun valikolle.

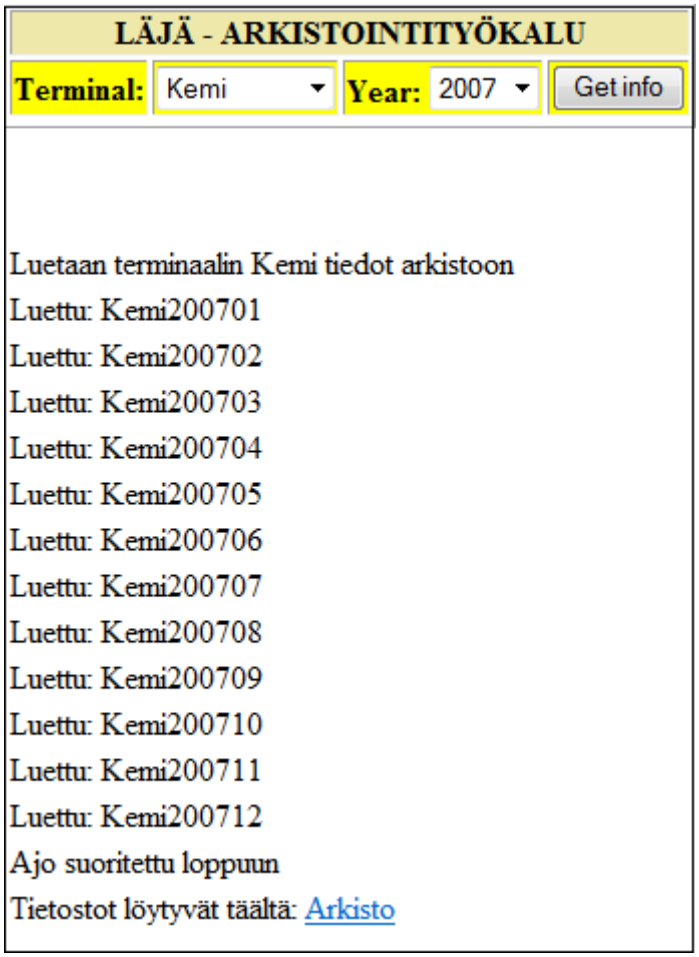
Toiseen tiedostoon sisällytetään tämä ensimmäinen tiedosto. Toisen tiedoston alussa otetaan yhteys LÄJÄ:ään. Arkistoon tuleva tiedosto luodaan seuraavaksi. Arkistointityökalusta valitaan haluttu jakeluterminaali sekä vuosi. Tämän jälkeen Get info- painike ajaa toisen ASP-tiedoston. Tämän jälkeen skripti luo tiedoston etukäteen määriteltyyn hakemistoon. Valitusta jakeluterminaalista ja vuodesta tulee tämän uuden tiedoston nimi ja tämä nimeämiskäytäntö helpottaa oikean tiedoston löytämistä myöhemmin. Tämä tiedosto tallennetaan muotoon .csv, jotta sen saa myöhemmin helposti auki Microsoft Excelillä. FileSystemObject:n avulla voidaan tiedoston luonnissa käyttää päätettä .csv, niin että ohjelma osaa luoda tiedoston automaattisesti oikeaan muotoon. Alla olevassa koodissa luodaan ensin tiedostolle nimi (Kuva 11.). Tämän jälkeen näytöllä näkyy minkä varaston tietoja ajetaan sillä hetkellä. Jos tämän nimistä tiedostoa ei vielä ole, niin tiedosto luo tämän tiedoston. Jos samanniminen on jo olemassa, tulee tämä poistamaan sen ja luomaan uuden.

```
'luodaan tiedosto terminaalin ja vuoden mukaan
    filename_w = "log/" & var & vuosi & ".csv"

    %>
    <tr><td>Luetaan terminaalin <% response.write Varasto_nimi %> tiedot arkistoon <td></tr>
    <%
    response.flush

    Set RS=Nothing
    Set FS=Nothing
    Set FS=Server.CreateObject("Scripting.FileSystemObject")

'tarkistaa onko tiedosto olemassa
    if FS.FileExists(Server.MapPath(filename_w)) then
        fs.DeleteFile(Server.MapPath(filename_w))
    end if

'luo uuden tiedoston
    Set RS=FS.CreateTextFile(Server.MapPath(filename_w), 3, false)

        Set rsCustomers = Server.CreateObject("ADODB.Recordset")


'hae kaudet 1-12
kausi = 1
Do while kausi < 13
    if kausi < 10 then
        pkausi = "0" & kausi
    else
    pkausi = kausi
    end if


    strSQLCustomers = "SELECT * FROM LAJAKU.TOSITE "
    strSQLCustomers = strSQLCustomers  & " WHERE OG_KAUSI = '" & vuosi & pkausi & "'"

        Set rsCustomers = Cnxn.Execute(strSQLCustomers)
```

Kuva 11. Koodinosa tiedoston luomisesta

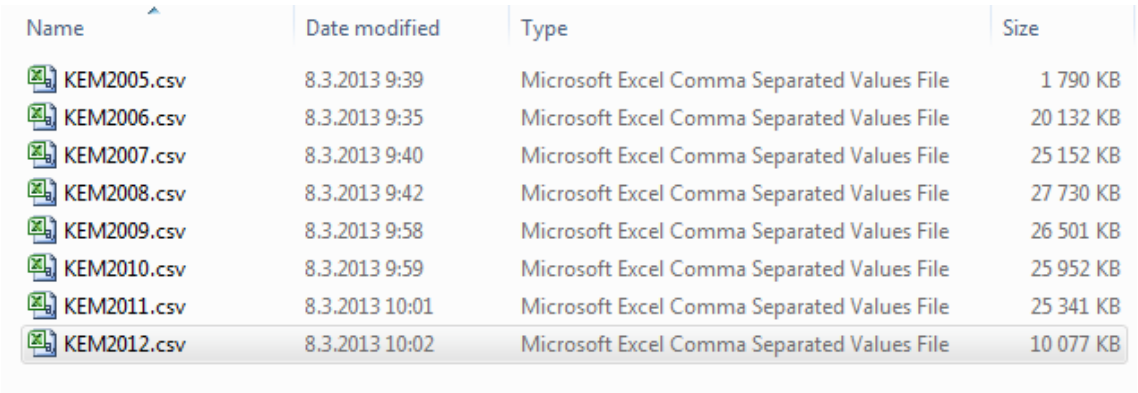
Kun tiedosto on luotu, ajetaan skriptissä SQL-lausekkeet. Nämä lausekkeet hakevat halutut tiedot alussa valitulta vuodelta ja jakeluterminaalilta. Jokainen vuosi on jaettu kuukausien mukaan kausiin, alkaen kaudesta 01 ja päättyen kauteen 12. Do while-loop ajetaan niin kauan, kunnes kaikki kaudet on kirjoitettu tiedostoon. Kaudet 1-9 sisältää etunollan, joten nolla lisätään if-lausekkeella kaikkiin alle kymmenen oleviin kausiin. Ennen do while-loopin päättymistä kauden numeroon lisätään yksi, jotta seuraava tiedon hakeminen kohdistuu seuraavaan kauteen. Näin käydään kaikki kaksitoista kautta läpi. Lopuksi on kaikki kymmenen jakeluterminaalia ja kaikki kahdeksan vuotta käyty läpi ja niistä tehty omat tiedostot. Tiedot ajettiin vain kerran, jos ajo oli onnistunut.

Ohjelmaa ajettaessa näytöllä näkyy, missä vaiheessa tiedoston luku on. Lopuksi ilmoituksena tulee, että tiedosto on luettu arkistoon. Alla olevassa kuvassa näkyy onnistunut ajo. Valintana oli Kemin terminaali sekä vuosi 2007. (Kuva 12.)



Kuva.12. Onnistunut ajo, tiedostot viety arkistoon.

Valmiit tiedostot menevät valittuun hakemistoon. Arkistokansiossa tiedot ovat jakeluterminaaleittain. (Kuva 13.)



Name	Date modified	Type	Size
KEM2005.csv	8.3.2013 9:39	Microsoft Excel Comma Separated Values File	1 790 KB
KEM2006.csv	8.3.2013 9:35	Microsoft Excel Comma Separated Values File	20 132 KB
KEM2007.csv	8.3.2013 9:40	Microsoft Excel Comma Separated Values File	25 152 KB
KEM2008.csv	8.3.2013 9:42	Microsoft Excel Comma Separated Values File	27 730 KB
KEM2009.csv	8.3.2013 9:58	Microsoft Excel Comma Separated Values File	26 501 KB
KEM2010.csv	8.3.2013 9:59	Microsoft Excel Comma Separated Values File	25 952 KB
KEM2011.csv	8.3.2013 10:01	Microsoft Excel Comma Separated Values File	25 341 KB
KEM2012.csv	8.3.2013 10:02	Microsoft Excel Comma Separated Values File	10 077 KB

Kuva.13. Kansiokuvaa arkistosta

6 JÄRJESTELMÄN ALASAJO

Liiketoiminnalta on tullut päätös ajaa alas lähettämöjärjestelmä. Vanhaan järjestelmään ei tule uutta tietoa, ja käytössä on jo kaksi korvaavaa järjestelmä. Tämä perinnejärjestelmä tuo turhia kustannuksia yritykselle muun muassa huoltosopimuksen ja lisenssien takia.

6.1 Toteutus, testaus ja arkistointi

Työn kulku:

- Tutustuminen lähettämöjärjestelmään sekä sen sisältöön ja käyttöön
- Tiedustelut laitteesta ja sen hävittämisestä
- Lakiasioiden tarkistus ja säilytysajan varmistus
- Siirrettävän datan valikointi
- Tietoturva-asioiden tarkistus
- Datan säilytyskohteen valinta
- Tallennustilan tilaus
- Työkalu
- Siirto
- Lisenssien irtisanominen
- Dokumentointi
- Laitteiston hävittäminen

Työ eteni suunnitelman mukaisesti. Tutustuminen järjestelmään ja sen käyttötarkoitukseen helpotti asioiden ymmärtämistä ja eteneminen oli helpompaa. Palavereiden järjestäminen oikeiden henkilöiden kanssa on tärkeätä. Haastattelemalla muun muassa yrityksen lakimiestä ja tietoturva-asiantuntijoita, pääsi asioiden kanssa eteenpäin. Lakimiehen kanssa käytiin läpi asiat, että mitä tietoa LÄJÄ:stä arkistoidaan ja sen perusteella saatiin selville, että kuinka kauan näitä tietoja pitää arkistoida. Muiden kanssa käytiin läpi LÄJÄ:n käyttöönottoprosessi sekä se, miten LÄJÄ on kehittynyt vuosien varrella. Siinä vaiheessa, kun kaikki tiedot oli saatu selville, oli mahdollista alkaa suunnitella työkalua, jolla siirretään halutut tiedot arkistoon. Työkalun valmistuttua, siirrytään testaukseen, jotta voidaan varmistaa tietojen oikeellisuus ja että kaikki tarvittavat tiedot ovat mukana.

Testauksessa käytetään ensin pientä osaa tiedoista, yhtä siirtovuotta, jotta syntynyttä tiedostoa voidaan tarkistella ja katsoa onko kaikki tarvittava tieto tullut mukaan ja oikeassa muodossa. Tämän jälkeen siirrytään kaikkien loppujen tietojen siirtämiseen vuosi kerrallaan.

Lopputuloksena kaikki tiedot vuosilta 2005-2012 on siirretty valittuun kansioon. Tiedostot on merkitty vuoden ja jakeluterminaalin mukaan, jotta myöhemmin tiedostojen tarkastelu olisi mahdollisimman helppoa. Kansiosta tiedostot siirretään tallennuslevylle, jossa ne säilytetään.

Dokumentti siirretään yrityksen sisäiseen hakemistoon. Arkisto-levylle ei kaikilla ole oikeuksia, mutta dokumentista löytyy tieto siitä, kenellä oikeudet on ja mistä niitä tarvittaessa voi pyytää.

Testauksessa jäi huomaamatta viimeisimpien sarakkeiden tiedot. Sarakkeita on yli 40, joista jokaista ei katsottu tarpeeksi tarkasti. Näissä sarakkeissa oli päivämäärä- ja kellonaikatiedot, jotka eivät siirrossa tulleet oikeassa muodossa. Tämä virhe huomattiin vasta siinä vaiheessa kun kaikki tiedot oli jo siirretty. ”FormatDateTime”-lisäys tarvittiin niiden kohtien eteen, jotka haluttiin ulos tässä muodossa. Virhe ei ollut suuri, mutta uudet tiedonsiirrot oli tehtävä. Tältä olisi vältytty, jos virhe olisi huomattu aikaisemmin testausvaiheessa.

Ongelmia ilmeni työn aikana hyvin vähän. Yhtenä ongelmana oli suurien tietomäärien siirtäminen arkistoon. ASP-skriptin ”timeout” on automaattisesti 90 sekuntia, mutta tämä aika ei riittänyt esimerkiksi Porvoon jalostamon tietojen siirtoon. Tämän ongelman sai helposti kierrettyä lisäämällä koodiin pienen määrityksen, jossa valitaan riittävä aika. Tämän korjauksen jälkeenkään Porvoon jalostamon tiedostojen ajot eivät menneet loppuun. Ajon lopuksi web-sivulla pitäisi lukea jakeluterminaalin nimi, vuosi ja kaudet sekä teksti, joka kertoo, että ajo on suoritettu loppuun. Kuitenkin osoittautui, että nämäkin tiedostot sisälsivät kaikki kaudet 1-12, mutta datamäärän oikeellisuudesta ei voinut olla varma, koska ajo ei mennyt loppuun. Ellei ajon päätteeksi tule ilmoitusta, että ajo on suoritettu loppuun, ei voi olla varma onko kaikki tiedot tallessa. Tähän ei ollut muuta ratkaisua, kuin ajaa skripti uudestaan. Porvoon osalta datamäärällisesti suurimmat vuodet jouduttiin jakamaan kahteen eri osaa, jotta ajo saatiin suoritettua loppuun.

6.2 Laitteiston hävittäminen

Järjestelmän alasajon viimeinen vaihe on laitteiston hävittäminen. Siinä vaiheessa, kun kaikki asiat alasajon suhteen on hoidettu, voidaan alkaa miettiä mitä laitteistolle tehdään. Päätöksenä on viedä laite romun kierrätykseen. Asianmukaisesta hävittämisestä vastaa Kuusakoski Oy, joka hoitaa laitteen metalliromun hävittämisestä.

7 LOPUKSI

Ennen työn aloittamista, minulla ei ollut ollenkaan tietoa perinnejärjestelmästä eikä myöskään siitä, että mitä vaiheita alasajo vaatii. Työn edetessä huomasin, että alasajo on yksi projekti muiden projektien joukossa ja työ aloitetaan tarvittaessa opiskelemalla aihetta ja siitä edetään asteittain. En ole aikaisemmin ollut mukana isoissa projekteissa, joten tämä oli todella opettavainen kokemus.

Lopputuloksena oli onnistunut alasajo ja arvokkaat tiedot turvallisesti arkistoituna. Asetetut tavoitteet saatiin täytettyä, sillä järjestelmä saatiin ajettua alas, tiedot saatiin arkistoon ja koko projekti dokumentoitua.

ASP-ohjelmointikielen käyttö oli hyvä valinta. Työkalun tekemiseen olisi jokin toinen ohjelmointikieli saattanut sopia paremmin, mutta ASP oli siinä mielessä järkevä valinta, sillä työpaikalla oli jo erilaisia esimerkkejä olemassa ja niitä pystyi käyttämään pohjana tälle työkalulle. ASP ei ollut minulle tuttu entuudestaan, mutta sen opetteleminen sujui hyvin.
Omasta mielestäni ASP sopi tähän työkaluun ihan hyvin. Voisin tulevaisuudessakin käyttää samaa kieltä vastaavanlaisissa työkaluissa.

Alasajo oli projektina suurempi ja hitaampi kuin osasin odottaa. Lähes 30 vuotta käytössä olleen järjestelmän alasajo ja ennen kaikkea tietojen arkistointi vaati suuren määrän haastatteluita ja tiedon keruuta jo ennen itse arkistoinnin ja alasajon aloittamista. Tämä työmäärä yllätti minut laajuudellaan. Oppimani perusteella, uskoisin että vastaavan projektin tekeminen onnistuisi minulta nyt itsenäisemmin, vaikka jonkin verran apua tarvitsisinkin järjestelmän käyttäjiltä ja muilta asiantuntijoilta – muun muassa yrityksen lakimiehiltä.

LÄHTEET

Coyle, Frank P. 2000, *Does COBOL exist?* [www].

Luettavissa:

http://ieeexplore.ieee.org.ezproxy.arcada.fi:2048/stamp/stamp.jsp?tp=&arnumber=839205

Haettu: 21.1.2013

Datpro. 2012 [www].
Luettavissa: http://www.datpro.fi/yritys/mitae-master-data-on
Haettu: 27.12.2012

Deloitte. 2012, *Counting the costs: decommissioning legacy, The Australian Banking Technology Report* [www]. Luettavissa: http://www.deloitte.com/assets/Dcom-Australia/Local%20Assets/Documents/Industries/Financial%20services/Banking%20and%20Securities/Deloitte_Australian_Banking_Technology_Report_2012_Counting_The_Cost.pdf
Haettu: 23.1.2013

EMC. 2013, *Data archiving* [www]. Luettavissa:
http://www.emc.com/corporate/glossary/data-archiving.htm
Haettu: 24.1.2013

Green, Bob. *The History of the HP 3000* [www]. Luettavissa:

http://www.robelle.com/library/smugbook/classic.html

Haettu: 17.1.2013

HP. 2009, *ALLBASE/SQL overview and features* [www]. Luettavissa:
http://www.hp.com/products1/evolution/e3000/mpeix/operating/allbase_sql.html
Haettu: 15.1.2013

Meyer, Jeanine. 06/2003, *ASP programming: Creating Database Web Applications with PHP and ASP* [www], Charles River Media / Cengage Learning, Herndon, VA, USA.

Luettavissa:

http://site.ebrary.com.ezproxy.arcada.fi:2048/lib/arcada/docDetail.action?docID=10061198&p00=asp%20programming

Haettu: 21.1.2013, 429 s.

Microsoft. 2013, *An ASP You Can Grasp* [www]. Luettavissa:
http://msdn.microsoft.com/en-us/library/ms972317.aspx
Haettu: 29.1.2013

Minisoft. 2013, *ODBC Driver for UNIX* [www]. Luettavissa:
http://www.minisoft.com/pages/middleware/odbc_unix/odbc32_unix.html
Haettu: 17.1.2013

Neste Oil vuosikertomus. 2011.
haettu: 20.12.2012

Omnidex. 2012, *Omnidex Overview* [www]. Luettavissa:
http://www.disc.com/omnidex/overview.html
Haettu:21.1.2013

Peltomäki, Juha & Inkinen, Ville & Rantala, Ari. 2000, *CGI- ja ASP-ohjelmointi*, Teknolit, Jyväskylä, 659s.

Robelle. a, *HP'S MPE Operating System* [www]. Luettavissa:
http://www.robelle.com/library/smugbook/mpe.html
Haettu: 17.1.2013

Robelle. b, *IMAGE/SQL: Reliable and Fast Database* [www]. Luettavissa:
http://www.robelle.com/library/smugbook/image.html
Haettu: 17.1.2013

Roy, M. K. & Dastidar, Gosh D. 1989, *COBOL Programming including MS-COBOL and COBOL-85*, toinen painos, Tata McGraw-Hill Education Pvt. Ltd., 498s.

Seacord, Robert C. & Plakosh Daniel & Lewis Grace A. 2003, *Modernizing legacy systems: software technologies, engineering processes and business practices*, Pearson Education, Inc., USA, 335 s.

TIOBE Software. 2013, *TIOBE Programming Community Index for March 2013* [www]. Luettavissa: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
Haettu: 5.4.2013

USDM. 2013, *ECM Legacy System Decommissioning* [www]. Luettavissa:

http://usdatamanagement.com/ecm-legacy-system-decommissioning.html

Haettu:23.1.2013

van den Heuvel W. J. 03/2007, *Aligning Modern Business Processes and Legacy Systems : A Component-Based Perspective* [www], MIT Press, Cambridge, MA, USA, 231 s. Luettavissa:
http://site.ebrary.com.ezproxy.arcada.fi:2048/lib/arcada/docDetail.action?docID=10173610&p00=retirement+of+legacy+system
Haettu: 27.12.2012

Weissinger, Keyton. 2000, ASP in a Nutshell, 2nd Edition, 494s.

Wikipedia. 2013a, *CSV* [www]. Luettavissa: http://fi.wikipedia.org/wiki/CSV
Haettu: 24.1.2013

Wikipedia. 2013b, *HP Multi-Programming Executive* [www]. Luettavissa:

http://en.wikipedia.org/wiki/HP_Multi-Programming_Executive

Haettu: 15.1.2013

w3schools. 1993-2013, *ASP Introduction* [www]. Luettavissa:

http://www.w3schools.com/asp/asp_intro.asp

Haettu: 21.1.2013