TAMPEREEN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

MICROSOFT DYNAMICS AX TOIMINNANOHJAUSJÄRJESTELMÄN VERSIONVAIHDON SUUNNITTELU

Asko Heikka

Tietojenkäsittelyn koulutusohjelma
Syyskuu 2008
Työn ohjaaja: Maritta Hoffrén

TAMPERE 2008

TAMPEREEN
AMMATTIKORKEAKOULU

Tekijä(t): Asko Heikka

Koulutusohjelma(t): Tietojenkäsittely

Opinnäytetyön nimi: Microsoft Dynamics AX –toiminnanohjausjärjestelmän versionvaihdon suunnittelu

Työn valmistumis-kuukausi ja -vuosi: Syyskuu 2008

Työn ohjaaja: Maritta Hoffrén **Sivumäärä:** 36

TIIVISTELMÄ

Opinnäytetyön toimeksiantajana toimii Mepco Oy. Yrityksessä ei ole ennen toteutettu Microsoft Dynamics AX -tuotteen versionvaihdosta, joten yrityksen kannalta oli tarpeellista tutkia koko päivitysprosessin kulku. Microsoft Dynamics AX -tuotteesta on myös vähän dokumentoitua materiaalia.

Opinnäytetyön tavoitteena oli selvittää Microsoft Dynamics AX -tuotteen versionvaihtoon kuuluvia toimia, kuten ylläpitoa, uudistamista ja testaamista. Tulosta käytetään erityisesti pohjana Microsoft Dynamics AX:n versionvaihtoprojektia suunniteltaessa, jotta versionvaihtoon kuuluvia työmääriä voitaisiin paremmin arvioida.

Opinnäytetyölle asetetut tavoitteet saavutettiin hyvin. Tuloksena saatiin kokonaiskuva prosessin laajuudesta, tarkempi kuva prosessiin liittyvistä tehtävistä ja tietoa siitä, mihin asioihin prosessin eri vaiheissa tulee kiinnittää huomiota.

Työn tuloksia tullaan hyödyntämään uuden Microsoft Dynamics AX 4.0 -järjestelmän varsinaisessa tuotantoonotossa, joka tapahtuu Mepco Oy:ssä vuoden 2008 aikana, sekä mahdollisesti asiakkaille tehtävissä ympäristön versionvaihtoprojekteissa.

Avainsanat: Toiminnanohjausjärjestelmä Ylläpito Uudistaminen

TAMPEREEN AMMATTIKORKEAKOULU
University of Applied Sciences

Author(s):	Asko Heikka
Study programme(s):	Business Information Systems
Title of the thesis	Planning the Version Upgrade of Microsoft Dynamics AX Enterprise Resource Planning System
Month and year of completion:	September 2008
Supervisor:	Maritta Hoffrén

Number of pages: 36

ABSTRACT

The client of this thesis was Mepco Ltd. The whole Microsoft Dynamics AX product version update had not been carried out at the company before, so the company felt it was necessary to examine the entire upgrade process. Microsoft Dynamics AX product is also fairly low-documented.

The aim of this thesis was to identify the version upgrade functions of the Microsoft Dynamics AX product, such as maintenance, re-engineering and testing. The results of this thesis are used as a basis for planning the Microsoft Dynamics AX's version upgrade and in particular how the workload could be better assessed.

The objectives of the thesis were achieved very well. The result was an overall picture of the process, and information about what stage of the process needs to be given special attention to, as well as more an accurate picture of the process-related tasks.

The results of this work will be used with the new 4.0 system, which will take implemented in Mepco Ltd. in 2008. The results of this work could also be used in the environment upgrade projects for the customers of the company.

Key words: Enterprise recourse planning maintenance upgrade

Sisällysluettelo:

1 Johdanto

Olen tehnyt tämän opinnäytetyön toimeksiantona Mepco Oy:lle (ks. www.mepco.fi), jossa ei ole aikaisemmin toteutettu Microsoft Dynamics AX -tuotteeseen kohdistuvaa versionvaihdosta. Yritys on operatiivisiin yritysohjelmistoihin keskittynyt tietotekniikkayhtiö, joka kehittää, toimittaa ja integroi kokonaisvaltaisia ratkaisuja asiakkuudenhallintaan, toiminnanohjaukseen sekä palkka- ja henkilöstöhallintoon. Mepco Oy toimii Microsoft Gold Certified -partnerina, joten yrityksen toiminta perustuu kolmitasoiseen yhteistyömalliin, jonka osapuolia ovat Mepco, Mepcon asiakkaat ja Microsoft.

Olen työskennellyt yrityksessä noin puolitoista vuotta ja toimin sovelluskonsulttina, vastaan enimmäkseen Microsoft Dynamics AX ohjelmiston kehittämisestä sekä sen ylläpidosta. Tuote kuuluu yrityksen perustuotteisiin, joita se jälleenmyy sekä tarjoaa tuotteen räätälöintiä asiakkaan tarpeita paremmin palvelevaksi kokonaisuudeksi. Noin 1,5 vuoden aikana olen saanut varsin kattavan kuvan tuotteesta. Yrityksellä on omassa käytössään tuotteesta 3.0 versio. Uusi 4.0 versio on tarkoitus ottaa Mepcon omaan tuotantokäyttöön kesäkuussa 2008, jolloin koko päivitysprosessin kulusta tulee olla selkeä kuva, jotta versionvaihdos on mahdollista toteuttaa sujuvasti.

Työn tavoitteena on kartoittaa Microsoft Dynamics AX -tuotteen versionvaihtoon kuuluvia töitä. Tulosta käytetään pohjana Microsoft Dynamics AX:n versionvaihtoprojektia suunniteltaessa, erityisesti, jotta versionvaihtoon kuuluvia työmääriä voitaisiin paremmin arvioida. Myöhemmässä vaiheessa tätä työtä apuna käyttäen voidaan myös laatia selkeä ohje koko prosessin kulusta.

Microsoft Dynamics AX -tuotteen versionvaihdokseen liittyy vahvasti ohjelmien ylläpito ja uudistaminen, koska versionvaihdosprosessiin liittyy vanhan koodin uudistamista sekä asiakkaille että omaan käyttöön räätälöityjen toiminnallisuuksien uudelleen rakentamista. Versionvaihdoksen jälkeen tuote vaatii luonnollisesti ylläpidollisia toimia, kuten vanhan ohjelmakoodin muokkaamista sekä mahdollisesti uusien toiminnallisuuksien toteuttamista. Työssä tarkastellaankin ohjelmien uudistamista ja ylläpitoa yleisellä tasolla sekä tutkitaan niihin liittyviä ongelmia ja ratkaisuja. Tavoitteena on myös perehtyä testaukseen, koska se liittyy vahvasti ylläpito- ja uudistamisprosesseihin. Työssä käydään läpi myös Microsoft Dynamics AX -tuotteen versionvaihdokseen 3.0:sta 4:aan liittyviä keskeisiä toimintoja.

Versionvaihtoprosessi edellyttää näiden asioiden tutkimista, jotta koko prosessi saataisiin toteutettua mahdollisimman sujuvasti ja

osattaisiin paneutua prosessin vaiheisiin tarkemmin, sekä varautua sen aikana ja sen jälkeen mahdollisesti ilmeneviin ongelmiin.

Tutkimusaihetta koskevan lähdeaineiston kerääminen ei ollut kovin yksinkertaista. Tämä johtuu siitä, että suurta osaa kirjallisuudesta ei ollut lainattavissa tai muuten hankittavissa tiedossa olleista lähteistä. Suurin osa kerätystä aineistosta olikin englanninkielistä, mikä osaltaan lisäsi työn haasteita.

Tutkimuksen esisijaisena aineistona on käytetty Maarit Harsun Ohjelmien ylläpito ja uudistaminen -teosta (Harsu 2003), joka tätä työtä kirjoitettaessa osoittautui kattavimmaksi opinnäytetyön aihetta käsitteleväksi yleisteokseksi. Työssä käytetty lähdeaineisto on koottu työn lopussa olevaan lähdeluetteloon.

Työssä edetään siten, että alkuosassa kerrotaan yleistä asiaa toiminnanohjausjärjestelmistä sekä Microsoft Dynamics AX -tuotteesta, jonka jälkeen käsitellään ohjelmistojen ylläpitoon sekä uudistamiseen liittyviä yleisiä asioita ja ongelmia, sekä testaukseen liittyviä yleisiä asioita. Työn loppuosassa kuvataan Microsoft Dynamics AX -tuotteen versionvaihdon yhteydessä tehtävät toimet sekä esitetään yhteenveto aiheesta.

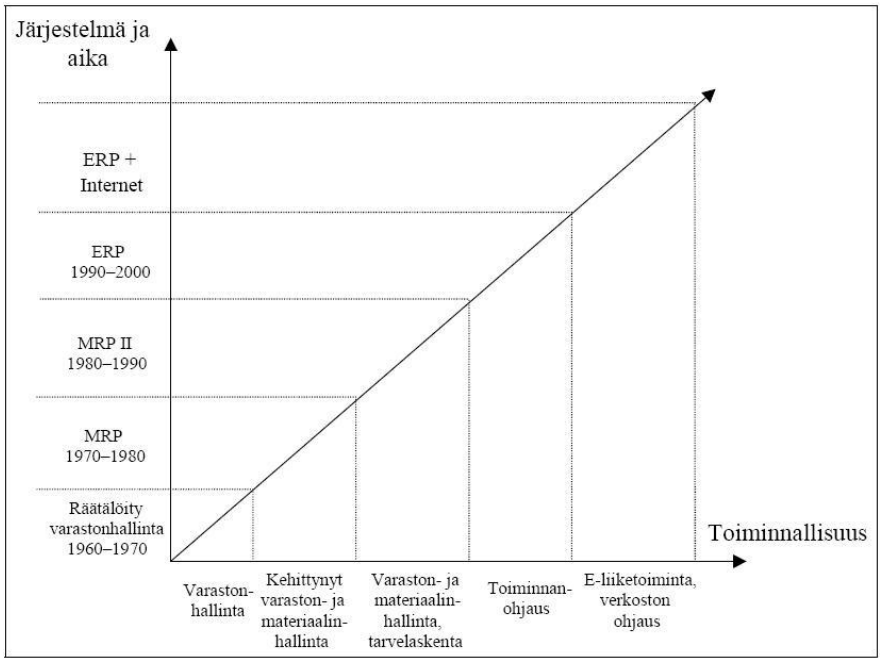
2 Toiminnanohjausjärjestelmät

2.1 Yleistä toiminnanohjausjärjestelmistä

ERP-järjestelmien (Enterprise Resource Planning, ERP) kehityksen voidaan katsoa alkaneen 1960-luvulla, jolloin varastonseurantaan aloitettiin kehittämään ohjelmistoja. Ohjelmistot olivat nykymittapuun mukaan melko yksinkertaisia. (Kalliokoski & Simons 2001: 46-49)

1970-luvulla materiaalitoimintojen suunnittelua ja hallintaa helpottamaan kehitettyjä MRP-järjestelmiä sekä 1980-luvulla valmistuksen suunnitteluun kehitettyjä MRP II-järjestelmiä (Manufacturing Recourse Planning, MRP II) voidaankin pitää pääasiallisena ERP-kehitystyön lähtökohtana. (Kalliokoski & Simons 2001: 46-49)

MRP-ohjelmistojen käyttö oli melko kankeaa ja ne olivat aika vaatimattomia verrattuna nykypäivän kehittyneisiin ERP-järjestelmiin. ERP-järjestelmät käyttävät uudempaa teknologiaa tuottaakseen kokonaisen ohjelmiston, jonka avulla voidaan hallita ja automatisoida yrityksen kaikkia prosesseja. Kuvassa 1 on havainnollistettu miten ohjelmistojen ominaisuudet ja tuotesukupolvet liittyvät toisiinsa. (Kalliokoski & Simons 2001: 46)



Kuva1: ERP-ohjelmistojen ominaisuudet ja tuotesukupolvi

ERP-järjestelmät ovat viime vuosina yleistyneet sekä suurissa että pk-yrityksissä. (Kalliokoski & Simons 2001: 48) ERP-järjestelmän tarkoituksena on yhdistää kaikki tietohallinnon piiriin kuuluva tieto yhdeksi hallittavaksi kokonaisuudeksi kuten myynnin, henkilöstöhallinnon, kirjanpidon, projektinhallinnan, tuotannonohjauksen ja tuotesuunnittelun tiedot. Järjestelmän avulla on muun muassa mahdollista tuottaa talousraportteja sekä organisoida tuotehinnoittelua, joiden avulla voidaan hallita henkilöstö-, materiaali- ja rahallisia resursseja.
(Markus&Tanis&van Fenema 2000: 26)

ERP-järjestelmä ei tarjoa kuitenkaan suoraa ratkaisua yrityksen ongelmiin, vaan sen avulla voidaan helpottaa yritystoiminnan eri osa-alueiden hallintaa. Järjestelmän avulla voidaan tehostaa yrityksen toimintaa kuten asiakaspalvelua, säästämällä samalla kustannuksia ja investointeja. (CNerps stydy : 19)

Nykypäivän markkinoilla olevat toiminnanohjausjärjestelmät perustuvat client-server -arkkitehtuuriin, mikä tarkoittaa, että yrityksellä on käytössä yritystason palvelin ja tarvittava määrä työasemia. Modulaarisen toiminnanohjausjärjestelmän komponentit kommunikoivat suoraan keskenään tai tekevät päivityksiä yhteiseen, keskitettyyn tietokantaan. Järjestelmät on tarkoitettu pääsääntöisesti yrityksen sisäisen toiminnan ohjaukseen, mutta yhä enenevässä määrin on mukana myös välineitä yritysten välistä tiedonsiirtoa varten, etenkin laskutuksessa ja tilausten käsittelyssä. (Kalliokoski & Simons 2001: 48-49)

ERP-järjestelmän sisältämät tiedot sijaitsevat keskitetyssä tietokannassa, joita käsitellään reaaliajassa, joten tieto on jatkuvasti ajan tasalla. (O'Leary 2002: 27)

2.2 Käsitteitä

ERP-järjestelmät sisältävät paljon omia käsitteitä, jotka tulevat tutuiksi yleensä vasta ohjelman käytön yhteydessä. Jokainen ohjelmistotoimittaja käyttää ohjelmassaan eri nimityksiä ohjelmaan liittyvistä osa-alueista. Käytännössä kuitenkin lähes kaikki ohjelmistot sisältävät kuitenkin samat komponentit.

2.2.1 Moduuli

ERP-järjestelmät koostuvat tyypillisesti laajasta määrästä erilaisia toimintoja, joita kutsutaan usein moduuleiksi (osa-alueiksi). Eri moduulit kattavat eri osa-alueita yrityksen toiminnasta. Jokaisella toimittajalla on omat variaationsa moduuleista. Esimerkkejä moduuleista ovat kirjanpito, projektin hallinta, varastonhallinta, osto, myynti ja henkilöstöhallinto. ERP-järjestelmä integroi organisaation osat tiiviisti yhteen.
(http://www.sysoptima.com/erp/erp_modules.php)

2.2.2 Partneri

ERP-ohjelmistotalot eivät itse jälleenmyy tuotettaan, vaan markkinointi tapahtuu partnereiden eli kumppaneiden kautta. Yleensä partnerit ovat suuria tai suurehkoja ohjelmistotaloja, jotka ovat erikoistuneet tiettyyn alueeseen ERP-järjestelmässä.

Partnerit voivat räätälöidä markkinoimaansa ohjelmistotuotetta vastaamaan paremmin asiakkaan tarpeita tai lisätä ohjelmistoon tietyn maan ominaispiirteitä.

2.3 Hyödyt yritykselle

Yrityksen tietojen hallinnassa ja toiminnanohjauksessa tietojärjestelmien asema on kasvanut yhä tärkeämmäksi. ERP-järjestelmien pohjana on tietojenkäsittelyn ja toiminnanohjauksen yhdistäminen. (Saari&Oijennus 2004: 11)

ERP -järjestelmä on koko yrityksen toiminnan laajuinen työkalupakki, jolla voidaan ennustaa, suunnitella ja järjestää yrityksen toimintoja. ERP:n avulla voidaan yhdistää asiakkaat ja toimittajat yhdeksi toimitusketjuksi yrityksen toiminnassa. Järjestelmässä olevat hyvät toimintatavat auttavat yritysjohtoa päätöksenteossa. Järjestelmän avulla voidaan koordinoida myyntiä, markkinointia, yrityksen eri operaatioita, logistiikkaa, ostoja, taloutta, tuotekehitystä ja henkilöstöhallintoa. (Wallace&Kremzar 2001: 10 - 12)

ERP -järjestelmän yksi tärkeimmistä eduista on mahdollisuus jakaa tietoa useamman käyttäjän kesken. Järjestelmään tarvitsee vain kerran laittaa tarvittava tieto ja sitä voidaan sen jälkeen hyödyntää yrityksen eri osa-alueissa. Näin tieto ei korruptoidu useiden syöttövaiheiden tai käyttäjien myötä. (Bishop&Lucas 2005: 51)

3 Microsoft Dynamics AX

3.1 Historia

Microsoft Dynamics AX, tunnettiin aikaisemmin nimellä Microsoft Axapta. Tuotteen alkuperäinen kehittäjä on tanskalainen ohjelmistotalo nimeltä Damgaard, joka yhdistyi Navision ohjelmistotaloon vuonna 2000. Microsoft osti Navisionin vuonna 2002. Suurin osa Microsoft Dynamics AX:n toiminnoista on siis kehitetty Tanskassa. Dynamics AX:n ja Microsoftin muiden tuotteiden integraatio on toteutettu yhtiön pääpaikassa Redmondissa, Washingtonissa. (http://www.itworld.com/App/670/060612dynamicsax4/)

3.2 Käsitteitä

Microsoft Dynamics AX sisältää joukon omia käsitteitään. Seuraavassa on kerrottu tähän opinnäytetyöhön liittyviä tärkeimpiä käsitteitä.

3.2.1 Tasot

Microsoft Dynamics AX -tekniikka perustuu tasoihin, joiden ansiosta järjestelmän rakentaminen ja ylläpito on helppoa.

Jokainen taso sisältää omat lähdekoodit. Ylemmän tason lähdekoodi on kopio alemman tason lähdekoodista. Alemman tason lähdekoodi voidaan siis ylikirjoittaa ylemmäntason lähdekoodilla, jolloin korkeammalla oleva taso korvaa kaikkien alatasojen elementit. Esimerkiksi, jos lisään napin ikkunaan, niin koko ikkuna kopioituu käytössä olevalle tasolle. Jos taas tuhoat koko elementin käytössä olevalta tasolta, käytetään alemman tason elementtiä. Tasot ovatkin käytännössä lähdekoodin hierarkkinen rakenne.

Dynamics Ax sisältää kahdeksan eri tasoa (Kuva 2). Neljä alimmaista tasoa (SYS, GLS, DIS, LOS) kuuluvat standardipakettiin, joita partnerit tai asiakkaat eivät voi mukauttaa. Neljää ylimmäistä tasoa (BUS, VAR, CUS, USR) taas voidaan mukauttaa ilman, että muutokset vaikuttavat muiden tasojen ominaisuuksiin. Näin ollen järjestelmää voi huoletta mukauttaa ja päivittää ajan myötä häiritsemättä yritystoimintaa tai aiheuttamatta ylimääräisiä kustannuksia.
(http://www.microsoft.com/finland/dynamics/ax/default.Microsoft.mspx)

USR	Korkein taso. Käyttäjät voivat luoda omia määrityksiä kuten raportteja.
CUS	Asiakas voi luoda omia muutoksia tämän tason avulla
VAR	Partnerit käyttävät tätä tasoa. Muutokset ovat asiakaskohtaisia. Vaatii lisenssin
BUS	Alin taso johon partnereilla on pääsy. Partnerit voivat käyttää tasoa omia moduuleita varten. Vaatii lisenssin
LOS	Käytetään paikallisiin muutoksiin moduuleille jotka eivät ole globaalisti sertifioituja.
DIS	Maakohtaisiin lokalisaatioihin käytettävä taso. Microsoft käyttää tätä tasoa myös 4.0:n hotfixien toimittamiseen.
GLS	Microsoftin alihankkijoiden käytössä.
SYS	Alin taso, jossa sijaitsee ohjelman perustoiminnallisuus

Kuva 2: Microsoft Dynamics AX tasot hierarkkisessa järjestyksessä

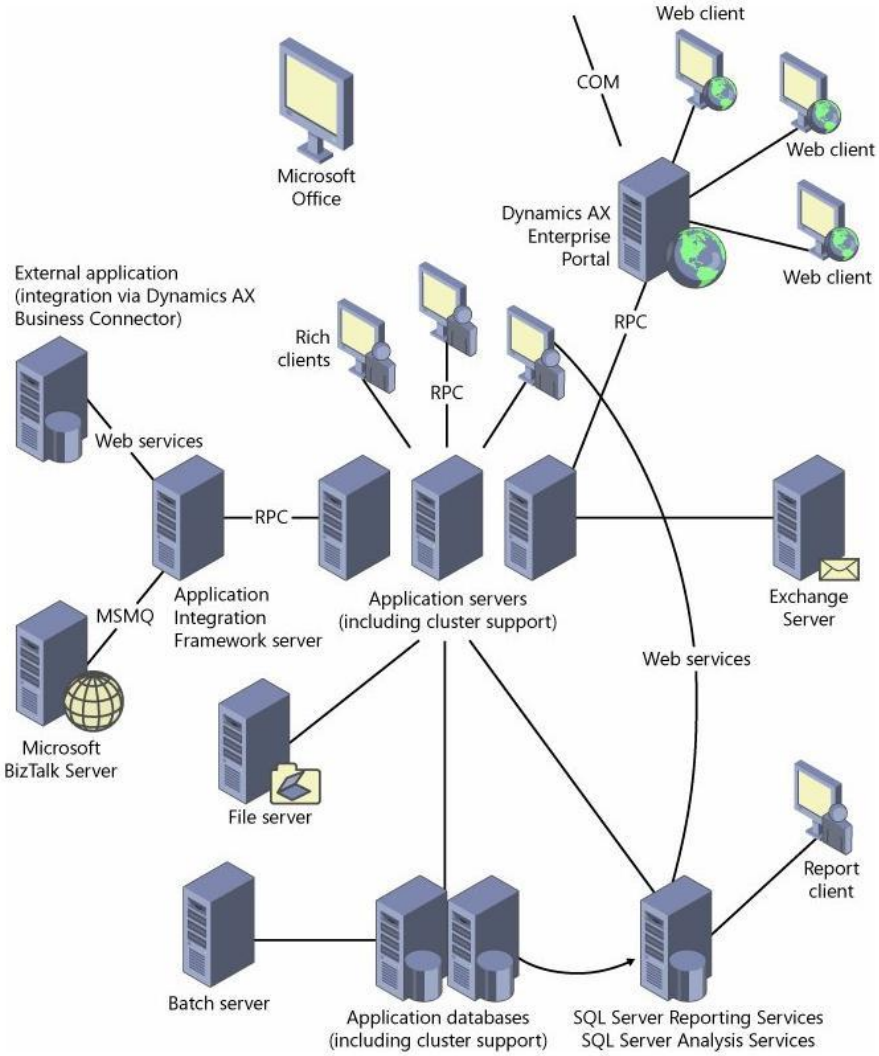
3.2.2 MorphX

Microsoft Dynamics AX:n kehitysympäristöä kutsutaan nimellä MorphX. MorphX on integroitu graafinen kehitysympäristö, joka koostuu AOT:sta (Application Object Tree) ja muutamista muista työkaluista. (MorphX IT: 19) Sen avulla kehittäjät voivat luoda omia datatyyppejä, tauluja, kyselyjä, näkymiä, valikoita ja raportteja. MorphX tukee vedä ja pudota tekniikkaa ja on hyvin intuitiivinen. MorphX sisältää oman koodieditorin, jonka avulla kehittäjät voivat muokata kaikkia ohjelmaan liittyviä luokkia

MorphX-ympäristössä tehdyn kehitystyön muokkaukset ja yhdistelyt vaikuttavat koko järjestelmään ja tulevat näkyviin välittömästi.

3.3 Ohjelmiston rakenne ja toiminta

Microsoft Dynamics AX 4.0 on 3-tasoinen asiakas-palvelin (client-server) ohjelmisto. Tyypillisesti se levittäytyy toimintaympäristöiksi, jotka ovat konfiguroitu parhaiten vastaamaan asiakkaan tarpeita. Kuvassa 3 on esimerkki tapaus Microsoft Dynamics 4.0 toimintaympäristöstä, johon kuuluvat muun muassa tiedostopalvelin, sovelluspalvelin, asiakkaat, tietokantapalvelimet ym.



Kuva 3: Esimerkki Microsoft Dynamics AX toimintaympäristöstä

Dynamics AX Application Object Server (AOS), voidaan sijoittaa yhdelle palvelimelle, mutta se voidaan myös skaalata useammallekin palvelimelle, riippuen käyttäjälukumäärästä. Palvelu voi kommunikoida joko yhden tietokannan tai klusteroidun tietokannan kanssa, mikäli uhkana on, että tietokannasta tulee järjestelmän ongelmakohta.

Dynamics AX:n raskaat asiakkaat(client) kommunikoivat AOS palvelun kanssa käyttäen Microsoft RPC-teknologiaa.

4 Ylläpito, uudistaminen ja testaus

Tässä luvussa käsittelen ohjelmien ylläpitoa, uudistamista sekä testausta. Testaukseen kuluva aika ja kustannukset pitäisi pystyä optimoimaan. Hyvä suunnittelu ja toteutus vähentävät testaukseen käytettävän ajan tarvetta.

4.1 Ohjelmien ylläpito

Ohjelmien ylläpito on perinteisesti määritelty muutosten teoksi ohjelmistoon sen jälkeen, kun se on luovutettu asiakkaille. (http://users.jyu.fi/~koskinen/lectio.htm) Ohjelmien ylläpito on merkittävä osa tietojärjestelmien kehittämisestä. Ian Sommervillen (Software Engineering 1996) mukaan on arvioitu, että ylläpitoon kuluu n. 65-75% ohjelmiston elinkaaren aikaisista kustannuksista.

Ohjelmien ylläpito on myös vaativaa työtä. Vaativuutta lisääviä tekijöitä ovat mm.: ylläpidettävien ohjelmistojen ja tehtävien muutosten laajuus, muutosten tekeminen koodiin, joka ei ole muutoksien tekijälle ennestään tuttua, sekä puutteellinen dokumentointi. (http://users.jyu.fi/~koskinen/lectio.htm)

Karkeasti jaoteltuna ylläpito voi olla joko vanhan toiminnallisuuden korjaamista tai uuden lisäämistä. Ennen korjauksen tai lisäyksen toteuttamista on ohjelmakoodista kuitenkin löydettävä sopiva kohta, johon muutos toteutetaan. (Harsu 2003: 77)

Erityisen tärkeää ylläpidossa on vaikutusten huomioonottaminen, joita koodiin tehdyt muutokset aiheuttavat. Yksinkertaisetkin muutokset voivat aiheuttaa muutostarvetta myös muualla ohjelmakoodissa. Vaikutuksia ei kuitenkaan aina heti huomata. Hyvin tavallista on, että tietyn virheen korjaaminen aiheuttaa uusia virheitä, jotka huomataan vasta myöhemmin. Näiden uusien virheiden korjaaminen aiheuttaa taas lisää uusia virheitä. Tällaista aaltomaista muutostarpeiden ilmenemistä kutsutaan väreilyvaikutukseksi (ripple efect). (Harsu 2003: 77)

4.1.1 Ylläpitoon liittyviä ongelmia

Ylläpito on vaikeampaa kuin uuden koodin kirjoittaminen. Ylläpitoon kuuluu osittain samanlaisia toimintoja kuin varsinaiseen ohjelmointiin. Ylläpitoon kuuluva uusien toimintojen lisääminen muistuttaa hyvin paljon uuden ohjelman kirjoittamista. Ohjelmien ylläpitäjät joutuvat lisäksi tarkastelemaan toisten kirjoittamaan koodia, joka voi olla vaikea ymmärtää.(Harsu 2003: 78)

Ylläpidollisia ongelmia ovat muun muassa koodin huono rakenne, ylläpitäjien tiedot järjestelmästä voivat olla riittämättömät sekä dokumentit voivat olla huonoja tai puuttua kokonaan. (Harsu 2003: 79)

Ongelmia aiheuttaa myös se, että ylläpidollisia tehtäviä pidetään yleensä ikävinä, eikä niin houkuttelevina ja palkitsevina, kuin varsinaista ohjelmistokehitystä. Ylläpitäjät voivat joutuvat tekemisiin tyytymättömien asiakkaiden kanssa ja lisäksi he joutuvat tarkastelemaan toisten kirjoittamaa lähdekoodia, joka voi olla usein huonorakenteista. (Harsu 2003: 79)

Ylläpitotehtäviin laitetaan usein kokemattomat ohjelmoijat, vaikka tilanteen pitäisi olla juuri päinvastoin, sillä ylläpitotehtävissä tarvitaan hyvin monenlaisia ja laaja-alaisia taitoja. Virheiden etsinnässä ja niiden korjaamiseksi vaaditaan usein hyvää tietoa koko järjestelmän toiminnasta, ohjelmien tarkastelusta ja mahdollisten työkalujen toiminnasta. (Harsu 2003: 79)

Ylläpito voi olla uusien toimintojen lisäämistä, jolloin käydään kaikki elinkaaren vaiheet läpi uuden toiminnan määrittelystä testaukseen. Kaikissa ylläpitotoiminnoissa tarvitaan tietoja testaamisesta, koska tehtyjen muutosten jälkeen täytyy tarkistaa, että järjestelmä toimii oikein ja kuten on tarkoitettu. Näiden syiden perusteella ylläpitäjiksi tulisikin kelpuuttaa vain kokeneimmat ja parhaimmat ohjelmoijat. (Harsu 2003: 79)

Järjestelmien ylläpitäjien kohtaamia haasteita: (Vuorenpää 2007, 22)

- vaikeudet vastata nopeisiin muutoksiin liiketoimintavaatimuksissa,
- vaikeudet vastata koko toimialaa koskeviin tai lainsäädännön puolelta tuleviin vaatimuksiin,
- asiakkailta tulleiden aloitteiden toteuttamisriskit, jotka johtuvat datan redundanssista, integroituvuuden puutteesta ja epäjohdonmukaisuudesta,
- kyvyttömyys vastata sähköisen liiketoiminnan puolelta tuleviin vaatimuksiin yhtä monipuolisesti kuin perinteisessä järjestelmässä on totuttu,
- puutteet tiedon jäljitettävyydessä useampaan toimintoon tai liiketoiminta-alueeseen,

- vaikeudet saada osaavaa henkilöstöä ylläpitämään vanhempia kieliä, tietokantoja ja ympäristöjä,

- perinteisten järjestelmien ja laitteistojen leasing- ja ylläpitokustannukset,

- tyytymättömyys ulkoistamiseen, sovellusvuokraukseen tai muuhun kolmansien osapuolien kanssa tehtyyn työhön ja

- liiketoimintaan aiheutuneet virheet, jotka parhaimmillaan antavat kuvan osaamattomuudesta, pahimmillaan aiheuttavat tappioita tuottojen menetyksinä tai kustannusten lisääntymisinä.

4.1.2 Ratkaisuja ylläpidon ongelmiin

Ylläpitoon liittyviin ongelmiin on olemassa erilaisia ratkaisuja, joiden avulla voidaan vähentää myös ylläpitokustannuksia. Kun ylläpitoa tarkastellaan osa-alueittain, voidaan löytää seuraavanlaisia tapoja säästää ylläpitokustannuksissa. (Harsu 2003: 81)

- Korjavaan ylläpidon kustannuksia vähennetään parantamalla koodin laatua, kehittämällä testausohjelmia, parantamalla ja yhtenäistämällä dokumentteja sekä noudattamalla vakiintuneita käytäntöjä ja standardeja ohjelmoinnissa ja dokumentoinnissa.

- Täydellistävän ylläpidon kustannuksia voidaan vähentää ottamalla käyttäjät paremmin huomioon. Tämä voi tapahtua prototyyppitekniikoita käyttämällä ja käyttäjien intensiivisemmällä osallistumisella analyysi- ja suunnitteluvaiheisiin.

- Mukauttavaa ylläpitoa tehostetaan ennakoimalla tulevia muutoksia etukäteen, eli jo määrittely- ja suunnitteluvaiheissa. Olioparadigmaa sovellettaessa ohjelmista saadaan yleisempiä ja useampiin tilanteisiin sopivia käyttämällä perimistä ja geneerisiä tyyppejä.

- Ylläpitotehtäviä vähennetään vähentämällä koodia, mikä voidaan toteuttaa uudelleenkäyttämällä koodia.

Kaikki edellä esitetyt ratkaisut liittyvät varsinaiseen ohjelmistokehitykseen. Ohjelmien ylläpidettävyyttä voidaan siis parantaa kiinnittämällä huomiota näihin seikkoihin varsinaisessa ohjelmistokehitysprosessissa. On kuitenkin selvää ettei kaikkeen pystytä varautumaan varsinaisen ohjelmistokehityksen aikana, koska evoluutiolakien mukaisesti ohjelmien rakenne heikkenee toistuvien muutosten seurauksena.
(Harsu 2003: 81)

Ylläpidollisien ongelmien ratkaisemiseksi voidaan löytää kaksi lähestymistapaa. Ylläpidettävyyttä voidaan ensinnäkin parantaa takaisinmallinnuksen, uudelleenrakentamisen ja uudistamisen kautta. Toiseksi ylläpidettävyyttä voidaan parantaa myös hallinnollisilla toimilla, kuten ottamalla käyttöön yhteneväiset tavat suunnittelussa, ohjelmoinnissa ja dokumentoinnissa. (Harsu 2003: 81)

4.2 Ohjelmien uudistaminen

Ohjelmien uudistaminen on laaja-alaisempaa ja suunnitelmallisempaa toimintaa kuin ylläpito. Siksi uudistaminen hoidetaankin usein erityisenä uudistamisprojektina, jolloin se vaiheineen muistuttaa ohjelmistoprojektia. Kun uudistaminen kuvataan prosessina, siihen kuuluu esimerkiksi määrittely-, suunnittelu- ja toteutusvaiheet kuten uuden ohjelmiston kehittämiseenkin. Erona on kuitenkin se, että uudistaminen tapahtuu olemassa olevan järjestelmän pohjalta, jolloin siihen kuuluu myös vanhojen ohjelmien analysointia, arviointia ja takaisinmallinnusta.(Harsu 2003: 165)

Ohjelmien uudistaminen eroaa uuden ohjelmiston kehitystyöstä siinä, että uudistettavasta ohjelmistosta tiedetään jo vaatimukset. Näin ollen vaatimusmäärittely jää pois ja vanhaan järjestelmään tehtyjä vaatimusmäärittelyjä voidaan käyttää hyväksi.

4.2.1 Uudistamisen tavoitteet

Ohjelmien uudistamisen tavoitteena on saada vanhoista lähdeohjelmista aikaiseksi uudet kohdeohjelmat, jotka on suunniteltu ja toteutettu käyttäen uudenaikaisia tekniikoita. Uudistamisen on ainakin joissakin tapauksissa tarkoitus tapahtua niin, että ohjelmien toiminnallisuus säilyy samana. (Harsu 2003: 178)

Harsu mainitsee kirjassaan (Harsu 2003: 178), että uudistamisen tavoitteisiin voidaan lukea toiminnallisuuden lisäämisen valmistelu, vaikka se ei olisikaan uudistamisen tavoitteena. Ennen uusien toiminnallisuuksien lisäämistä voi olla hyvä uudistaa ohjelma.

Uudistamisen seurauksena pyritään myös siihen, että ohjelman ylläpidettävyys helpottuisi. Uudistamisvaiheen dokumentaatiosta voi olla myös apua ohjelmiston ylläpidossa. Uudistamisella pyritään myös parantamaan ohjelmiston luotettavuutta. Vanha järjestelmä on voinut olla hyvinkin luotettava, mutta siihen tehtävien muutosten myötä luotettavuus on voinut kärsiä

Uudistaminen liittyykin hyvin läheisesti ohjelmien ylläpitoon. Uudistamisella voidaan korjata ylläpidon aiheuttamaa rakenteellista huononemista. Uuteen ympäristöön siirtymistä voidaan pitää sekä ylläpitona että uudistamisena, riippuen siitä miten suuria muutoksia se aiheuttaa ohjelmistoihin. (Harsu 2003: 179)

4.2.2 Uudistamisen riskejä ja riskeihin varautuminen

Vaikka uudistamista usein käytetään keinona lieventää riskejä ja vähentää ylläpitokustannuksia, myös itse uudistamiseen liittyy riskejä. Riskitekijät kannattaakin tunnistaa mahdollisimman aikaisessa vaiheessa ja varautua niihin etukäteen. (Harsu 2003: 174)

Itse uudistamisprosessiin liittyy riskejä. Uudistamista ei aina suunnitella riittävän huolellisesti eikä sille varata riittävästi aikaa. Tällöin voidaan olla siinä tilanteessa, että uudistamista tehdään muuan toiminnan ohella eikä kukaan ole siitä varsinaisesti vastuussa. (Harsu 2003: 174)

Uuden järjestelmän toteuttamiseen liittyy monia riskejä. Uudistamisessa voidaan uuteen järjestelmään yrittää lisätä liikaa uusia ominaisuuksia ja toimintoja yhdellä kertaa. Olemassa olevien tietorakenteiden siirtäminen uuteen järjestelmään voi myös tuottaa suuria haasteita. Vanhoista järjestelmissä olevat oliopiirteet, kuten luokat ja oliot, eivät välttämättä sovi uuteen järjestelmään. (Harsu 2003: 174-175)

Rosenberg (Rosenberg 12-13) mainitsee tekstissään uudistamiselle muun muassa seuraavia riskitekijöitä:

- Käsityö tulee kalliiksi.
- Käsityö on aikaa vievää
- Virheiden todennäköisyyden kasvaminen käsityössä
- Johdon hyväksynnän saavuttaminen.
- Asiantuntijoiden puute
- Vastuuhenkilön puuttuminen
- Tavoitteet ovat epäselviä.
- Uusien toimintojen lisäys, jotka eivät integroidu uuteen järjestelmään
- Uudistamisprosessin huono suunnittelu
- Uudistaminen tapahtuu muun toiminnan ohella

Riskeihin voidaan varautua tekemällä riskianalyysi, johon kuuluu riskien tutkiminen, riskien arviointi sekä riskien hallinta.

Toiminnanohjausjärjestelmän implementoinnissa olemassa olevat järjestelmät tulee määritellä ja evaluoida tarkasti, jotta pystytään

arvioimaan niiden ongelmien luonne ja laajuus, joita organisaatio tulee kokemaan implementoinnissa. Mitä monimuotoisempia, useampia ja monimutkaisempia olemassa olevat aikaisemmat järjestelmät ovat, sitä vaikeampaa on siirtyminen uuteen järjestelmään.

Useimmiten eniten hankaluuksia aiheuttaa käytössä olevien ohjelmien useat tietovarastot ja niitä hyödyntävät, usein pienehkön käyttäjäryhmän käyttämät pienehköön tehtäväkokonaisuuteen tehdyt sovellukset. Kun niitä käsitellään kokonaisuutena, ne ovat yksi hankalimmista esteistä liiketoiminnan tuottavuudelle ja suorituskyvylle. (Vuorenpää: 24)

4.3 Testaus

Tässä luvussa käsittelen testaukseen liittyviä yleisiä asioita, järjestelmätestausta sekä testauksen riittävyyttä ja laajuutta, koska testaus on iso osa Microsoft Dynamics AX versiovaihtoa.

4.3.1 Yleistä testauksesta

Testaukseen liittyvät työvaiheet ovat testauksen suunnittelu, testiympäristön luonti, testin suorittaminen ja tulosten tarkastelu. Näihin työvaiheisiin ja niihin läheisesti liittyvään virheen jäljitykseen ja korjaukseen (debugging) kuluu tyypillisesti yli puolet ohjelmistoprojektin resursseista, joten testauksen läpivientiin parhaalla mahdollisella tavalla kannattaa kiinnittää huomiota. Testauksen määrä on aina kompromissi käytettävissä olevien resurssien (aika, raha, välineet ym.) ja luotettavuudessa saavutetun varmuuden välillä. (Haikala 2002: 281)

Testaukseen käytettyjen työtuntien tai testitapausten määrä ei välttämättä ole tae testauksen tehokkuudesta. Muutamien tuntien testaus pienellä huolellisesti suunnitellulla testitapausjoukolla voi johtaa parempaan tulokseen kuin päiväkausien umpimähkäinen kokeilu. (Haikala 2002: 282)

4.3.2 Järjestelmätestaus

Järjestelmätestausta suoritetaan järjestelmän kokonaisvaltaisen toiminnallisuuden varmistamiseksi. Siinä todennetaan, että kaikki vaatimukset on täytetty ja että ne on täytetty riittävän laadukkaasti.(McConnell : 220) Tarkastuksia ja ohjelmakoodin staattista analysointia eli arviointia kutsutaan usein myös testaukseksi. Järjestelmätestaus onkin tyypillisesti tarkastelun, arvioinnin ja testisuoritusten kombinaatio. Toimintojen tarkoituksena on tutkia ohjelmavirheitä. (McGregor & Sykes 2001: 27-28)

Tarkastelu perustuu järjestelmän tyypillisten virheiden listan tarkasteluun. Suurin osa listalla olevista virheistä perustuvat konversiossa tehdyn ohjelmakoodin varmistamiseen. Tarkastelun kohteena ovat muuttujat, pointterit ja referenssit, joiden tulee kaikkien olla alustettuja järkevillä arvoilla ennen niiden käyttöä. Moderneissa olio-ohjelmoinnin kääntäjissä onkin mahdollista ajaa jonkinlainen tarkistuslista-ajo. (McGregor & Sykes 2001: 27-28)

Arviointi on taas ohjelman lähdekoodin tutkimista siten, että tavoitteena löytää toiminnallisia vikoja ja virheitä ohjelmistosta. Arvioinnissa poraudutaan ohjelmiston jokaiseen osioon vaikka tietty osio ohjelmistosta ei olisikaan käytössä. Arvioinnin tarkoituksena on siis löytää ohjelmistosta löytyvät logiikkaviat, joita ei ole aikaisemmin löydetty tai ne on väärinymmärretty. Arvioinnissa tarkastellaan muun muassa sitä, että muuttujan nimet on valittu oikein ja algoritmit ovat juuri niin tehokkaita kuin niiden kuuluisikin olla. (McGregor & Sykes 2001: 27-28)

Testisuoritukset tehdään ajamalla ohjelmaa. Testaajan tavoitteena on analysoida toimiiko järjestelmä halutulla tavalla, kun sille syötetään tiettyjä arvoja. Testaajien haasteena on kuitenkin se, että he varmistavat syötettävien arvojen sekä palautusarvojen oikeellisuuden, sekä että he osaavat tarkastella palautusarvoja oikealla tavalla. (McGregor & Sykes 2001: 27-28)

Hyväksi havaittu tapa on toteuttaa testaus siten, että testaajat jaetaan omiin ryhmiin. Jokainen ryhmä testaa sen osan ohjelmasta, jonka toiminnallisuuden he tietävät parhaiten.

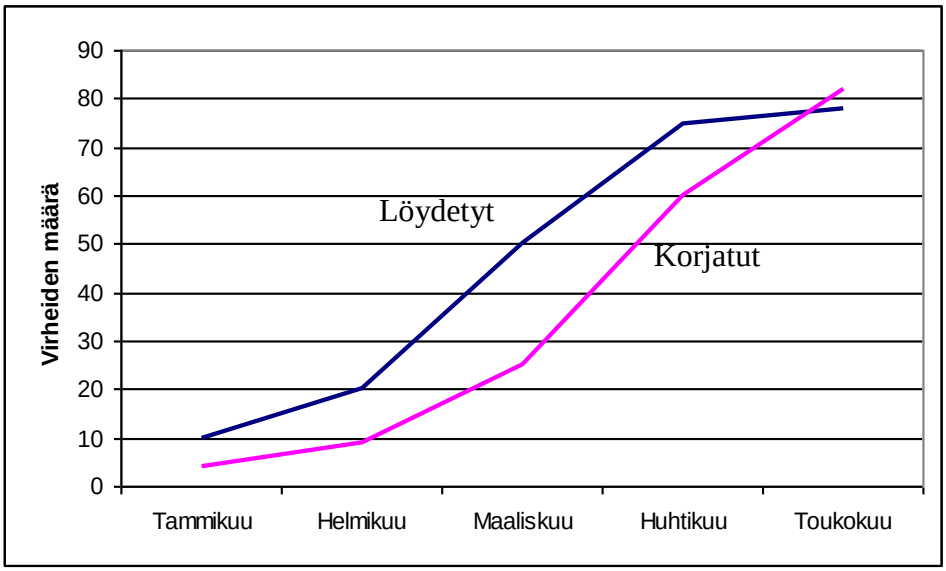
Testausta voidaan suorittaa joko samaan aikaan kun kehitystä vielä tehdään tai vasta siinä vaiheessa, kun ohjelma on saatu vaiheeseen jossa se olisi valmis toimitettavaksi asiakkaalle.

4.3.3 Testauksen riittävyys ja laajuus

Tarvittavan testauksen määrää on vaikea arvioida. Varsinkin järjestelmätestauksessa voidaan testausta jatkaa ”kunnes aika ja/tai rahat loppuvat”. Tuotekehitystyössä testauksen lopettaminen on usein kompromissi tuotteessa olevien vikojen aiheuttamien kustannusten ja markkinoilta myöhästymisen aiheuttaman tuoton menetyksen välillä.
(Haikala 2002: 290)

Testauksen lopettamiselle tulisi aina asettaa hyväksymiskriteerit, jotka määritellään testaussuunnitelmassa. Järjestelmätestauksessa kriteerit voi liittyä esimerkiksi kumulatiivisesti löydettyjen virheiden määrään; kun virhekäyrä tasaantuu, voidaan testaus lopet-

taa(Kuva 4). Vaikeutena testauksen riittävyyden arvioinnissa on yleensä se, että aikaisemmin määritetyt kiinteät resurssit ja aikataulukin on jo lyöty lukkoon ja kerrottu asiakkaalle. Saattaa olla vaikeaa arvioida projektin kestoa, mikäli testauksen lopetuskriteerinä pidetään vain virhekäyrän tasaantumista, koska tasaantumiseen tarvittava työmäärä ei ole tiedossa etukäteen. (Haikala 2002: 291)



Kuva 4: Esimerkki järjestelmätestauksen virhekäyrästä

McConnelin mukaan järjestelmätestauksessa tulisi kuitenkin testata järjestelmä koko laajuudessaan. Testitapaukset olisi myös suunniteltava niin, että niiden avulla voitaisiin todentaa ohjelmiston jokaisen vaatimuksen toteutus ja jokaisen koodirivin virheetön toiminta. (McConnell : 221)

5 Microsoft Dynamics AX versionvaihto

Microsoft Dynamics AX -tuotteen versionvaihtoon liittyy, sekä uuden toiminnallisuuden lisäämistä, vanhan toiminnallisuuden säilyttämistä, sekä jo olemassa olevien toimintojen uudistamista. Tässä luvussa keskitytään tuotteen versionvaihdokseen liittyviin keskeisiin vaiheisiin tarkemmin.

5.1 Rajoitukset

Vain Microsoft Dynamics AX 3.0 -versio voidaan päivittää 4.0 versioon.

Ennen päivitysprosessin aloittamista täytyy varmistaa, että käyttöoikeudet ovat riittävät tilillä, jolla versionvaihdon tehtäviä suoritetaan. Alla olevassa taulukossa on määritelty minimivaatimukset tehtäville.

Toiminto	**Vaadittava käyttöoikeustaso**
Sovelluspalvelimen asennus (Application Object Server (AOS))	Lokaalin palvelimen järjestelmän valvoja ryhmän jäsen
Microsoft SQL Server tietokannan luonti	SQL Server, Database Creators jäsenyys
Connect AOS to a SQL Server database	SQL Server, Database Security Administrators jäsenyys
Sovelluksen tiedostopalvelimen asennus	Lokaalin palvelimen järjestelmän valvoja ryhmän jäsen
Microsoft Dynamics AX työasema asennus	Lokaalin palvelimen järjestelmän valvoja ryhmän jäsen
Tietokannan valmistelevan työkalun ajaminen (DB Upgrade Preparation Tool)	SQL Server – minimissään, ddl_admin, db_datareader ja db_datawriter roolit.
Päivityksen tarkistuslistan ajaminen(Upgrade checklist)	Microsoft Dynamics AX järjestelmän valvoja.

5.2 Toteutussuunnitelman hahmottaminen

Toteutussuunnitelma koskee ohjelman laajennuksia tai nykyisen Microsoft Dynamics AX version muutoksia.

Päivittämisen kustannusten arvioimiseksi tarvitaan perusteellinen analyysi siitä kuinka paljon ja kuinka laajoja muutoksia edelliseen standardiversioon on tehty.

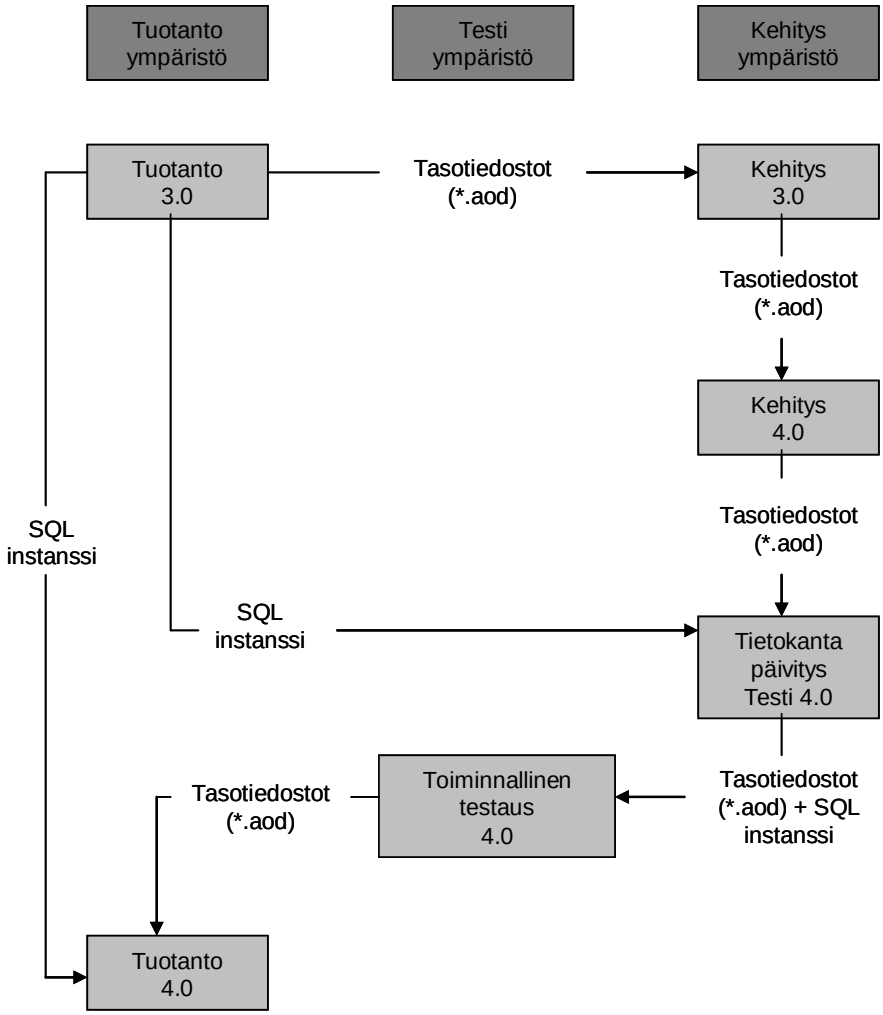
Hyvänä lähtökohtana kustannusten arvioinnissa voidaan käyttää apuna listaa (Kuva 5), jossa on lueteltu versionvaihdon kohteena olevan ohjelmiston objektimuutosten (Counter) ja niin sanottujen varjo-objektien (Shadows) lukumäärä. Varjo-objektit ovat kopio standardiohjelmiston objektista, johon muutokset on toteutettu ja jolla standardiohjelmiston objekti on korvattu. Alemman tason muutokset eivät vaikuta tähän objektiin. Microsoft Dynamics AX sisältää taulun, johon on listattu kaikki muutokset eri objekteissa, sitä voidaan käyttää hyväksi tämän kaltaisen listan tekemisessä.

Type	Counter	Shadows
Table	21	8
TableField	69	
TableFieldGroup	32	8
TableIndex	14	1
TableRelation	2	
TableInstanceMethod	20	10
TableStaticMethod	7	
TableMap	2	2
ExtendedType	23	
Enum	6	3
ConfigurationKey	1	
SecurityKey	1	
Macro	2	1
Class	82	16
ClassStaticMethod	153	8
ClassInstanceMethod	1225	32
Form	32	17
Report	5	1
Job	12	
Menu	1	
DisplayTool	7	
OutputTool	4	
ActionTool	1	
Grand Total	1722	107

Kuva 5: Listaus muutetuista objekteista

Tuotantoympäristöjen päivitykseen käytettävä aika on asiakkaalle kriittinen, sillä toiminnassa oleva vanha versio on poistettava käytöstä päivityksen ajaksi. Päivitykseen kuluva aika riippuu myös tietokannassa olevan tiedon määrästä sekä asiakkaan käytössä olevasta infrastruktuurista.

Tietojen päivittäminen testiympäristöön onkin tässä tapauksessa tärkeässä roolissa, sillä sen avulla saadaan käsitys siitä kuinka paljon aikaa tietojen päivittämiseen kuluu. Testiympäristön täytyykin olla verrattavissa tuotantoympäristöön, jotta tulokset olisivat luotettavia. Kuvassa 6 on kuvattu päivityksen etenemistä eri ympäristöjen välillä.



Kuva 6: Päivityksen vaiheet usean ympäristön välillä

5.3 Valmistelevat toimet

Päivityksen perustana käytetään asiakkaan tuotantoympäristöä, jota asiakas käyttää aktiivisesti tuotantopäivitykseen asti. Tästä johtuen täytyy 3.0 ympäristöstä luoda kopio, jossa voidaan tehdä tarvittavat valmistelut lähdekoodille ennen sen siirtoa 4.0 järjestelmään.

5.3.1 Vanhan koodin ja tietokannan varmistus

Varmistuksen tekeminen aikaisemmasta versiosta tekee paluun vanhaan järjestelmään mahdolliseksi, jos jotain menee vikaan.

Varmistus on hyvä suunnitella etukäteen: missä päivityksen vaiheessa se toteutetaan ja mitä on tarkoitus varmistaa.

Minimissään varmistus tehdään tiedostopuusta, jossa ohjelmisto sijaitsee. Tiedostopuu sisältää kaikkien tasojen tiedostot (.aod). Myös tietokannasta täytyy ottaa varmistus.

5.3.2 Uuden ympäristön luonti

Uusi ympäristö luodaan asentamalla standarditoiminnallisuudet sisältävä Microsoft Dynamics AX 4.0. Ensijulkaisun jälkeen järjestelmä on saanut lukuisan määrän korjauksia, joten niiden asentaminen tässä vaiheessa on suotavaa.

Kehitysympäristö, johon asiakkaalle tehtyjä muutoksia päivitetään, on siis standarditoiminnallisuuden sisältävä ympäristö, johon on asennettu kaikki Microsoftin julkaisemat päivitykset.

5.4 Vanhan järjestelmän päivittäminen

Varsinainen järjestelmän päivittäminen koostuu kahdesta osiosta: ohjelmakoodin päivittämisestä ja tietokannan konvertoimisesta.

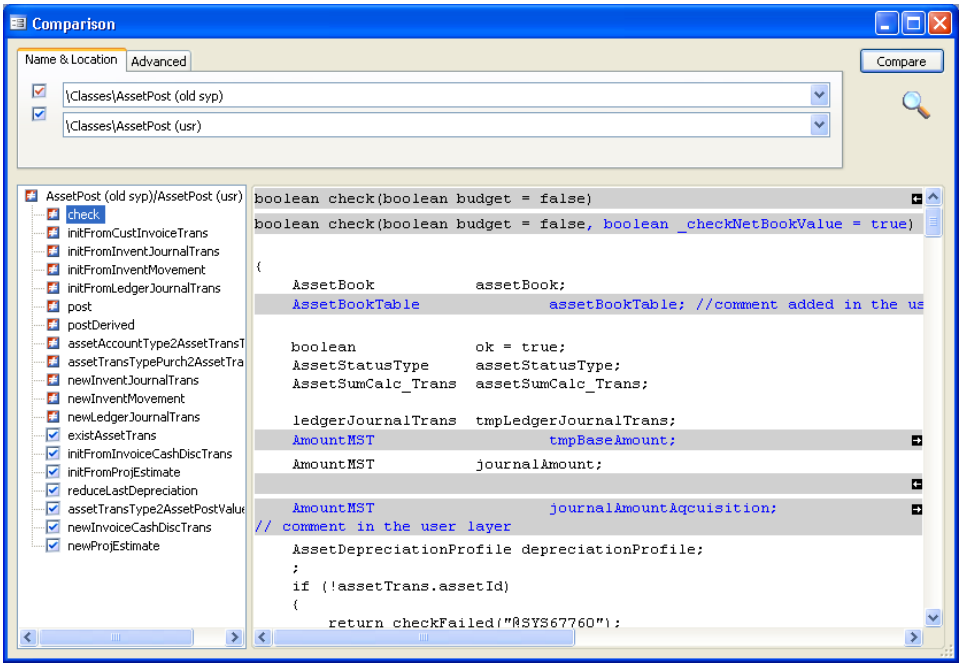
5.4.1 Ohjelmakoodin päivitys kehitysympäristöön

Suurin ja työläin vaihe on vanhan ohjelmakoodin päivittäminen uuteen järjestelmään. Vanhoja asiakkaalle tehtyjä muutoksia ei voida siirtää uuteen järjestelmään suoraan, koska toiminnon aikaisempi perustoiminnallisuus on voinut muuttua. Tämä tarkoittaa käytännössä sitä, että jokainen itse kirjoitettu ohjelmakoodin rivi täytyy käydä läpi ja varmistaa että se on vielä toiminnallisuudeltaan sellainen kuin se on alun perin suunniteltu olevan.

Työn määrä onkin täysin riippuvainen siitä, kuinka paljon asiakaskohtaisia muutoksia järjestelmään on tehty. Järjestelmästä saatava lista muuttuneista objekteista on hyvä tapa tunnistaa, tarpeettomat ja käytöstä poistuneet toiminnot. Jokin vanha standarditoiminnallisuus voi käyttää muuttuneita tai kokonaan uusia funktioita. Myös vanhojen toiminnallisuuksien uudistamista on hyvä harkita tässä vaiheessa, sillä monen vanhan komponentin sisältämä koodi voi olla joko rakenteellisesti huonoa tai se ei edes sovi uuden järjestelmän kassan yhteen, esimerkiksi standardi toiminnallisuuteen tulleiden luokkamuutosten takia.

Jokaiselle tasolle täytyy perustaa oma päivitysympäristönsä. Jos muutoksia on siis vaikka VAR ja CUS tasoilla, täytyy päivitysympäristöjä luoda kaksi. Usean tason päivittämisen ongelmana yhdessä ympäristössä on se, ettei alemman tason muutoksia voida poistaa.

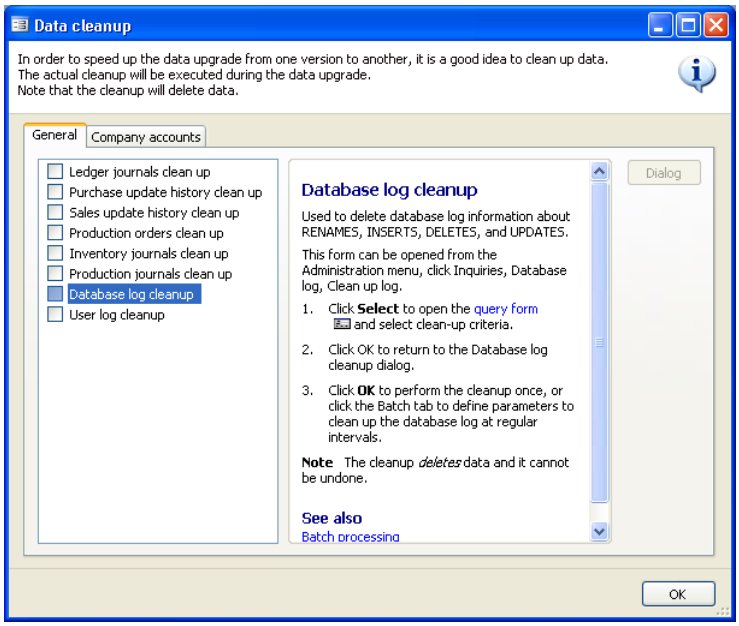
Vanhan ohjelmakoodin siirrossa uuteen järjestelmään voidaan käyttää hyväksi Microsoft Dynamics AX järjestelmästä löytyvää lähdekoodin vertailutyökalua (Kuva 7). Työkalulla voidaan verrata kohde- ja lähdetason ohjelmakoodia samassa ikkunassa. Kuvassa näkyvät siniset tekstit ovat USR-tasolla olevaa koodia ja ne eroavat käytössä olevasta SYP-tasosta.



Kuva 7: Kahden tason eroavaisuusvertaus.

5.4.2 Tietokannan puhdistus

Tietokannan olemassa olevat lokit on hyvä puhdistaa. Nämä lokit sisältävät tietokantaan tehtyjen muutosten historian, kuten tiedon uudelleen nimeämiset, lisäykset ja poistot. Lokin sisältämät tiedot eivät vaikuta tietokannan alkuperäisyyteen ja eheyteen, ne vain hidastavat tietokantaa ja kasvattavat sen kokoa. Data cleanup työkalun avulla voidaan (Kuva 8) valita mitä lokeja halutaan puhdistaa.

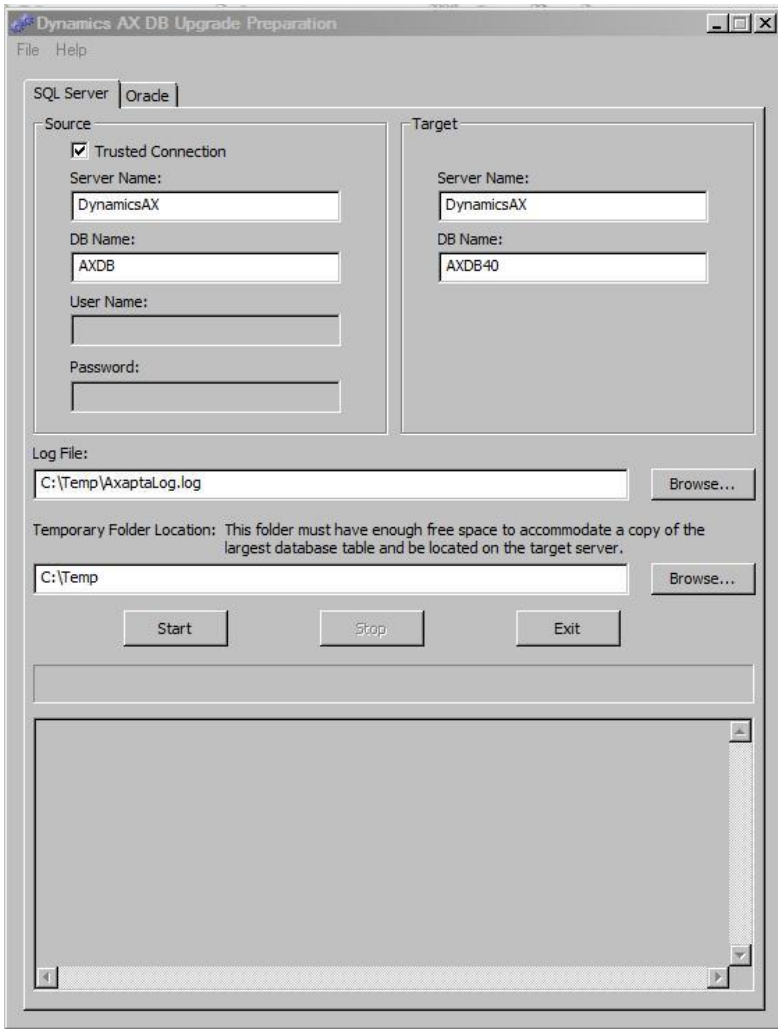


Kuva 8: Tietokannan puhdistustyökalu, jossa valitaan mitä lokeja puhdistetaan.

5.4.3 Tietokannan konvertointi

Version muutos tuo mukanaan tyypillisesti tietokantaan joitain muutoksia. Jotkut taulut ja kentät poistuvat käytöstä. Version muutos tuo tullessaan myös uusia tauluja ja kenttiä. Poistuneiden taulujen ja kenttien tietosisältö siirretään yleensä joihinkin versionvaihdoksessa luotuihin uusiin tauluihin ja kenttiin. Päivitysframework osaa käsitellä näiden tietojen muutokset ja siirtämiset standarditoiminnallisuuden osalta. Mikäli standarditoiminnallisuuteen on tehty muutoksia, esimerkiksi kenttien lisäyksiä, täytyy ne huomioida päivitysprosessin aikana.

Päivityspaketissa tulee mukana ulkopuolinen työkalu (Dynamics AX DB Upgrade Preparation tool), joka valmistelee tietokannan uutta versiota varten. Työkalu lukee vanhan version tietokannan ja siirtää sen uuden järjestelmän tietokantaan.(Kuva 9)

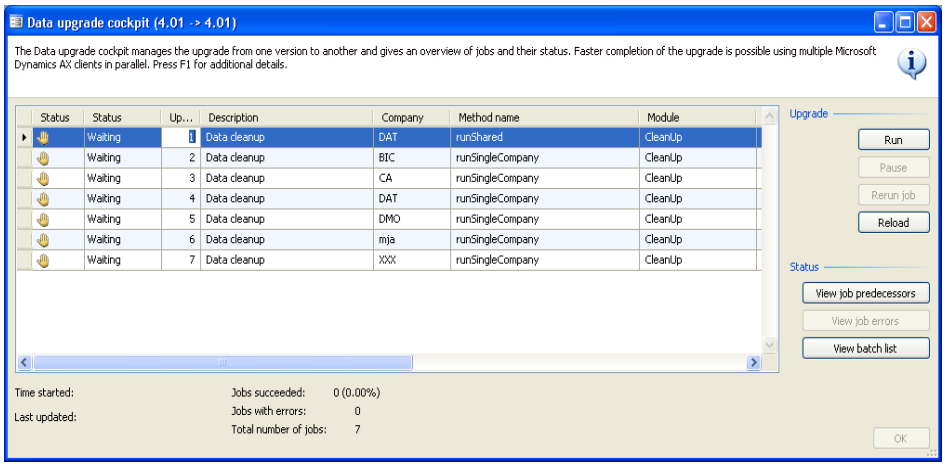


Kuva 9: Dynamics AX DB Upgrade Preparation tool

Tietokannan lopullinen päivitys koostuu kahdesta osasta, esisynkronoinnista ja jälkisynkronoinnista. Ensimmäinen osa toteutetaan ennen kuin uusi skeema eli tietokannan rakenne on synkronoitu tietokantaan ja toinen osa välittömästi sen jälkeen.

Ensimmäinen osa eli esisynkronointi tarkoittaa sitä, että kaikki olemassa oleva data esikäsitellään siten, ettei uniikkeja indeksejä rikota. Tässä vaiheessa esisynkronoinnissa käytettävissä skripteissä, uusiin tauluihin ja kenttiin ei voi vielä viitata, sillä ne eivät ole vielä fyysisesti kannassa.

Toinen osa eli jälkisynkronointi luo uudet taulut ja kentät fyysisesti kantaan, sekä purkaa esikäsittelyä varten tehdyt muutokset. Synkronointi tapahtuu erillisellä työkalulla. (Kuva 10)



Kuva 10: Ikkuna jolla jälkisynkronoinin tehtävät ajetaan.

5.5 Testaus

Testausta varten täytyy luoda kokonaan oma ympäristönsä, johon uusi asiakkaan muutokset sisältävä ohjelmakoodi kopioidaan ja johon testauksen aikana ilmenevät virheet voidaan korjata mahdollisimman nopeasti.

Riippuen siitä kuinka paljon standarditoiminnallisuuteen on tehty asiakaskohtaisia muutoksia, on järjestelmä testattava ainakin siinä laajuudessa kuin asiakas tulee sitä käyttämään. Moduulit joihin asiakaskohtaisia muutoksia ei ole kohdistunut on myös hyvä testata, sillä jokin toiseen moduulin tehty muutos voi heijastua vääränä toiminnallisuutena toisen moduuliin.

Testaukset on hyvä suunnitella etukäteen, jotta testauksesta saataisiin mahdollisimman tehokasta. Muutamien tuntien testaus pienellä, huolellisesti suunnitellulla testitapauksella voi johtaa parempaan tulokseen kuin tuntien umpimähkäinen kokeilu.

Ohjelman sisäinen logiikka on muuttunut joiltain kohdin, joten tarkastelevaa ja arvioivaa testausta on myös hyvä suorittaa. Tällä tavalla varmistutaan myös, että ohjelmaan tehdyt asiakaskohtaiset muutokset toimivat loogisesti edelleen määritellyllä tavalla.

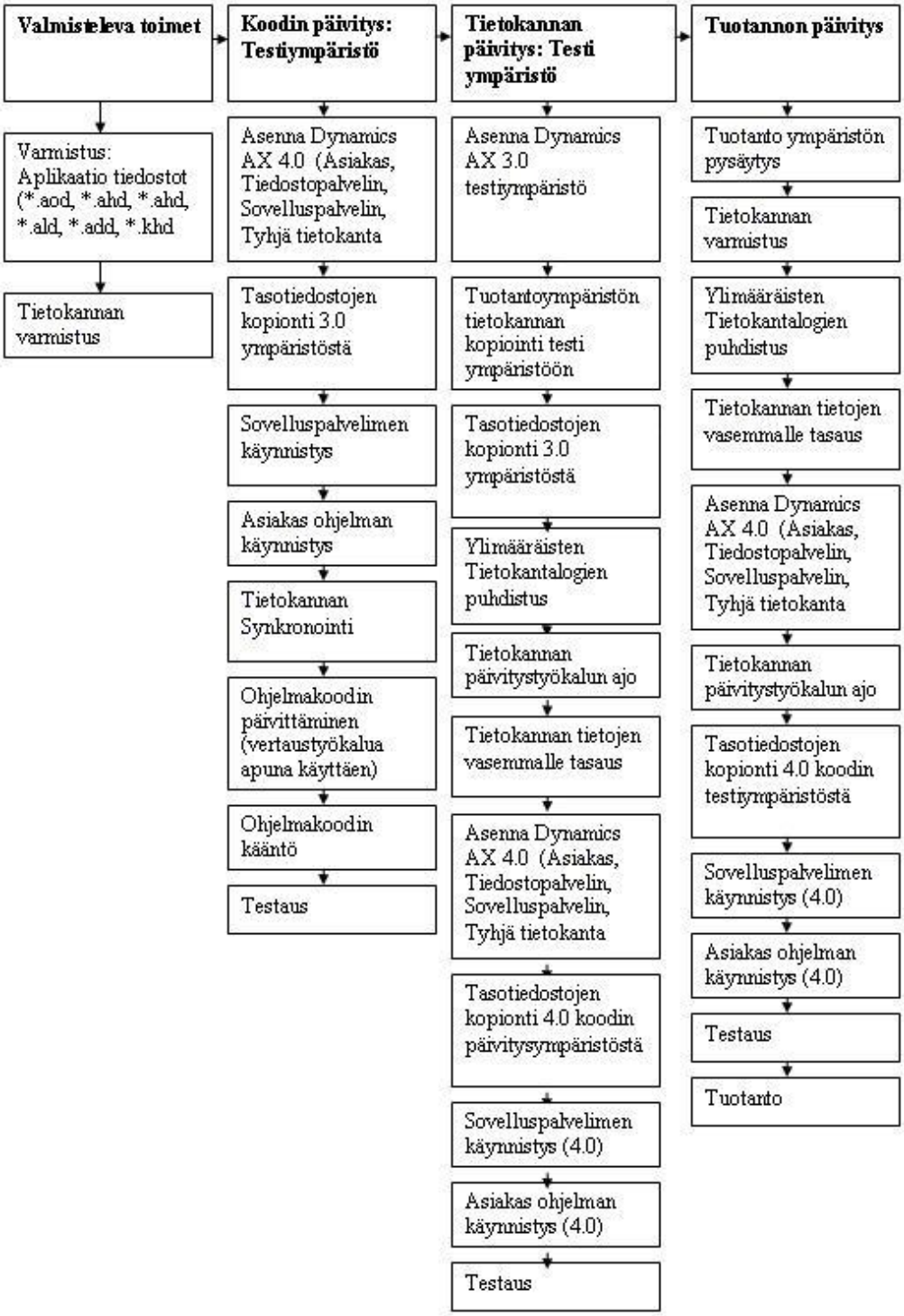
Microsoft Dynamics AX 4.0:n jokainen moduuli vaati erikoistietämystä moduulin toiminnasta ja sen ominaisuuksista. Testaajat täytyykin valita siten, että jokaiselle moduulille on oma tai omat testaajansa, jotka ovat erikoistuneet moduulin toimintaan.

5.6 Tuotantoympäristön käyttöönotto

Koska vanhan järjestelmän ohjelmakoodi on käyty läpi ja ohjelman toiminnallisuus on testattu, voidaan tuotantoympäristön luoda testiympäristön pohjalta. Tuotantoympäristön luonti vaatii kuitenkin vielä tuotannon tietokannan konvertoinnin vanhasta järjestelmästä uuteen.

5.7 Prosessikaavio

Alla olevassa prosessikaaviosta (Kuva 11) käy ilmi kaikki tehtävät, joita versionvaihdoksessa tarvitaan. Kuvasta nähdään myös kuinka tehtävissä tulisi edetä.



Kuva 11: Päivityksen prosessikaavio

5.8 Versionvaihdon toteutus

Ensimmäinen versionvaihtokokeilu toteutettiin luomalla uusi puhdas 4.0 ympäristö, johon päivitettiin toukokuussa 2008 Mepcon tuotantokäytössä olevan 3.0 version toiminnallisuudet sekä lisättiin muita omaan käyttöön hyödylliseksi katsottuja lisätoiminnallisuuksia, joita oli toteutettu asiakkaille.

Vanhan ohjelmakoodin siirron jälkeen uudessa järjestelmässä ilmeni muutamia loogisia ongelmia lähdekoodin osalta, jotka aiheuttivat virheitä järjestelmän toiminnallisuudessa, näihin virheisiin osattiin kuitenkin varautua hyvin ja ne saatiin pikaisesti korjattua.

Tällä hetkellä uusi 4.0 ympäristö on jo ohjelmakoodin osalta suurimmaksi osaksi valmiina ja osa vanhoista ja uusista toiminnallisuuksista on jo testattu. Pieni määrä ohjelmakoodia vaatii vielä optimointia sekä uudistamista, jotta järjestelmä saadaan toimimaan mahdollisimman tehokkaasti. Tietokannan konvertointia on myös testattu, siinä ilmeni joitain pieniä ongelmia. Ongelmakohtiin täytyy vielä perehtyä tarkemmin.

6 Yhteenveto

Tämän opinnäytetyön tarkoituksena oli paneutua ylläpidollisiin ja uudistamista koskeviin toimenpiteisiin, niissä kohdattaviin ongelmiin sekä testaukseen, jotta tulosten pohjalta voitaisiin suunnitella Microsoft Dynamics AX versionpäivitysprosessi mahdollisimman tarkasti ja osattaisiin varautua mahdollisesti siinä esiintyviin ongelmiin.

Opinnäytetyön tuloksena huomattiin, että prosessin tärkein ja vaativin vaihe on vanhan ohjelmakoodin päivittäminen uuteen järjestelmään, johon täytyy kiinnittää eritystä huomiota. Ohjelmakoodin päivittämisessä tulikin ottaa huomioon oliko jokin komponentin uudelleenrakentamista hyötyä tulevaisuuden ylläpitokustannuksia silmälläpitäen.

Opinnäytetyölle asetetut tavoitteet saavutettiin hyvin. Tuloksena saatiin kokonaiskuva prosessin laajuudesta ja tietoa siitä mihin asioihin prosessin eri vaiheissa tulee kiinnittää huomiota, sekä tarkempi kuva prosessiin liittyvistä toiminnoista. Ongelmakohtiin osataan nyt varautua paremmin ja aikataulujen laadinta on nyt helpompaa, kun tiedetään mihin täytyy kiinnittää erityistä huomiota ja mikä vaihe vie eniten aikaa.

Opinnäytetyön myötä olen myös saanut paremman kuvan ylläpitoon ja uudistamiseen liittyvistä asioista, sekä niihin liittyvistä ongelmista.

Jatkossa työn tuloksia tullaan hyödyntämään uuden 4.0 järjestelmän varsinaisessa tuotantoonotossa, joka tapahtuu kesäkuussa 2008, sekä mahdollisesti asiakkaille tehtävissä ympäristön versionvaihtoprojekteissa.

LÄHTEET

Bishop, Ronnie & Lucas, Mary Ellen, (toim.) Anderegg, Travis & Knox, Janice 2005.
CNERPS study guide a novice's guide to ERP systems.
Eau Claire (Wis.): Resource Publishing

Harsu, Maarit 2003.
Ohjelmien ylläpito ja uudistaminen.
Helsinki: Talentum Media Oy

Sysoptima 2005.
Functional Modules of ERP Software
[online][viitattu 13.03.2007]
http://www.sysoptima.com/erp/erp_modules.php

Haikala Ilkka, Märijärvi Jukka, 2002.
Ohjelmistotuotanto Helsinki:
Talentum Media Oy

John D. McGregor, David A. Sykes 2001.
A practical guide to testing object-oriented software.
Addison Wesley Professional

Kettunen, J. ja Simons, M. (Toim.). 2001.
Toiminnanohjausjärjestelmän käyttöönotto pk-yrityksessä. Espoo:
VTT Automaatio

Linda H. Rosenberg. Software Re-engineering.
[online][viitattu 14.01.2008].
http://satc.gsfc.nasa.gov/support/reengrpt.PDF

IDG News Service 2006. Microsoft starts shipping Dynamics AX 4.0
[online][viitattu 28.02.2007].
http://www.itworld.com/App/670/060612dynamicsax4/

Markus, M. Lynne & Tanis, Cornelis & van Fenema, Paul C. 2000.
Enterprise resource planning: multisite ERP implementations.
Communications of the ACM.Volume 43 issue 4 s. 42-46

Microsoft. Microsoft Dynamics AX Dynamics AX
[online][viitattu 14.1.2008]
http://www.microsoft.com/finland/dynamics/ax/default.Microsoft.mspx

Pohjonen, Risto 2002. Tietojärjestelmien kehittäminen.
Jyväskylä: Docendo Finland Oy

Jussi Koskinen, 2000. Ohjelmien ylläpito ja käänteistekniikat
Jyväskylän yliopisto, Informaatioteknologian tiedekunta
[online][viitattu 15.11.2007].
http://users.jyu.fi/~koskinen/lectio.htm

Ruuska K., 1997.
Projekti hallintaan. Jyväskylä, Talentum Media Oy.

Steve, McConnell 1997.
Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last
Redmond, Wa.: Microsoft Press

Tuomo Vuorenpää 2007.
Toiminnanohjausjärjestelmän kehittämisen arviointi ja sen hyödyntäminen ylläpidossa.
Tampereen yliopisto Tietojenkäsittelytieteiden laitos

Wallace, Thomas F. & Kremzar, Michael H. 2001.
ERP making it happen: the implementers' guide to success with enterprise resource planning.
New York, (N.Y.): Wiley