HAAGA-HELIA
ammattikorkeakoulu

Toiminnanohjausjärjestelmän päivittäminen

Case: Microsoft Dynamics AX

Åke Heino

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2013

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

17.10.2013

Tietojenkäsittelyn koulutusohjelma

Tekijä tai tekijät Åke Heino	**Ryhmätunnus tai aloitusvuosi** 2008
Raportin nimi Toiminnanohjausjärjestelmän päivittäminen Case: Microsoft Dynamics AX	**Sivu- ja liitesivumäärä** 47 + 4
Opettajat tai ohjaajat Mutikainen Jukka	

Tutkimuksen tavoitteena oli selvittää, mitkä ovat tyypillisimmät syyt toiminnanohjausjärjestelmien päivitysprojektien epäonnistumisille ja miten epäonnistumisia voitaisiin välttää. Toiminnanohjausjärjestelmien päivitysprojektien haasteita ja ongelmia käsiteltiin teoriaosuudessa rajaamatta aihetta mihinkään tiettyyn toiminnanohjausjärjestelmään. Haastattelut rajattiin kuitenkin koskemaan vain Microsoft Dynamics AX toiminnanohjausjärjestelmää.

Tutkimuksessa käytettiin kvalitatiivista eli laadullista menetelmää. Se toteutettiin haastattelemalla pitkään Microsoft Dynamics AX toiminnanohjausjärjestelmän parissa työskennelleitä ihmisiä. Haastatellut edustivat järjestelmätoimittajan tai -valmistajan puolta. Kaikkien kanssa sovittiin yksilölliset haastattelut ja heille toimitettiin haastattelukysymykset etukäteen tutustuttavaksi. Haastateltavat saivat kuitenkin vapaasti kertoa kokemuksistaan ja käsityksistään kysymysten toimiessa vain runkona.

Haastateltavat kuvasivat omien kokemustensa pohjalta useita syitä, miksi toiminnanohjausjärjestelmien päivitysprojektit epäonnistuvat tai onnistuvat. Mielenkiintoista oli se, että tässä kohtaa sekä teoria että käytäntö kohtasivat hyvin, joskin ulkopuolisen projektipäällikön käyttämisessä oli eroa haastateltavien mielipiteiden ja aikaisempien julkaisujen välillä. Päivitysprojektien epäonnistumisten seurauksista usea haastateltava nosti esiin liikevaihdon menetyksen, kun tilauksia ei saada toimitettua eikä laskuja lähetettyä. Järjestelmämuutosten päivittämisessä ja siirtämisessä pidettiin tärkeänä kokonaisuuden hallintaa sekä toiminnallisesti että kustannuksellisesti.

Tutkimuksessa onnistuttiin kartoittamaan runsaasti erilaisia syitä päivitysprojektien epäonnistumisille. Toisaalta löydettiin paljon keinoja, joilla epäonnistumisia voitaisiin välttää.

Toiminnanohjausjärjestelmiä päivitetään jatkuvasti ja valitettavan usein päivitysprojektit epäonnistuvat. Yrityksissä onkin tarve saada selkeitä ohjeita ja suosituksia parhaista käytännöistä toiminnanohjausjärjestelmien päivittämiseen. Tutkimuksessa annetaankin ohjeita ja neuvoja päivitysprojektien onnistumisen edistämiseksi ja karikkojen välttämiseksi.

Asiasanat
Toiminnanohjaus, päivitys, atk-järjestelmät

HAAGA-HELIA
University of Applied Sciences

Abstract

17 October 2013

Degree Programme in Information Technology

Authors Åke Heino	**Group or year of entry** 2008
The title of thesis Upgrading ERP system Case: Microsoft Dynamics AX	**Number of pages and appendices** 47 + 4
Supervisor(s) Mutikainen Jukka	

The goal of this Bachelor's thesis was to find out the most typical reasons for ERP systems upgrade projects to fail and how failures could be avoided. The challenges and problems of ERP systems upgrade projects were addressed without limiting them to any specific ERP system. The interviews though were limited only to Microsoft Dynamics AX ERP system.

Qualitative research method was used in this thesis. The study was carried out by interviewing people that had a long career with Microsoft Dynamics AX ERP system. The interviewees worked either for Microsoft or for partners delivering ERP systems. All interviews were conducted individually and the interviewees were given interview questions in advance. The interviewees were allowed freely to express their views and experiences while questions worked only as a framework.

The interviewees gave several reasons for why ERP system upgrade projects fail or succeed based on their own experience. Interesting part here was that both the theory and practice met very well, although there were differences between interviewee opinions and previous publications regarding usage of an outsourced project manager. As regards upgrade project failures, many of the interviewees brought up loss of revenue, inability to deliver orders and inability to send invoices. Taking both functionally and cost- wise overall picture into account was considered crucial when upgrading or transferring system modifications.

This study succeeded in finding several different reasons for why upgrade projects fail and also discovered plenty of ways to avoid failures.

ERP systems are constantly upgraded and unfortunately quite often they fail. Companies have a demand for clear instructions and recommendations for best practices to upgrade ERP projects. This study tries to meet that demand and gives several instructions and guidelines to help manage upgrade projects successfully and to avoid pitfalls.

Key words
ERP, Upgrade, IT-systems

Sisällys

1 Johdanto

Toiminnanohjausjärjestelmä on yrityksen keskeisin tietojärjestelmä, joka yleensä koskettaa yrityksen kaikkia toimintoja. Sillä voidaan ohjata mm. kirjapitoa, reskontraa, varastohallintaa, tuotantoa, huoltoa, laskutusta ja projekteja. Toiminnanohjausjärjestelmille on tyypillistä kaikkien siinä olevien moduulien saumaton ja reaaliaikainen yhteistoiminta. Ne pyrkivätkin tehostamaan yritysten toiminta poistamalla manuaalisia työvaiheita. Toiminnanohjausjärjestelmät tarjoavat yleisiä ja laajalti käytössä olevia tapoja hallita liiketoimintaa, mikä tarkoittaa että yrityksen on joko sopeutettava oma toimintatapansa niiden mukaiseksi tai muokattava tuominnanohjausjärjestelmää vastaamaan yrityksen prosesseja. Mikäli yritys lähtee tekemään tai teettämään asiakaskohtaisia järjestelmämuutoksia eli ns. räätäleitä toiminnanohjausjärjestelmään, sen tulee ottaa huomioon muutosten aiheuttamat haasteet tulevissa päivityksissä. Tässä tutkimuksessa paneudutaan noihin haasteisiin.

Toiminnanohjausjärjestelmiä päivitetään jatkuvasti ja valitettavan usein päivitysprojektit epäonnistuvat. Lukuisissa yrityksissä ollaan valinnan edessä, jossa pitää päättää, miten menetellä vanhentuvan toiminnanohjausjärjestelmän kanssa. Yrityksissä onkin tarve saada selkeitä ohjeita ja suosituksia parhaista käytännöistä toiminnanohjausjärjestelmien päivittämiseen. Tässä tutkimuksessa asiaa lähestyttiinkin toiminnanohjausjärjestelmää päivittämään lähtevän yrityksen näkökulmasta, eikä järjestelmätoimittajan näkökulmasta. Tutkimus tarjoaa kuitenkin paljon hyödyllisiä huomioita ja vinkkejä myös järjestelmätoimittajille.

Hyvin monia toiminnanohjausjärjestelmiä on muokattu asiakaskohtaisesti vastaamaan paremmin yrityksen liiketoimintaprosesseja. Haasteeksi muodostuu kuitenkin, miten toimia järjestelmämuutosten kanssa. On mahdollista, että uudessa versiossa on vakiona toimintoja, jotka vastaavat aikaisemmin tehtyjä järjestelmämuutoksia. Näiden toiminnallisuuksien vertailu on hyvin työlästä, koska se pitää tehdä sekä ohjelmointikoodi- että liiketoimintalähtöisesti.

Tutkimuksen tavoitteena oli selvittää, mitkä ovat tyypillisimmät syyt toiminnanohjausjärjestelmien päivitysprojektien epäonnistumisille ja miten epäonnistumisia voitaisiin välttää. Lisäksi haluttiin löytää vastaus siihen, miten asiakaskohtaisten järjestelmämuutosten kanssa tulisi toimia ja minkälaisia työvälineitä tai standardeja niiden hallintaan on kehitetty.

Toiminnanohjausjärjestelmien päivitysprojektien haasteita ja ongelmia käsitellään tietoperustassa rajaamatta aihetta mihinkään tiettyyn toiminnanohjausjärjestelmään. Haastattelut rajattiin kuitenkin koskemaan vain Microsoft Dynamics AX toiminnanohjausjärjestelmää. Rajauksen syynä oli mahdollisuus haastattelujen osalta tukeutua vuosien varrella rakentuneeseen suhdeverkostoon. Tämän suhdeverkoston kautta oli mahdollista käyttää Microsoft Dynamics AX asiantuntijoita haastateltavina.

2 Toiminnanohjausjärjestelmän päivittäminen

2.1 Miksi päivittää?

Yritysten tulisi kohdella toiminnanohjausjärjestelmäänsä yhtenä kaikkein kriittisimmistä työvälineistä. Järjestelmävalmistajan tuki vanhemmille versioille loppuu jossain vaiheessa, kun uudempia versioita toiminnanohjausjärjestelmästä tulee markkinoille. Onkin hyvin riskialtista jättää liiketoimintakriittinen järjestelmä tuen ulkopuolelle, etenkin kun yleensä maksetaan ylläpitomaksua, joka oikeuttaa uudemman version käyttöön. Pelkät ylläpitomaksut muodostavat huomattavan summan vuosittain ja ovat tyypillisesti 10 - 20 % alkuperäisestä lisenssimaksusta joka vuosi. Tällä summalla saadaan lakien ja asetusten muuttumisesta johtuvia muutoksia, korjauksia ja oikeus päivittää toiminnanohjausjärjestelmä uusimpaan versioon. Sekä Shepherd että Stackpole toteavat, että uusien versioiden myötä voidaan myös saada uusia toiminnallisuuksia, jotka tehostavat yrityksen toimintaa. Versiopäivitykset ovat myös loistava hetki tarkastella tarpeettomia järjestelmämuutoksia ja vanhentuneita ohjelmistoja. (Shepherd 2007, 2; Stackpole 2008.)

Toiminnanohjausjärjestelmän alusta eli palvelimet, käyttöjärjestelmät ja tietokannat ikääntyvät ja vanha versio toiminnanohjausjärjestelmästä yleensä estää niiden päivittämisen. Jos alustaa ei päivitetä, jää se auki mm. tietoturva- ja käytettävyysriskeille. Myös muut edut, kuten parantunut tehokkuus ja raportointi jäävät saamatta, jos alustaa ei päivitetä. Voidaan myös ajautua päivitysprosessiin, kun palvelin päätyy käyttöikänsä päähän ja se joudutaan uusimaan. Uuteen palvelimeen yleensä asennetaan mahdollisimman uusi käyttöjärjestelmä, eikä uusi palvelin välttämättä tue vanhaa käyttöjärjestelmää.

Stackpole (2008) listaa viisi tunnusmerkkiä joiden mukaan on aika päivittää toiminnanohjausjärjestelmä. Toiminnanohjausjärjestelmän tulisi toimia kaiken kriittisen tiedon lähteenä ja siitä pitäisi pystyä saamaa analyysejä ja raportteja, jotka auttavat yritystä tekemään parempia päätöksiä. Jos tämä ei toteudu tai jokin seuraavista tunnusmerkeistä tuntuu tutulta, on luultavasti aika päivittää toiminnanohjausjärjestelmää.

Päivitä toiminnanohjausjärjestelmä, jos se on yli viisi vuotta vanha. Vaikka toiminnanohjausjärjestelmillä ei olekaan viimeistä käyttöpäivää, tulisi viiden vuoden ikään päässeitä järjestelmiä arvioida kriittisesti ja miettiä pystyvätkö ne enää tukemaan tehokkaasti liiketoimintoja. (Stackpole 2008.)

Päivitä kun toiminnanohjausjärjestelmän integraatiot ovat vaikeita toteuttaa. Integraatioiden eli toiminnanohjausjärjestelmän liittäminen kolmannen osapuolen ohjelmistoihin on yksi vaikeimmista asioista, joita päivityksessä tulee vastaan. Jos integraatioiden tekeminen ei ole helppoa, päädytään helposti kalliisiin ja monimutkaisiin asiakaskohtaisiin muutoksiin, joiden hallinnoiminen vie paljon resursseja. (Stackpole 2008.)

Päivitä kun toiminnanohjausjärjestelmästä puuttuu ominaisuuksia ja toimintoja, joita tarvitaan liiketoiminnan tehokkaaseen ohjaamiseen. Uusissa toiminnanohjausjärjestelmänversioissa on vakiona toiminnallisuuksia, jotka puuttuvat vanhemmista versioista ja joiden räätälöinti vanhaan versioon tulisi kalliiksi. (Stackpole 2008.)

Päivitä kun työntekijät, partnerit tai konsultit eivät enää käytä järjestelmää tai eivät ole käytettävissä korjaamaan sitä. Mitä vanhempi versio on käytössä, sitä vaikeampaa on löytää päteviä asiantuntijoita auttamaan vastaantulevissa ongelmissa. Myös sisäiset käyttäjät voivat vältellä iäkkään järjestelmän käyttämistä, jos se ei toimi hyvin Microsoft Officen tai muiden tärkeiden työvälineiden kanssa. (Stackpole 2008.)

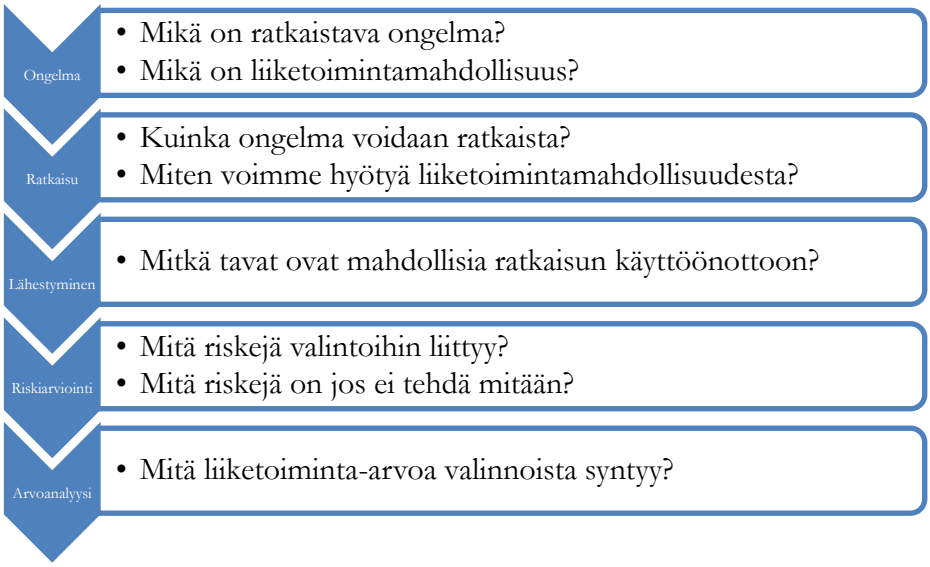
Päivitä kun ilmiselvästi on sen aika, riippumatta siitä tukevatko laskelmat sitä tai eivät. Järjestelmän päivittämisessä kannattaakin käyttää maalaisjärkeä. Toiminnanohjausjärjestelmää tulisikin kohdella kuin infrastruktuuria, jolloin sille on selkeä huoltosuunnitelma. Ei pääomantuottoa lasketa sille, tarvitaanko yrityksessä vettä tai sähköä. Samalla tavalla hyväksytään, että toiminnanohjausjärjestelmä on kannattavan liiketoiminnan edellytys. (Stackpole 2008.)

2.2 Päivitysprojektien epäonnistuminen

Jotta päivitysprojekti voidaan todeta epäonnistuneeksi, on ensin määriteltävä mitä epäonnistuminen tai onnistuminen juuri tuon projektin osalta tarkoittaa. Tarkoittaako projektin budjetin ylittäminen epäonnistumista vai voidaanko katsoa projektin onnistuneen, jos aikataulussa pysytään? Projekteja pitäisikin aina katsoa kokonaisuutena, eikä keskittyä liiaksi yhden mittarin tuloksiin.

Feldmanin mukaan projektin onnistumista tai epäonnistumista mitataan usein kolmella mittarilla: projektin rajauksilla, aikataulussa pysymisellä ja budjetissa pysymisellä. Hän jatkaa, ettei näillä ole kuitenkaan varsinaista merkitystä, koska projekti, jolla ei ole edellytyksiä, tulee lopettaa empimättä. Toisaalta pitää antaa lisää aikaa ja rahaa sellaisille projekteille, joiden tarkan harkinnan jälkeen on todettu olevan sellaisia, jotka ansaitsevat lisäpanostuksia. Tärkeintä onkin, että projekteja seurataan riittävän tarkasti ja että projektihenkilöt uskaltavat ilmoittaa jos huomaavat ongelmia. Kun ongelma huomataan riittävän ajoissa, niin projektin keskeyttäminen ei aiheuta suurta taloudellista menetystä. Keskeyttämisestä voidaan itse asiassa hyötyä kertyneenä kokemuksena. Projektipäälliköiden ei tulisikaan ottaa projektin keskeyttämistä henkilökohtaisena epäonnistumisena, vaan arvioida sitä miksi se keskeytyi ja pyrkiä oppimaan siitä. Ihannetilanne olisi, jos yrityksen organisaatio kannustaisi arvioimaan projekteja kriittisesti ja ”epäonnistumaan” ajoissa, ennen kuin projektissa ollaan niin pitkällä, että sen keskeyttämisestä aiheutuu suuria kustannuksia. (Feldman 2012.)

Osasyy projektin epäonnistumiselle on usein räätälöintien eli asiakaskohtaisten järjestelmämuutosten hallitsematon määrä. Järjestelmämuutosten osalta voidaan miettiä, kannattaako niitä tehdä ollenkaan, sillä järjestelmämuutokset voivat tehdä projektista hyvin kalliin ja pitkäkestoisen. Toisaalta hyvin perustellut ja harkitusti tehdyt järjestelmämuutokset voivat tuoda asiakkaalle kilpailuetua tai kustannussäästöjä. Kysymys ei siis olekaan, kannattaako järjestelmämuutoksia tehdä, vaan miten ne perustellaan. Alla on Beaubouefin käyttämä kaavio (kaavio1), jota voidaan käyttää järjestelmämuutosten perusteluja mietittäessä ja ns. Business Casea tehtäessä. (Beaubouef 2012.)



Kaavio 1. Järjestelmämuutoksen liiketoiminnalliset perustelut (Beaubouef 2012)

Beaubouefin (2012) kaaviossa (kaavio 1) **ongelma**lohkon avulla määritellään ratkaistava asia ja ketkä ongelman ratkaisusta hyötyvät ja miksi se pitäisi ratkaista.

Ratkaisulohko taas käsittelee suoraan ongelman ratkaisua, mutta myös menetelmiä joilla varmistetaan, että ehdotettu järjestelmämuutos todella ratkaisee käsiteltävän ongelman. Lisäksi siinä mietitään tarkoin määriteltyjä hyötyjä joita ratkaisusta syntyy. (Beaubouef 2012.)

Lähestyminen-lohkossa arvioidaan mahdollisia eri tapoja ratkaisun käyttöönottoon. Käytännössä vaihtoehdot voidaan typistää kolmeen eri tapaan: täysi järjestelmämuutos, osittainen järjestelmämuutos ja ei lainkaan järjestelmämuutosta. Täydessä järjestelmämuutoksessa ei hyödynnetä tai hyödynnetään hyvin vähän alkuperäistä toiminnallisuutta. Tässä vaihtoehdossa etuna on järjestelmän suurempi sopeutuminen asiakkaan organisaation tarpeisiin. Haittoina ovat taas muita vaihtoehtoja suuremmat kustannukset ja pidempi käyttöönottoaika. Osittaisessa järjestelmämuutoksessa hyödynnetään alkuperäistä toiminnallisuutta, mutta muokataan sitä tukemaan paremmin asiakasta. Etuina tässä vaihtoehdossa ovat pienemmät kustannukset ja lyhyempi käyttöönotto kuin täydessä järjestelmämuutoksessa. Haittoina ovat lisääntyneet kustannukset ja resurssitarpeet, jotka ovat kuitenkin pienemmät kuin täydessä järjestelmämuutoksessa.

Ei järjestelmämuutosta-vaihtoehdossa ei tehdä lainkaan järjestelmämuutosta, vaan muokataan organisaation toimintamalleja ja -tapoja. Etuina tässä vaihtoehdossa ovat nopea käyttöönotto ja pienet kustannukset. Haittoina ovat taas ratkaisun jäykkyys ja riski, etteivät käyttäjät hyväksy uutta toimintamallia. (Beaubouef 2012.)

Riskiarviointilohkossa määritellään kuhunkin käyttöönottotapaan liittyvät riskit. Käyttöönottotapoihin liittyvien riskien lisäksi arvioidaan tilannetta, jossa ei tehdä mitään ongelman ratkaisemiseksi. (Beaubouef 2012.)

Viimeisessä **arvoanalyysi**lohkossa käsitellään lyhyen ja pitkän tähtäimen vaikutuksia. Jotta ei tehtäisi kauaksi kantavaa päätöstä lyhyen tähtäimen vaikutusten perusteella, on huomioitava ainakin seuraavat kolme asiaa: (Beaubouef 2012.)

- Järjestelmämuutosten vaikutukset tuleviin järjestelmäpäivityksiin ja ylläpitoon
- Vaikutus IT resurssien hankintaan
- Järjestelmämuutosten kumulatiivinen vaikutus.

Kun asiakaskohtaisia järjestelmämuutoksia arvioidaan huolellisesti käyttäen hyväksi järjestelmämuutosten liiketoiminnallisia perusteita voidaan olla huomattavasti varmempia, ettei projekti epäonnistu niiden takia. Lisäksi näin pystytään myös vähentämään tulevien järjestelmäpäivitysten kustannusyllätyksiä, joita seuraa, mikäli ei tiedetä tarkkaan, mitä on tehty ja miksi.

Herrig ja Scanlon (2012) puolestaan listaavat seuraavat kymmenen syytä, miksi toiminnan ohjausjärjestelmän päivitysprojekti epäonnistuu:

1. Ei kerrota käyttäjille mitä uusi järjestelmä tarkoittaa heille. Varmin tapa saada päivitysprojekti epäonnistumaan, on pitää käyttäjät pimennossa. Vaikka päivitysprojekti onnistuisi teknisesti, voivat käyttäjät hyljeksiä sitä, jos he eivät ota sitä omakseen.
2. Ei tehdä järjestelmälle rasitustestejä. Järjestelmää tulisi rasittaa oikeilla käyttäjillä, mutta myös ohjelmallisesti, jotta voidaan varmistua, että se toimii myös käyttöönottotilanteessa.

3. Ei suoriteta kenraaliharjoitusta. Jos käyttöönotto on tarkoitus pitää viikonloppuna, tulisi kenraaliharjoituskin pitää viikonloppuna, jotta voidaan varmistua myös muista järjestelmän ulkopuolista, mutta kriittisistä asioista, kuten ovien aukiolo tai tietokantavarmistukset.
4. Ei oteta muutoshallintaa ja testausta vakavasti. Yksittäin järjestelmämuutos voi vaikuttaa moneen järjestelmän osaan ja yleensä järjestelmämuutoksia tuodaan kerralla useita. Tämä vain korostaa huolellisen testauksen tarvetta.
5. Vain asiakkaalla on projektipäällikkö. Moni asiakas ajattelee säästävänsä paljon rahaa tällä tavoin. Toimittajan projektipäälliköillä on kuitenkin paljon aikaisempaa kokemusta päivitysprojekteista.
6. Ei viestitä muutoksista etukäteen. Loppukäyttäjät eivät pidä muutoksista, koska ne aiheuttavat heille ylimääräistä työtä ja sen takia he mieluummin pysyvät vanhan järjestelmän parissa.
7. Järjestetään vain ”perinteistä” luokkahuonekoulutusta. Useilla käyttäjillä on useita erilaisia tehtäviä. Enemmän tehtäviä = enemmän koulutusta = enemmän työtä = vähemmän aikaa perheen parissa. Tämän takia luokkahuonekoulutus ei yksinään toimi. Kannattaa täydentää luokkahuonekoulutusta videokoulutuksilla tärkeimmistä liiketoimintaprosesseista, joita voi katsoa tarvittaessa omalta koneeltaan milloin tahansa.
8. Ei siirrytä suljetusta järjestelmästä avoimiin standardeihin. Avointen standardien käyttäminen nopeuttaa päivitysprosesseja jatkossa. Näin tapahtuu etenkin liittymien ja raporttien osalta.
9. Ei käsitellä tietoturvakysymyksiä etukäteen. Päivitysprojektin aikana otetaan yleensä käyttöön uusia ominaisuuksia, ja mitä isommaksi järjestelmä kasvaa, sitä haavoittuvaisemmaksi se tulee. Kannattaakin mahdollisuuksien mukaan rajoittamaan turhaa pääsyä järjestelmään tai sen tietoihin.
10. Oletetaan omien teknisten henkilöiden oppivan 15 vuoden asiat parissa viikossa. Päivitysprojekteja ei ole joka päivä tai joka vuosi. Tämän takia on tärkeää käyttää päivitysprojektissa kokeneita teknisiä konsultteja, jotka pitävät järjestelmät toiminnassa koko päivityksen ajan.

2.3 Toiminnanohjausjärjestelmän päivittämisen menestystekijät

Toiminnanohjausjärjestelmän päivitysprojektia tulee ohjata ja johtaa, yhtä huolella kuin mitä tahansa toiminnanohjausjärjestelmän käyttöönottoprojektia. Päivitysprojektissa yritykseen on todennäköisesti kertynyt aikaisemmasta käyttöönottoprojektista osaamista, jota voidaan hyödyntää. Jo tarjouspyyntövaiheessa tulisi ilmaista, että työmääräarvioiden pitää olla realistisia ja että kunnon työstä maksetaan kunnon hinta. Näin mahdollistetaan se, että toimittaja tuo projektiin parhaat resurssit, jotka mahdollisuuksien mukaan kannattaa sitouttaa projektiin, vaikkapa sopimalla kirjallisesti käytettävistä henkilöresursseista. Tällöin projektin laatu paranee ja kustannukset pysyvät kurissa ja projektin aikataulu pitää. (Myllymäki, Hintikka, Dahlberg & Uimonen 2010, 254.)

Taulukko 1. Keskimääräisiä vuosipalkkoja Microsoft Dynamics AX tehtävistä Suomessa (Nigel Frank International 2012)

Tehtävä	Junior	Mid	Senior
Tekninen konsultti	41 478 €	50 342 €	58 003 €
Toiminnallinen konsultti	39 443 €	48 663 €	54 435 €
Sovelluskehittäjä	36 112 €	48 611 €	56 454 €
Tekninen arkkitehti	47 531 €	52 664 €	65 469 €
Ratkaisuarkkitehti	48 855 €	53 795 €	67 946 €

Taulukossa 1 (Nigel Frank International 2012) on keskimääräisiä vuosipalkkoja eri Microsoft Dynamics AX tehtävistä Suomessa. Siinä on myös kolmiportainen kokemukseen perustuva jaottelu. Taulukosta voidaankin saada viitteitä siitä, miten vähän kokeneemman senioriosaajan kiinnittäminen projektiin maksaa verrattuna junioriosaajaan. Senioriosaajan kokemus ja tietotaito ovat moninkertaisia verrattuna junioriosaajaan, jonka pitää opetella koko ajan uusia asioita. Yritykselle paras kustannus/hyötysuhde saavutetaan, kun päivitysprojektissa on muutama senioriosaaja pitämässä huolen projektin etenemisestä ja valvomassa ja auttamassa juniorikonsultteja. Nämä projektille elintärkeät resurssit ovat juuri niitä joita yrityksen tulisi pyrkiä kiinnittämään projektille, jotta sen laatu ja aikataulu saadaan pysymään (Myllymäki ym. 2010, 254).

Tietojärjestelmän hankinta-käsikirjassa (Talentum 2005, 16) listataan joukko tietojärjestelmäprojektin menestystekijöitä, jotka pätevät myös päivitysprojekteihin.

- johdon tuki
- asiakkaan ja loppukäyttäjän sitoutuminen ja palaute
- osaavat ja motivoituneet tekijät
- realistiset tavoitteet
- selkeä vaatimusmäärittely
- riittävä seuranta ja ohjaus.

Tämä listaus ei ole kattava, vaan sitä voidaan pitää esimerkkeinä niistä asioista, joihin tulisi kiinnittää enemmän huomiota. Voidaan myös ajatella, että jos jossain näistä listatuista kohdista epäonnistutaan kunnolla, on päivitysprojektilla suuri mahdollisuus epäonnistua. Epäonnistumisella tarkoitetaan tässä yhteydessä budjetin ylittämistä, aikataulun pettämistä ja sitä, ettei päivityksellä saada odotettuja parannuksia yrityksen toimintoihin. On myös mahdollista, ettei päivitystä koskaan saada tuotantokäyttöön asti.

Berg (2013) listaa viisi askelta onnistumiseen, jotka ovat hyvin samankaltaisia kuin Tietojärjestelmän hankinta-käsikirjassa (Talentum 2005, 16) listatut menestystekijät. Hän vertaakin tietojärjestelmähanketta talonrakennusprojektiin, jossa perustukset pitää olla kunnossa ennen kuin itse taloa voi lähteä rakentamaan niiden päälle.
Bergin (2013) listaamat viisi askelelta ovat:

- **Selkeä tavoite.** Kun kaikilla on samanlainen ja selkeä näkemys siitä mitä ollaan tekemässä ja miksi, helpottuu onnistuminen huomattavasti. Jonkun täytyy lisäksi pitää huolta kokonaisarkkitehtuurista eli siitä, mitä palasia mihinkin kohtaan asetetaan ja miten ne toimivat yhdessä unohtamatta ympäröivää todellisuutta eli integraatioita.
- **Johdon sitoutuminen**. Johdon on avoimesti tuettava projektia, myös kick-off – tilaisuuden jälkeen. Projektin aikana johdon on oltava mukana tukemassa projektipäällikköä ja tekemässä strategisia valintoja. Kehityskohteille, joiden halutaan saavan rahoitusta ja menevän eteenpäin, on nimettävä omistaja, joka pystyy edesauttamaan kehityskohteen edistymistä.

- **Realistinen aikataulu**. Laaditaan aikataulut todellisten resurssien mukaan, jotka ovat aidosti irrotettavissa mukaan projektiin. Myös sopimusneuvottelut voivat olla yllättävän pitkiä ja viedä enemmän aikaa kuin alun perin oli ajateltu. Jos projektin laajuus matkanvarrella muuttuu, päästäänkö muuttamaan myös aikataulua ja budjettia vastaavasti? Kaikissa projekteissa tulee aina yllätyksiä, joten niihin tulee varautua.
- **Selkeät roolit ja vastuut**. Oikeitten ihmisten tulee tehdä oikeita asioita. On myös tärkeää kirjata selkeästi mikä kuuluu järjestelmätoimittajan tehtäviin ja mikä kuuluu asiakkaan tehtäviin. Kun projektipäällikkönä on ammattilainen, joka osaa johtaa projektia helpottaa se projektin onnistumista. Myös projektin jälkeisestä ylläpidosta on sovittava ajoissa.
- **Kommunikaatio**. Asia joka usein jää tekemättä, on muutoksista tiedottaminen. Ei olekaan sellaista asiaa kuin liikatiedotus ja vaikka miten asioista tiedottaa, niin silti löytyy aina joku, joka ei niistä ole kuullut. Mietittäviin asioihin kuuluukin kuka projektin asioista tiedottaa ja millä tavalla. Projektipäällikkö ei välttämättä ole paras henkilö hoitamaan tiedotusta muun projektijohdon ohella.

Toiminnanohjausjärjestelmän päivitysprojekti on yleensä kevyempi ja edullisempi kuin käyttöönottoprojekti, mutta siihen liittyy aivan samoja liiketoimintariskejä ja aikataulujen ja budjetin ylityksiä. Päivitysprojektia tuleekin ohjata samoilla prosesseilla kuin käyttöönottoprojektia, vaikka hieman kevyemmin. Päivittäessä tulee huomioida ainakin seuraavat kolme asiaa: (SearchManufacturingERP 2010.)

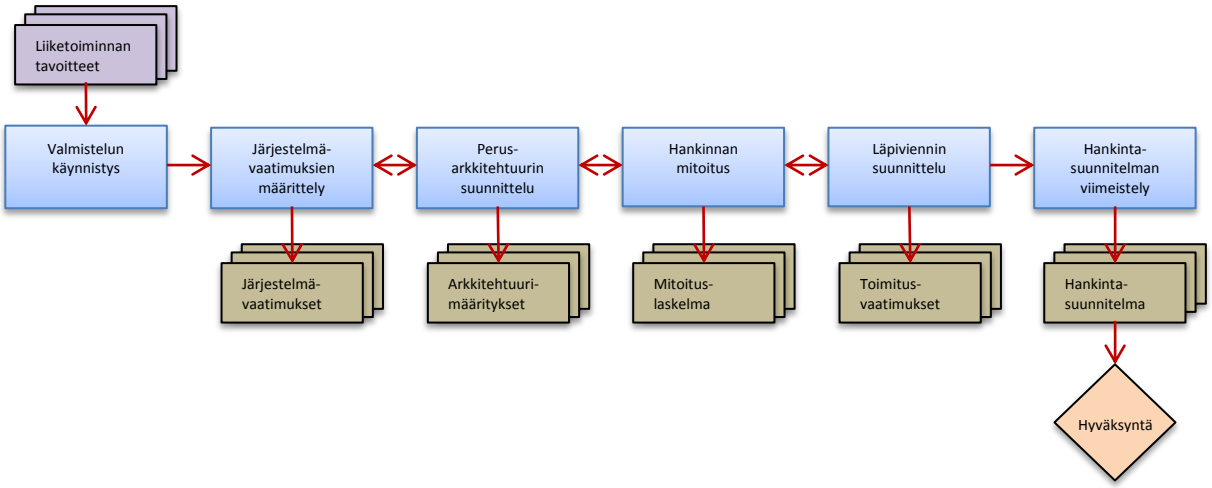
Ensiksi, älä ota käyttöön uutta versiota ensimmäisten joukossa, ainakaan ilman että otetaan tietoinen riski. Yleensä kannattaa odottaa, että järjestelmävalmistaja julkaisee ensimmäisen korjauspaketin ns. Service Packin. On myös järkevää selvittää muiden asiakkaiden kokemuksia ko. versiosta ennen siihen päivittämistä. (SearchManufacturingERP 2010.)

Toiseksi, ei kannata aliarvioida järjestelmämuutosten uuteen versioon siirtämiseen kuluvaa aikaa ja rahaa. Ei myöskään kannata kuvitella, että uuden version uudet toiminnallisuudet automaattisesti korvaisivat vanhat järjestelmämuutokset. On todennäköistä, että järjestelmävalmistaja on uuden version myötä muuttanut alla olevan tietokannan rakennetta. Tästä syystä vanhojen järjestelmämuutosten ”pieni” siirtäminen uuteen versioon voi muuttua isoksi ohjelmointiprojektiksi. Sama koskee myös vanhan version liittymiä ja liittymälogiikkaa. Kun tietokanta ja sitä käyttävä logiikka muuttuvat, joudutaan helposti rakentamaan myös liittymät täysin uudestaan. (SearchManufacturingERP 2010.)

Kolmanneksi, testausta ei kannata unohtaa. Kannattaa testata, kuin oltaisiin aivan uuden käyttöönoton edessä, sillä käytännössä kyse on samasta asiasta. Kaikki liiketoimintaprosessit pitää testata hyvin ja ennen kaikkea ne pitää testata loppukäyttäjien toimesta. Näin siitä huolimatta ovatko käyttöliittymä tai keskeiset toiminnot muuttuneet. Käyttäjiä kannattaa tukea testitapauksilla ja – prosesseilla, jotta testaus sujuu helposti. Tulee kuitenkin muistaa antaa testaukselle aikaa, myös uudelleentestaukselle. (SearchManufacturingERP 2010.)

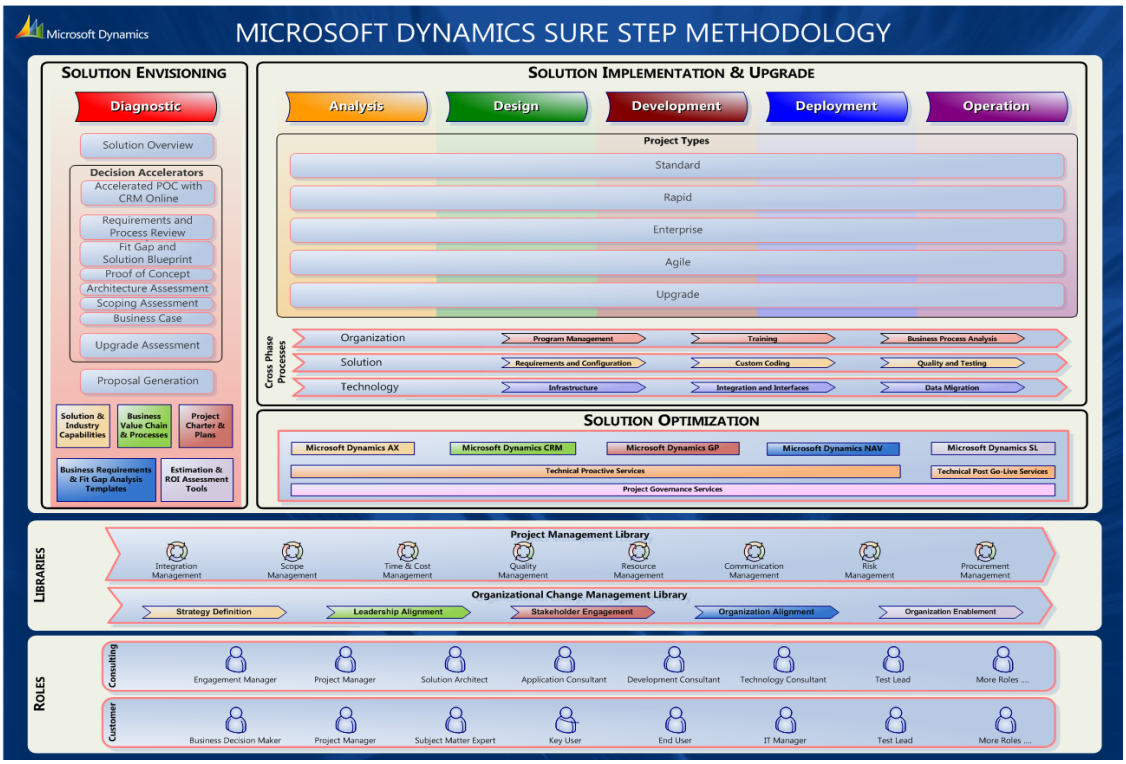
Toiminnanohjausjärjestelmän päivitysprojektiin voidaan melko hyvin soveltaa samoja perusasioita kuin varsinaiseen käyttöönottoprojektiinkin. Näin on etenkin silloin, kun käyttöönotosta on kulunut pitkä aika eikä välissä toiminnanohjausjärjestelmää ole päivitetty. On varsin perusteltua käydä läpi hankintaprosessi, vaikka nykyinen toiminnanohjausjärjestelmä olisikin toiminut hyvin. Kannattaa arvioida myös muita markkinoilla olevia sopivia toiminnanohjausjärjestelmiä käytössä olevan lisäksi.

Tietotekniikan liiton 4V-malli soveltuu hyvin toiminnanohjausjärjestelmän hankintaan. 4V-mallissa vaiheita on nimensä mukaisesti neljä: valmistelu, valinta, valvonta ja viimeistely.



Kaavio 2. Hankinnan valmistelu (Talentum 2005, 21)

Valmistelussa nimensä mukaisesti valmistellaan hankintaa ja luodaan ja annetaan sille tarvittavat resurssit. Kaavio 2 on hankinnan valmistelusta, jossa lähdetään liikkeelle liiketoiminnan tavoitteista ja päädytään hankintasuunnitelman hyväksyntään. Valmisteluprosessin aikana myös määritellään järjestelmävaatimukset, suunnitellaan perusarkkitehtuuri, mitoitetaan hankinta ja suunnitellaan läpivienti. **Valinnassa** valitaan, joko tarjouskilpailussa tai muilla aikaisemmassa vaiheessa asetetuilla kriteereillä toiminnanohjausjärjestelmä ja sen toimittaja. **Valvonta** pitää sisällään hankintaprosessin seurannan ja laadun varmistamisen. Siinä annetaan työkaluja, kuten ohjausryhmä ja projektiryhmät, joilla seurataan päivitysprojektin etenemistä ja laadun toteutumista. **Viimeistely-**vaiheessa kootaan yhteen tulokset ja tarkistetaan, että kaikki hankitut osat on toteutettu. Siinä myös kerätään kokoon kokemuksia ja dokumentoidaan ne tulevia projekteja varten. Näin saadaan organisaatiolle arvokasta kokemusta. (Talentum 2005, 9-11.)



Kaavio 3. Sure Step metodologian yleiskuva (Microsoft a)

Päivitysprojektin onnistumista edesauttaa jos järjestelmätoimittajalla ja asiakkaalla on yhteisesti sovittu projektimetodologia käytössä. Tällöin myös projektihenkilöt tietävät paremmin mitä heiltä odotetaan ja mistä he saavat tarvitsemansa tiedot. Esimerkkinä tällaisesta metodologiasta voidaan mainita Microsoftin Sure Step (kaavio 3). Sure Step on virallinen metodologia Microsoft Dynamics-tuotteiden käyttöönottoon ja päivittämiseen. Se kertoo, mitä kukin tekee ja missä järjestyksessä, sekä kuka vastaa mistäkin. Sure Step tukee useita eri projektityyppejä, joita voi tarvittaessa yhdistellä. Projektityyppejä ovat Standard, Rapid, Enterprise, Agile ja Upgrade. (Microsoft a.)

Tämä metodologia lähtee jo myyntivaiheesta eli tilanteesta ennen kuin asiakas on tehnyt päätöksen toiminnanohjausjärjestelmän hankinnasta tai päivittämisestä. Myyntivaihetta Sure Step metodologiassa kutsutaan Diagnostic-vaiheeksi. Muita vaiheita ovat Analysis, Design, Development, Deployment ja Operation. Näistä vaiheista Operation on käyttöönoton jälkeinen vaihe. Sure Step sisältää myös prosesseja jotka ulottuvat usean vaiheen yli. Metodologian etuina ovat sen tarjoamat valmiit dokumentit ja toimintatavat Microsoft tuotteiden käyttöönottoon ja päivitykseen. Sen selkeänä haittana ainakin Suomessa on suomenkielen puute eli Microsoft ei toimita dokumentaatiota tai työkaluja suomeksi, vaikka muutamilla muilla kielillä ne ovat saatavilla. (Microsoft a.)

3 Järjestelmämuutokset

Päivitettäessä toiminnanohjausjärjestelmää kannattaa harkita huolellisesti, onko mahdollista olla siirtämättä asiakaskohtaisia järjestelmämuutoksia ja ottaa käyttöön vakiojärjestelmä. Tarvittaessa järjestelmämuutoksia voidaan tehdä myös käyttöönoton jälkeen. Ajatus siitä, että toiminnanohjausjärjestelmä on muuttumaton eikä sille voi tehdä mitään käyttöönoton jälkeen, alkaa vähitellen väistyä. Päivitysstrategia, jossa alussa otetaan käyttöön vakiojärjestelmä eikä siirretä järjestelmämuutoksia, toimii etenkin silloin, jos on mahdollista ottaa käyttöön liiketoimintaa, vakiojärjestelmää vielä paremmin tukeva vertikaali eli toimialaratkaisu. (Weiss 2011.)

Järjestelmämuutokset voidaan jakaa karkeasti kahteen ryhmään. Ensimmäiseen ryhmään voidaan sijoittaa yleisemmät muutokset ja ns. vertikaalit eli sellaiset kolmannen osapuolen tekemät muutokset, jotka voidaan hankkia valmiina ja joiden päivityksestä kolmas osapuoli huolehtii. Toiseen ryhmään tulevat ne muutokset, jotka on tehty vain asiakkaan tarpeisiin ja yleensä asiakkaan omista liiketoiminnallisista lähtökohdista ponnistaen. Näitä järjestelmämuutoksia kutsutaan yleisesti räätälöinneiksi.

Vertikaalit ja muut kolmannen osapuolen tuotteistamat järjestelmämuutokset ovat yleensä saatavilla myös uusimpaan järjestelmäversioon. Vertikaalien etuna onkin niiden asiakaskohtaisia järjestelmämuutoksia helpompi päivitettävyys, ja se että ne on yleensä suunnattu jollekin tietylle alalle, jolloin niitä voidaan muokata jonkin verran asetusten kautta (Weiss 2011). Kysymyksiä aiheuttaa lähinnä miten muutokset ja vertikaalit on lisensoitu eli maksetaanko niistä ylläpitomaksua, joka oikeuttaa viimeisimmän version käyttöön vai ovatko ne olleet kertamaksulla hoidettuja, jolloin uudemman version käyttäminen vaatisi käytännössä ko. vertikaalin uudelleenostamista. Toinen harkittava asia on, tarvitaanko vanhoja muutoksia enää. Voi olla, että toiminnanohjausjärjestelmän uudet ominaisuudet tekevät vanhan vertikaalin ainakin osittain tarpeettomaksi. Voi myös olla, että päivityksen yhteydessä harkintaan tulee uusien vertikaalien käyttöönotto, jos niillä voidaan välttää asiakaskohtaisten järjestelmämuutosten tekeminen tai niiden päivittäminen uuteen versioon.

Kannattaako asiakaskohtaisia järjestelmämuutoksia tehdä, on kysymys, joka nousee helposti mieleen pohdittaessa kaikkia mahdollisia haittoja niistä. Suurin osa toiminnanohjausjärjestelmistä on suunniteltu sopimaan keskimääräiselle asiakkaalle. Jotta toiminnanohjausjärjestelmä sopii asiakkaan liiketoimintaprosesseihin, sitä pitää muokata. Tässä piilee kuitenkin ansa, sillä jokainen muutos järjestelmään vie sen vähän kauemmaksi siitä, mitä se oli alun perin. Sillä ei kuitenkaan ole merkitystä ensi alkuun, mutta ajan kuluessa voidaan joutua ratkaisemaan ongelmia, jotka johtuvat liiketoimintaprosesseja varten tehdyistä järjestelmämuutoksista. Babic listaakin seuraavat seitsemän syytä miksi järjestelmämuutokset ovat huonoja ideoita: (Babic 2009.)

Regressiotestaus. Kun järjestelmään tehdään jokin pieni muutos, on pakko tehdä regressiotestausta, jotta voidaan varmistua, ettei mikään aikaisemmin luotu ole mennyt samalla rikki. Eli toisin sanoen kun toiminnallisuuteen A tehdään pieni muutos, niin samalla toiminnallisuus B lakkaa toimimasta toivotulla tavalla. Mitä tahansa muutoksia tehdäänkin toiminnanohjausjärjestelmien kanssa, aina on entuudestaan paljon ohjelmistokoodia. Mitä enemmän järjestelmämuutoksia tehdään, sitä enemmän regressiotestausta on pakko tehdä ja mitä konsultit eivät löydä, loppukäyttäjät löytävät kyllä käyttöönoton jälkeen. (Babic 2009.)

Käyttöönotto. Järjestelmämuutokset viivästyttävät käyttöönottoa, sillä mitä enemmän niitä on, sitä pidempään käyttöönottovaiheeseen pääsy kestää. Vaikka budjetissa saattaa olla huomioitu kaikki järjestelmämuutoksista aiheutuneet kustannukset, mutta silti aikatauluylitykset saattavat tehdä projektista epäonnistuneen. Toiminnanohjausjärjestelmien kanssa käyttöönottopäivällä on väliä. On helpompaa ja halvempaa vaihtaa järjestelmää tilikauden vaihteessa. (Babic 2009.)

Järjestelmävalmistajan tuki. Ongelman ilmetessä saatetaan tarvita tukea järjestelmävalmistajalta. Näin toimittaessa järjestelmävalmistaja haluaa ensimmäisenä tietää esiintyykö ongelma myös vakioympäristössä, ilman järjestelmämuutoksia. Mikäli ongelmaa ei kyetä toistamaan ilman järjestelmämuutoksia, ei järjestelmävalmistaja kykene avustamaan ongelman ratkaisussa. Järjestelmävalmistajat vastaavat vain vakiotuotteestaan, eivät siihen tehdyistä järjestelmämuutoksista.

Pienten ongelmien kohdalla tämä ei välttämättä ole merkityksellistä, mutta kriittisen, liiketoiminnan pysäyttävän ongelman sattuessa kohdalle, myös se joudutaan selvittämään ilman järjestelmävalmistajan tukea. (Babic 2009.)

Päivitys. Järjestelmävalmistaja julkaisee toiminnanohjausjärjestelmän uusia versioita kahden kolmen vuoden välein. Päivittäminen uuteen versioon saattaa pahimmillaan tarkoittaa kaikkien järjestelmämuutosten uudelleen kehittämistä. Käytännössä tämä tarkoittaa sitä, että joudutaan maksamaan jo tehdyistä järjestelmämuutoksista joka version kohdalla uudestaan. Ei myöskään saa unohtaa regressiotestausta, jota joudutaan myös tekemään uudestaan joka muutoksen kohdalla. Tietenkin on mahdollista olla päivittämättä, mutta se tarkoittaa, että jättäydytään tarkoituksella kehityksestä ja sementoidaan koko infrastruktuuri ohjelmistoineen ja laitteistoineen vanhoihin versioihin. (Babic 2009.)

Tietotaito. Ihmisiä tulee ja menee, mutta niin tulee ja menee myös osaaminen heidän mukanaan. Mikäli käytössä on vakio toiminnanohjausjärjestelmä, voidaan palkata yrityksen ulkopuolelta osaajia. Mikäli käytössä on taas voimakkaasti muokattu versio, niin vakioversion osaamisesta ei ole juurikaan apua. Ongelma korostuu erityisesti osaajien lähtiessä yrityksestä, jolloin yritykseen jää osaamisvaje, jota voi olla hyvin vaikea paikata. Sama koskee myös järjestelmätoimittajan konsultteja. Henkilöt, jotka ovat olleet mukana kehittämässä järjestelmää järjestelmätoimittajalla, saattavat lähteä ja uusia tulla tilalle. Uusien henkilöiden on kuitenkin vaikea antaa tukea ja ylläpitää järjestelmää, vaikka olisivat sertifioituja osaajia. (Babic 2009.)

Järjestelmätoimittajaan lukittuminen. Vakioidun toiminnanohjausjärjestelmän valitsemalla vältetään järjestelmätoimittajaan lukittuminen. Mikäli järjestelmätoimittaja ei toimi tyydyttävällä tavalla, järjestelmätoimittajan voi vaihtaa toiseen. Mikäli toiminnanohjausjärjestelmää on voimakkaasti muokattu, saattaa järjestelmätoimittajan vaihtaminen osoittautua vaikeaksi ja kalliiksi. Järjestelmätoimittajat ovat myös hyvin varovaisia hyväksyessään ylläpitonsa piiriin sellaisia muutoksia, joita eivät ole itse kehittäneet. Heidän on pakko selvittää ja opetella jotain epästandardia ja sen jälkeen ylläpitää sitä ja se saattaa olla liian riskialtista tai kallista. (Babic 2009.)

Ohjeet. Usein järjestelmätoimittajan konsulteilta jää päivittämättä kätevät F1 ohjepainikkeet ohjetiedostot viimeisimmillä järjestelmämuutoksilla. He yleensä toimittavat järjestelmämuutosten käyttöohjeet tai loppukäyttäjädokumentoinnin, mutta niiden hyöty jää vähäiseksi. Käyttäjät eivät kykene erottamaan, mikä on vakiotoiminnallisuutta ja mikä on järjestelmämuutosta. Ohje voi löytyä esimerkiksi asiakasrekisteristä, mutta ei eräseurannasta tai ohje kertoo vain osan asiakasrekisterin toiminnoista tai kentistä tai kenties kertoo niistä jopa väärin. Tällöin käyttäjät lakkaavat käyttämästä ohjetoimintoa. Kun ohjetoiminnon käyttäminen loppuu, loppuu myös vähitellen omaehtoinen ajattelu ja käyttäjistä tulee robotteja, jotka vain seuraavat pilkun tarkasti heille annettuja ohjeita. Tästä seuraa virheitä, virheiden korjausta ja menetettyä tehokkuutta. (Babic 2009.)

Kun järjestelmämuutoksia katsotaan tältä näkökannalta, näyttää siltä että järjestelmämuutokset aiheuttavat vain pitkäkestoisia kustannuksia ja kasvaneita riskejä. Asia ei kuitenkaan ole aivan näin mustavalkoinen. Toisinaan järjestelmämuutokset ovat täysin välttämättömiä liiketoiminnalle, eikä ilman niitä tulla toimeen. Edellä mainitut seitsemän syytä eivät siis tarkoita, että kaikkia järjestelmämuutoksia tulisi aina välttää, vaan pikemminkin niin, että niitä tehdään harkitusti ja mietitään myös vaihtoehtoisia toteutustapoja. Kaikkien järjestelmämuutosten takana tulee kuitenkin olla liiketoiminnallinen syy eli niillä tulisi olla ns. Business Case. Lisäksi niiden tulisi kulkea projektin johdon kautta, jolloin hyväksytään myös niiden projektille tuomat kustannusvaikutukset sekä resurssi- ja aikatauluvaateet. Mikäli järjestelmämuutoksia päädytään tekemään, tehdään ne riskit tiedostaen ja niihin varautuen. Järjestelmämuutoksen liiketoiminnallisia perusteluja arvioitaessa apuna voidaankin käyttää Beaubouefin luomaa kaaviota (kaavio1). Kaaviossa lähestytään asiaa tarpeen eli ongelman kautta, päätyen siihen mitä liiketoiminnallista arvoa järjestelmämuutos toisi. (Babic 2009; Beaubouef 2012; Phillips 2010.)

Asiakaskohtaisten muutosten mahdollinen päivittäminen on järjestelmätoimittajan tehtävä ja asiakkaalle jääkin päätettäväksi sopivimman päivitysstrategian valitseminen. Valinnassa voi olla hyödyllistä käyttää joko järjestelmätoimittajaa tai ulkopuolista konsulttia. Ulkopuolisen konsultin käyttämistä puoltaa järjestelmätoimittajan mahdollinen puolueellisuus omien tekemistensä suhteen. Toisaalta ulkopuolisen konsultin käyttäminen vaatii aikaa ja tuo päivitysprojektille lisää kustannuksia.

3.1 Järjestelmämuutosten päivittäminen / siirtäminen vanhasta uuteen versioon

Päivitysprojektin alussa kannattaa miettiä tarkkaan, millä strategialla päivitykseen ryhdytään. Mennäänkö miltei vakiojärjestelmällä, jossa on uusia toiminnallisuuksia vai odotetaanko, että kaikki halutut muutokset on saatu tehtyä ja testattua. Lech ja Robertson muistuttavat 80/20 säännöstä. 80/20 sääntö sanoo, että 80 prosenttia tuloksista saavutetaan 20 prosentilla työstä. Näin ollen voi olla järkevää ottaa järjestelmä käyttöön, kun 80 prosenttia muutoksista on tehty ja jatkaa loppujen tekemistä järjestelmän käyttöönoton jälkeen. Näin päivityksen takaisinmaksuaika lyhenee huomattavasti ja uudet toiminnallisuudet saadaan aikaisemmin käyttöön. Jotta sääntö toimisi, vaatii se kuitenkin, että tekemistä kohdennetaan liiketoimintakriittisiin kohtiin. (Lech 2004; Robertson 2011.)

Järjestelmämuutokset pitää ensin kartoittaa eli yksinkertaisesti listata kaikki tehdyt muutokset ja niiden käyttötarkoitus (Oracle 2012, 8). Tämä voi olla huomattavan vaikeaa, sillä muutosten dokumentaatio on usein puutteellista ja henkilöstö on vaihtunut ajan kuluessa. Voi olla, että ainoa mahdollinen tapa kartoittaa muutoksia on tehdä se koodia läpikäymällä. Tämä on taas raskas ja melko kallis lähestymistapa. Mikäli muutoksia on paljon ja niistä ei ole saatavilla kunnon dokumentaatiota, kannattaa harkita puhdasta uutta käyttöönottoa ja käyttää vanhaa järjestelmää ainoastaan mallina ja inspiraation lähteenä.

Kun muutokset on kartoitettu, tulee niille suorittaa uudelleenarviointi, jossa tarkastellaan kriittisesti, onko niille vielä liiketoimintatarvetta (Oracle 2012, 8). Arvioinnissa on hyödyllistä pitää mukana niitä ihmisiä, jotka töitä käytännössä myös tekevät. On mahdollista, että jotkut muutokset ovat sellaisia, joita ei ole koskaan käytetty tai joista on ollut jopa suoranaista haittaa.

Muutosten uudelleenarvioinnin jälkeen niitä pitää arvioida uuden version uutta toiminnallisuutta vasten, jotta saadaan selville, mitkä voidaan korvata uudella toiminnallisuudella (Oracle 2012, 8). Tätä arviointia ei asiakas käytännössä pysty tekemään yksin, vaan se on tehtävä yhteistyössä järjestelmätoimittajan kanssa. On myös erittäin suositeltavaa käydä samalla läpi yrityksen liiketoimintaprosesseja ja miettiä, voiko niitä ohjata enemmän uuden ohjelmistoversion vakiotoiminnallisuuksien suuntaan. (Gibbons 2012, 9).

Samalla kun järjestelmämuutosten uudelleenarviointia tehdään, kannattaa niitä jakaa ensin niiden päivittämisen vaatiman työmäärän mukaan kolmiportaisesti pieniin, keskisuuriin ja suuriin. Sitten samat muutokset kannattaa jakaa niiden liiketoimintakriittisyyden perusteella samalla tavalla kolmeen ryhmään. Näin järjestelmämuutoksista on helpompi karsia pois ne, jotka eivät ole liiketoimintakriittisiä ja jotka toisaalta aiheuttaisivat runsaasti kustannuksia päivityksessä. Malli on karkea ja vaikka joissain tapauksissa onkin järkevämpää jakaa muutoksia useampiportaisella asteikolla, niin tämä on melko helppo työväline ottaa käyttöön.

3.2 Parhaat toimintatavat ja standardit

Päivitysstrategiat voidaan jakaa kolmeen ryhmään. Ensimmäiseen ryhmään tulevat ne, joissa noudatetaan ohjelmistotoimittajan päivityspolkua ja joissa nostetaan kaikki tehdyt asiakaskohtaiset muutokset sellaisinaan uuteen versioon. Toiseen ryhmään kuuluvat ne, joissa päivityspolkua ei ole tai asiakaskohtaisten muutosten määrä on niin suuri, ettei ole kustannustehokasta päivittää niitä, vaan tehdään uusi käyttöönotto uudella versiolla. Kolmanteen ryhmään kuuluvat sijoittuvat kahden aikaisemman vaihtoehdon välille. Tällaisessa hybridimallissa on tyypillistä, että ohjelmistomuutoksia joudutaan arvioimaan ja miettimään, mitä kannattaa siirtää ja mitä ei. Keskeisintä on arvioida kustakin vaihtoehdosta syntyvät kustannukset ja verrata niitä saataviin hyötyihin.

Ensimmäiseen vaihtoehtoon liittyy riski päätyä käyttämään uutta versiota vanhalla ja kenties tehottomalla tavalla. Mikäli kuitenkin nähdään, että ohjelmistomuutokset ovat järkeviä tai niitä on hyvin vähän eivätkä mene päällekkäin uusien toimintojen kanssa, voi tämä vaihtoehto olla asiakkaalle helpoin. Malli jossain määrin edellyttää, ettei ihmisten tapaan tehdä töitä tule muutoksia. Siihen liittyy paljon työtä, joka kohdistuu kuitenkin lähinnä järjestelmätoimittajaan. Asiakkaan kannalta etuna on selkeästi pienempi oma työmäärä ja pienempi koulutuksen tarve prosessien pysyessä lähes samoina. Ratkaisuna 1ClickFactory tarjoaa omaa toimintamalliaan vaihtoehdoksi järjestelmätoimittajien yksinään tekemille päivityksille. Tässä toimintamallissa 1ClickFactory tekee koodin alustavan arvioinnin ja varsinaisen päivityksen sekä koodin että datojen osalta. Etuna on, että koodin päivitystä tekee taho, joka on keskittynyt vain siihen. Järjestelmätoimittaja tekee kuitenkin aina varsinaisen asennuksen ja järjestelmäasetusten laittamisen eli parametroinnin. (1ClickFactory; Oracle 2012, 4.)

Uudella käyttöönotolla voidaan ehkäistä tilanteita, joihin liittyy vanhan version riskejä. Näissä tilanteissa on usein järkevää aloittaa alusta ja tehdä uusi käyttöönotto. Tilanteita ovat mm. isot ohjelmistomuutokset, joiden takia on jääty vanhaan versioon, jolle ei ole saatavissa virallista tukea tai päivityspolku uuteen versioon on työläs ja kallis. Liiketoimintaa, jolle järjestelmä alun perin oli hankittu, ei ole enää ja nykyisille liiketoiminnoille se ei ole toimivin vaihtoehto. Malli mahdollistaa myös aikaisemmassa käyttöönotossa tehtyjen virheiden tai vajavuuksien korjaamisen (Oracle 2012, 4). Tällöin on etuna myös aito mahdollisuus tarkastella nykyistä toiminnanohjausjärjestelmän valintaa. On sanottu, että on helpompaa lähteä uudella käyttöönotolla kuin päivittää, jos käytössä olevan version jälkeen on julkaistu kolme tai useampi iso päivitys. (Gibbons 2012, 3; Myllymäki ym. 2010, 226 - 227.)

Mallissa, jossa muutoksia viedään uuteen järjestelmään yksitellen arvioiden, on järkevää ensin tutustua riittävän hyvin uuteen versioon ja vasta tämän jälkeen lähteä arviomaan yksittäisiä muutoksia. Muutoin riskinä on vanhan toiminnallisuuden päivittäminen uuteen versioon, vaikka uusi vastaava toiminnallisuus olisi valmiina. Uudessa versiossa kannattaa myös välttää mahdollisimman paljon ohjelmistomuutosten tekemistä ja pyrkiä käyttämään uusia toimintoja. (Gibbons 2012, 3.)

3.3 Työvälineet

Työvälineitä järjestelmämuutosten hallinnointiin ja päivittämiseen on olemassa, mutta ne ovat yleensä ohjelmistotoimittajakohtaisia tai johonkin muuhun tarkoitukseen tarkoitettuja, kuten Microsoft Excel. Excelillä on kuitenkin etuna sen yleisyys ja tunnettavuus, mistä seuraa se, ettei käyttäjiä tarvitse samalla tavalla kouluttaa sen käyttöön. Excelin haittana on taas sen käyttö asiakkaan ja järjestelmätoimittajan välillä. Käytännössä Excel-tiedostoja joudutaan lähettelemään asiakkaan ja järjestelmätoimittajan välillä tai sitten käyttämään jotain yhteistä työtilaa, esim. Microsoft Sharepointia.

Atlassianin JIRA on projektin seurantaohjelmisto, joka mahdollistaa myös yksittäisten järjestelmämuutosten seuraamisen. Siitä löytyy toiminnallisuuksia, joilla pystytään luokittelemaan eri järjestelmämuutoksia useisiin luokkiin ja osoittamaan näitä kohteita tarvittaville resursseille. JIRA ohjelmistoa käyttämällä on mahdollista ottaa asiakas mukaan järjestelmämuutosten seurantaan, kehitykseen ja testaukseen. Ohjelmistossa on mahdollista antaa resursseille eri rooleja, joilla on erilaiset käyttöoikeudet. Roolien avulla asiakkaan ja järjestelmätoimittajan henkilöt näkevät ohjelmistossa eri asioita. (Atlassian.)

Microsoftin Team Foundation Server eli TFS on järjestelmä, joka auttaa hallitsemaan ohjelmistojen kehitystä. Se mahdollistaa useiden henkilöiden suorittaman samanaikaisen ohjelmiston kehittämisen ja siinä on myös versionhallinta. Sen kanssa on mahdollista käyttää mm. Microsoft Projectia ja Microsoft Exceliä. TFS:n työyksiköiden avulla järjestelmämuutoksia voidaan luokitella ja osoittaa tarvittaville resursseille. Sitä voidaan käyttää myös selainkäyttöliittymän kautta, mikä mahdollistaa asiakkaan pääsyn järjestelmään. (Microsoft b.)

Mikä tahansa järjestelmä tai ohjelmisto valitaankaan, tulee sen mahdollistaa asiakkaan ja järjestelmätoimittajan yhteistyö järjestelmämuutosten hallinnan, määrittelyn ja testausten osalta. Asiakkaalle on tärkeää päästä seuraamaan mitä on tehty ja missä vaiheessa järjestelmämuutokset ovat. Yhtä tärkeää on järjestelmätoimittajan pystyä seuraamaan asiakkaan suorittamia testauksia ja niiden tuloksia.

4 Microsoft Dynamics AX päivityskokemukset

4.1 Tutkimusmenetelmä

Tutkimusmenetelmänä käytettiin laadullista tutkimusta eli tutkimus tehtiin sopimalla yksilölliset haastattelut kaikkien haastateltavien kesken. Tämä tutkimusmenetelmä antoi haastateltaville mahdollisuuden puhua vapaammin ja syvällisemmin aiheesta. Lisäksi valittu tutkimusmenetelmä vähensi tutkimuksen tekijän omien ennakko-odotusten mahdollista vaikutusta, kun tutkimusta tehtiin vapaamuotoisilla haastatteluilla, eikä etukäteen laadituilla vastausvaihtoehdoilla. Haastateltaville toimitettiin haastattelukysymykset etukäteen tutustuttavaksi. Itse haastattelussa haastattelukysymykset toimivat kuitenkin vain runkona haastateltavien kertoessa omista kokemuksistaan ja käsityksistään aiheista vapaasti. Haastatteluissa esitettiin haastateltavien vastauksiin tarkentavia lisäkysymyksiä, joilla pyrittiin saamaan perusteellisempi käsitys kulloinkin kyseessä olevasta aiheesta tai asiasta. Haastateltavilta kysytyt kysymykset ovat liitteessä 2. Haastattelut tallennettiin äänitiedostoiksi ja niistä tehtiin muistiinpanoja. Muistinpanot ja tallenteet purettiin tuloksiksi.

Taulukko 2. Haastatellut

Nimi	Asema	Yritys
Andersson, J.	Toimitusjohtaja	iVersum Oy
Era, H.	Johtava konsultti	eCraft Oy
Henriksson-Vainio, T.	Vanhempi konsultti	Evry Business Solutions Oy
Kestilä, J.	Toimitusjohtaja	1ClickFactory Finland Oy
Laine, T.	Vanhempi konsultti	Tmi Lainetar
Niemelä, J.	Partner Account Manager	Microsoft Finland Oy
Sundström, J.	Vanhempi konsultti	Daxmate Oy

Kaikilla seitsemällä haastatellulla (taulukko 2) on hyvin pitkä kokemus Microsoftin Dynamics AX:sta ja usealla myös muista toiminnanohjausjärjestelmistä. He ovat kollektiivisesti olleet mukana useissa kymmenissä käyttöönotto- ja päivitysprojekteissa. Haastatellut edustavat järjestelmätoimittajan puolelta mm. arkkitehdin, konsultin, projektipäällikön ja hankejohdon näkemyksiä ja kokemuksia, kuin myös järjestelmävalmistajan edustajan näkemyksiä ja kokemuksia. Haastateltujen mielipiteet ja näkemykset ovat kuitenkin heidän omiaan, eivätkä välttämättä edusta heidän työnantajiensa virallista näkemystä.

Koska haastatellut edustavat eri organisaatioita ja toimivat erilaisissa tehtävissä Microsoft Dynamics AX:n parissa, saatiin tutkimukseen varsin kattava otos haasteista ja toisaalta myös onnistumisista, joihin suomalaiset yritykset toiminnanohjausjärjestelmiä päivittäessään joutuvat. Vaikka haastatteluissa keskityttiin Microsoft Dynamics AX:aan, niin tulokset ovat suoraan käytettävissä myös muiden toiminnanohjausjärjestelmien kanssa.

4.2 Tulokset

Haastateltavat kuvasivat omien kokemustensa pohjalta useita syitä, miksi toiminnanohjausjärjestelmien päivitysprojektit epäonnistuvat tai onnistuvat. Mielenkiintoista oli se, että tässä kohtaa sekä teoria että käytäntö kohtasivat hyvin, joskin ulkopuolisen projektipäällikön käyttämisessä oli eroa haastateltavien mielipiteiden ja aikaisempien julkaisujen välillä. Kaksi haastateltavista nosti esille sen, ettei asiakkaan palkkaamalla ulkopuolisella projektipäälliköllä välttämättä ole hyviä mahdollisuuksia onnistua tehtävässään. Tämä siksi, ettei hänelle anneta välttämättä riittäviä valtuuksia tehtävän suorittamiseen. Toisaalta Herrig ja Scanlon (2012), kuten myös Berg (2013) huomauttavat, että kustannuksista huolimatta on järkevää palkata ammattimainen projektipäällikkö, joka on keskittynyt tekemään projektipäällikön tehtäviä ja pystyy ehkä näin luovimaan ajoissa projektin pahimpien karikoiden ohi. Kaikki haastateltavat olivat kuitenkin sitä mieltä, että projektipäällikön rooli on hyvin tärkeä päivitysprojektin onnistumisessa.

Päivitysprojektien epäonnistumisten seurauksista usea haastateltava nosti esiin liikevaihdon menetyksen, kun tilauksia ei saada toimitettua eikä laskuja lähetettyä. Myös laskutuksen sujuvuuden varmistaminen tuli erikseen esille, jolloin lyhyellä tähtäimellä saadaan varmistettua yrityksen kassavirta.

Järjestelmämuutosten päivittämisessä ja siirtämisessä pidettiin tärkeänä kokonaisuuden hallintaa sekä toiminnallisesti että kustannuksellisesti. Yhtenä mahdollisuutena tähän mainittiin, että ulkopuolinen niihin keskittynyt taho, kuten 1ClickFactory, tekee alustavan arvioinnin kooditasolla ja arvioi siinä muutoksien merkittävyyttä ja kokoluokkaa.

Suurin osa haastateltavista painotti Sure Step – metodologian käyttöä toimintatapana. Myös Sure Step -metodologiasta irrotettujen toimintojen, kuten Proof of conceptin, käyttöä yksittäisten ongelmien ratkaisuun pidettiin toimivana. Testaukseen prosessi kerrallaan, eikä vain yksittäisten järjestelmämuutosten testaamiseen, kiinnitettiin huomiota. Testiympäristöistä taas nostettiin esille, että niissä tulee olla runsaasti ihmisiä tekemässä testausta.

Haastatellut kuvasivat, miten käytännössä kaikki päivitysprojektit saadaan jollain tasolla tehtyä, mutta niiden kanssa on olla voinut siitä huolimatta runsaasti ongelmia. Ongelmat voidaan yleisellä tasolla tiivistää yhteen sanaan: aika. Aika tosin tarkoittaa päivitysprojekteissa myös rahaa, sillä projektin venyminen aiheuttaa lisääntyneitä kustannuksia.

Haastatelluilla ei ollut mitään suurta suosikkityövälinettä järjestelmämuutosten hallitsemiseen, vaan he painottivat enemmän yhteistyötä asiakkaan kanssa. Oli työväline tai -tapa mikä tahansa, on sen mahdollistettava asiakkaan osallistuminen hallintaan, kehitykseen ja testaukseen.

Tulokset ovat kirjoitetut niin, ettei niistä pysty tunnistamaan haastatteluissa mahdollisesti mainittuja yksittäisiä projekteja tai asiakkaita, joiden kanssa haastatellut ovat olleet tekemissä.

4.2.1 Miksi päivitysprojektit epäonnistuvat?

Andersson toi haastattelussa esille, miten projekti voi epäonnistua jo kilpailutus- tai tarjousvaiheessa, kun valitaan halvin, ei parasta. Tällöin voidaan joutua muuttamaan tehtyjä määrittelyjä valinnan jälkeen. Samaa asiaa sivuaa myös Myllymäki ym. (2010, 220 - 224) joka huomauttaa, että tarjouspyynnöistä jätetään olennaisia kustannuksia pois tahallisesti, mutta myös osaamattomuuden takia. Voidaan mm. arvioida tarvittavan koulutuksen määrä huomattavasti todellista pienemmäksi tai ollaan ylioptimistisia. Toimittaja voi tarjota myös perustoiminnallisuutta ja luottaa, että asiakas tilaa loput tarvittavat ominaisuudet myöhemmässä vaiheessa projektin jo alettua. Asiakas taas voi pyytää projektin alkuvaiheessa budjettitarjousta ja peilata toteutuneen projektin hintaa alkuperäiseen budjettitarjoukseen huomioimatta tehtyjä lisätöitä. (Andersson 2012; Sundström 2012.)

Kannattavuuslaskelma tai laajemmin käsitettynä Business case, pitää tehdä etukäteen sekä päivitysprojektista kokonaisuutena että jokaisen toiminnallisuuden osalta vielä erikseen. Olemassa olevia toimintoja ja asiakaskohtaisia muutoksia pitää kyetä tarkastelemaan kriittisesti ja tässä kannattavuuslaskelmasta on suurta hyötyä. Tässä voidaan hyödyntää Beaubouefin (2012) luomaa kaaviota (kaavio 1), joka auttaa huomioimaan eri asioita, jotka muuten jäisivät huomiotta. Kannattavuuslaskelmat auttavat liiketoimintajohtoa ymmärtämään, mitä hyötyä toiminnanohjausjärjestelmän päivittämisestä on ja sitä kautta myös luovuttamaan päivitysprojektille sen tarvitsemat resurssit. (Andersson 2012; Henriksson-Vainio 2012; Kestilä 2012; Sundström 2012.)

Voidaan epäonnistua valitsemalla väärä strategia päivitykseen. Weiss (2011) huomauttaakin, että mikä tahansa strategia valitaan, on sen sovittava yritykselle ja yrityksen liiketoimintaan. Testataan testissä, kunnes kaikki on lähes valmista verrattuna siihen, että mennään tuotantoon lähes vakiolla järjestelmällä. Lech (2004) ja Robertson (2011) kannustavatkin valitsemaan vain lähes vakiojärjestelmän ja muistuttavat, että 80 prosenttia järjestelmämuutoksista saadaan tehtyä 20 prosentilla työstä. Näin saadaan päivityksen kustannusten takaisinmaksuaika selvästi lyhyemmäksi ja järjestelmämuutosten tekoa voidaan tarvittaessa jatkaa käyttöönoton jälkeen. (Andersson 2012; Niemelä 2012.)

Ennen varsinaista päivitysprojektia tulisi tehdä määrittelyprojekti sekä vertailu halutun ja vakiotoiminnallisuuden välillä, ns. fit-gap, jotta voidaan ylipäätään onnistua päivityksessä. Päivitysprojektissa tulisi muutenkin olla samoja vaiheita kuin normaalissa käyttöönottoprojektissa. Päivitysprojekteja helposti vähätellään ja ajaudutaan sen takia ongelmiin. Asiakkaat eivät aina halua osallistua päivitysprojekteihin, koska kyseessä on ”vain helppo” päivitysprojekti. Päivitystä tulisikin arvioida pikemmin uutena käyttöönottona, koska jokainen toiminnallisuus pitää kuitenkin arvioida uudestaan. Näin on etenkin silloin, kun ”hypätään” version yli. Projekteja voidaan kuitenkin jakaa kahteen: pieniin ja isoihin. Pienemmissä voidaan mennä kevyemmällä teknisellä päivityksellä, kun taas isommissa pitää olla täysi projektiorganisaatio. (Andersson 2012; Henriksson-Vainio 2012; Laine 2012; Sundström 2012.)

Testauksen osuutta ei saa unohtaa. Usein projekteissa on ongelmana, että samat ihmiset määrittelevät ja testaavat muutokset ja prosessit. Sen sijaan pitäisi saada tavalliset käyttäjät mukaan testaamaan. Näin myös silloin, vaikka itse käyttöliittymä tai päätoiminnot pysyisivät ennallaan (SearchManufacturingERP 2010). Lisäksi saatetaan pitää hyvin tiukasti kiinni suunnitellusta käyttöönottopäivästä ja sen takia tinkiä testauksesta. Testauksessa ei kuitenkaan saa kiirehtiä, vaan pitää testata huolellisesti. Joskus saattaa ongelmana olla, ettei asiakas ymmärrä lopullisen testausvastuun kuuluvan heille. Toimittaja toki testaa muutokset ennen niiden luovuttamista asiakkaalle testiin, mutta vastuu säilyy asiakkaalla. Järjestelmien käyttöönottoa tulisi simuloida siitä aiheutuvista kustannuksista huolimatta eli niille tulisi tehdä ns. kenraaliharjoitus. Herrig ja Scanlon (2012) ovat samaa mieltä ja lisäävät vielä, että jos käyttöönotto on suunniteltu viikonlopulle, tulisi myös kenraaliharjoitus pitää viikonloppuna ja näin paremmin löytää piileviä ongelmia, kuten tietokantavarmistukset ja ovien aukiolo. Liiketoimintajohdolle tällaisen kenraaliharjoituksen pitäminen on helpompi perustella, jos siitä on tehty kannattavuuslaskelma, missä on huomioitu pois jääneen työpäivän hinta liikevaihtoineen ja muine vaikutuksineen. (Andersson 2012; Henriksson-Vainio 2012; Laine 2012; Niemelä 2012.)

Projektin organisointi ja johtaminen on elintärkeää. Asiakkaan projektihenkilöille pitää varata aikaa ja johdon pitää varata projektille riittävästi resursseja, jotta projektipäällikkö voi onnistua. Projektipäällikön pitäisi pystyä keskittymään puhtaasti omaan työhönsä, eikä tehdä oman työn ohella projektipäällikön tehtäviä. Projektipäälliköiden rooli onkin voimakkaan korostunut, vaikka järjestelmäkonsulttien roolia ei voikaan väheksyä. Projektipäällikkö on päivitysprojektissa yhtä tärkeä kuin normaalissakin käyttöönotto-projektissa. Järjestelmätoimittajan projektipäällikön tulisi olla ajan tasalla oleva ammattilainen, jolla on mahdollisuus tarvittaessa tukea asiakkaan projektipäällikköä. Asiakkaan projektipäälliköllä taas pitää olla riittävästi valtaa ja auktoriteettia, jotta hän kykenee toimimaan projektin parhaaksi. Usein tehty virhe onkin palkata ulkopuolinen projektipäällikkö, jolla ei ole riittävästi valtaa pakottaa henkilöitä tekemään omaa osaansa ja tällöin vaarana on epäonnistuminen. Herrig ja Scanlon (2012), kuten myös Berg (2013) ovat jokseenkin eri mieltä ja huomauttavat, että on järkevää palkata ammattimainen projektipäällikkö, joka on keskittynyt tekemään projektipäällikön tehtäviä ja pystyy ehkä näin luovimaan ajoissa projektin pahimpien karikoiden ohi. (Andersson 2012; Era 2012; Henriksson-Vainio 2012; Kestilä 2012; Laine 2012; Niemelä 2012; Sundström 2012.)

Microsoft Sure Step -metodologiaa käyttämällä pystyy nostamaan onnistumisen mahdollisuuksia. Asiakkaat eivät kuitenkaan aina halua käyttää Sure Step -metodologiaa, jolloin hylätään hyvä työkalu ja paljon valmista Microsoftin tekemää pohjatyötä. (Laine 2012; Sundström 2012.)

Otetaan uusi versio liian aikaisessa vaiheessa käyttöön, jolloin järjestelmätoimittajien osaaminen ei ole vielä niin syvällä tasolla. Ensimmäisessä versiossa voi olla vielä virheitä, joita ei ole havaittu. SearchManufacturingERP (2010) ehdottaakin, että kannattaa odottaa, että järjestelmävalmistaja julkaisee ensimmäisen korjauspaketin ja on selvitelty muiden asiakkaiden kokemuksia ko. versiosta ennen siihen päivittämistä. (Kestilä 2012.)

Työmäärät on arvioitu väärin, etenkin asiakkaan työmäärät. Asiakkaan kannattaakin jo tarjouspyyntövaiheessa viestiä, että työmääräarvioiden halutaan olevan realistisia (Myl-

lymäki ym. 2010, 254.) Asiakkaat suunnittelevat tekevänsä asioita oman toimen ohella ymmärtämättä todellista työn määrää. (Niemelä 2012.)

Käyttäjät, jotka kokevat uuden version uhaksi, voivat ryhtyä aktiivisesti vastustamaan päivitysprojektia. Nämä henkilöt pitäisi saada jotenkin mukaan edesauttamaan projektia. Tässä voi olla ratkaisuna projektinaikainen viestintä, joka kuitenkin valitettavan usein epäonnistuu. Herrig ja Scanlon (2012) sekä Weiss (2011) ovat samoilla linjoilla ja heidän mukaansa käyttäjät eivät pidä muutoksista, koska ne aiheuttavat heille ylimääräistä työtä ja he tarvitsevat tietoa ja aikaa voidakseen sopeutua tuleviin muutoksiin. (Kestilä 2012; Niemelä 2012.)

Käytetty kieli voi aiheuttaa ongelmia. Jos esimerkiksi määrittelyt on tehty englanninkielellä eikä organisaatio kuitenkaan osaa sitä, niin väärinkäsitysten mahdollisuus kasvaa huomattavasti. Määrittelyt ja muu dokumentaatio tulisikin olla organisaation normaalilla työskentelykielellä riippumatta konsernikielestä. Konsernikielen lisäksi muita syitä eri kielellä tehtävään dokumentaatioon voi olla mm. järjestelmätoimittajan käyttämät alihankkijat tai järjestelmävalmistajan tuottama dokumentaatio. (Sundström 2012.)

Tietokantarakenteen muuttuessa tietojen siirtäminen vanhasta versiosta uuteen versioon tulee ongelmaksi ja siihen joudutaan käyttämään enemmän resursseja, jotta ne saadaan siirrettyä. Kun tietokanta ja sitä käyttävä logiikka muuttuvat, joudutaan helposti rakentamaan myös liittymät täysin uudestaan (SearchManufacturingERP 2010). Liittymien huomioonottaminen onkin erittäin tärkeää. Niinpä liittymiin liittyvää työmäärää ja testausta ei saisi väheksyä, vaan mieluummin jopa hieman liioitella. Onnistumisen mahdollisuus kuitenkin kasvaa, kun päivitetään saman järjestelmävalmistajan uudempaan versioon, koska järjestelmätoimittajan konsultit osaavat molemmat versiot. Järjestelmätoimittajan konsulttien ammattitaito nousee kuitenkin korostetusti esiin, etenkin uuden version ominaisuuksien tunteminen. Mitä suurempi ero järjestelmien välillä on, sitä vaikeampaa päivittäminen on, ja sitä järkevämpää on tehdä uusi käyttöönotto, eikä lähteä päivittämään vanhaa versiota uuteen. (Andersson 2012; Laine 2012; Sundström 2012.)

Asiakkaan asenne versiopäivitykseen voi olla: ”Päivitys on pakollinen paha”. On pakko päivittää, kun alusta vanhenee tai tuki loppuu tai jostain muusta syystä. Tästä seuraa, että päivityksessä pyritään säästämään kaikissa mahdollisissa kuluissa. Tällöin hyvän lopputuloksen saavuttamiselle ei anneta kovin isoja mahdollisuuksia. Monesti päivitysprojekti tulisikin nähdä uuden ekosysteemin luomisensa eikä versiopäivityksenä. (Andersson 2012; Kestilä 2012.)

Asiakkaan ymmärrys siitä mitä ollaan tekemässä, on kriittisessä asemassa. Asiakkaan pitäisi osata hankkia osaamista talon ulkopuolelta, jos sitä ei löydy omasta talosta, esimerkkinä valintakonsulttien käyttäminen apuna valinnassa. Toimiva ratkaisu voi olla myös strateginen kumppanuus jonkun järjestelmätoimittajan kanssa. Tietojärjestelmän hankintakäsikirjassa (Talentum 2005) todetaankin seuraavaa ”Kumppanuus mahdollistaa molemminpuolisen luottamuksen kehittymisen, tehokkaan oppimisympäristön ja hyvin toimivien yhteyksien kehittymisen toimittajan ja asiakkaan henkilöiden kesken.” Joissain erikoistapauksissa järjestelmätoimittajan toimiminen asiakkaan ”edunvalvojana” on myös mahdollista. (Era 2012.)

4.2.2 Milloin päivitysprojektit onnistuvat?

Päivitystä ei saisi katsoa vain kustannuksena, vaan mahdollisuutena kehittää yrityksen toimintaa. Päivitysprojektin aikana voidaan kehittää yrityksen toimintoja ja kilpailukykyä sekä ottaa käyttöön ohjelmiston uusia toiminnallisuuksia. Samaa mieltä ovat myös Shepherd (2007, 2) ja Stackpole (2008), jotka toteavat että versiopäivitykset ovat hyvä hetki tarkastella tarpeettomia järjestelmämuutoksia ja vanhentuneita ohjelmistoja. Monesti kuitenkin nojataan liikaa vanhaan tuttuun versioon, eikä uskalleta ottaa uusia toiminnallisuuksia käyttöön. Vaikka päivitysprojekti onnistuisikin teknisesti ja pysyisi aikataulussaan sekä budjetissaan, voi käydä niin, ettei päivityksestä ole mitään hyötyä liiketoiminnalle, koska uutta versiota käytetään täsmälleen samalla tavalla kuin vanhaa, eikä uusia toimintoja ole otettu käyttöön. Kun uskalletaan avoimesti arvioida uuden version tuomia hyötyjä, eikä jäädä vanhan version käyttötapoihin kiinni, voidaan päivitysprojektista saada isot hyödyt. (Andersson 2012.)

Päivitysprojektin tavoitteet pitää olla selkeät eli se mitä ollaan tekemässä, tulee määritellä tarkasti. Myös Berg (2013) kertoo onnistumisen helpottuvan, kun kaikilla on samanlainen käsitys siitä mitä ollaan tekemässä. Jos lähdetään samalla tyhjentämään ”toiveiden tynnyriä”, projektin budjetti tai aikataulu ei tule pitämään. Kehitysideat onkin syytä kerätä omiksi jatkokehitysprojekteikseen. (Henriksson-Vainio 2012.)

Kuka päättää mitä tehdään ja kenelle järjestelmää ollaan tekemässä: loppukäyttäjille, johdolle vai yritykselle? Päivitysprojekteissa, kuten myös käyttöönottoprojekteissa on mukana useita eri ryhmiä, joilla voi olla toisistaan poikkeavat tavoitteet projektille. Toiset voivat haluta pystyä tekemään asioita mahdollisimman yksinkertaisesti ja nopeasti välittämättä siitä miten tiedot tallentuvat. Toiset taas haluavat saada järjestelmästä mahdollisimman monipuolista ja tarkkaa raportointitietoa ja vaativat, että tiedot pitää sen takia syöttää monipuolisesti ja huolellisesti järjestelmään. Jotta toiminnanohjausjärjestelmästä tulee yrityksen työkalu, pitää projektin johdon hallita projektia ja ratkaista eri näkemysten aiheuttamia ristiriitoja. (Andersson 2012; Sundström 2012.)

Toimittajan kriittisten resurssien varmistamisesta tulisi huolehtia, vaikka tästä voi asiakkaan olla vaikea huolehtia. Kannattaa kuitenkin pyrkiä tarjouspyynnössä, sopimuksissa ja hinnoittelussa ottamaan huomioon, että toimittaja tarjoaa parhaita resurssejaan päivitysprojektille ja että ne myös pysyvät projektissa. Myös Myllymäki ym. (2010, 268) ehdottavat, että asiakkaan kannattaa sopimuksessa lukita parhaat konsultit projektiin, vaikka se maksaisikin hieman enemmän. Mikäli kriittisiä resursseja ei onnistuta varmistamaan, on vaarana, että ne vähitellen katoavat toimittajan muihin projekteihin. (Era 2012.)

Yritysten kannattaa pyrkiä rakentamaan suojakerroksia ytimen eli tuotannon ympärille, niin että muutoksista huolimatta kassavirta pysyy ennallaan. Tästä esimerkkinä ovat toiminnanohjausjärjestelmien ympärille rakennettavat raportointi- ja tiedonsyöttöjärjestelmät. (Andersson 2012.)

Jokaiselle liiketoimintaprosessille tulee löytää omistaja, joka sitoutuu päivitykseen ja on mukana viemässä prosessia päivityksen mukana maaliin saakka. Ihmiset pitää saada innostumaan päivityksestä ja näkemään se mahdollisuutena. Mukaan pitää saada yrityksen johto hallitusta myöden. Weissinkin (2011) mukaan keskeisenä asiana onnistumisessa on ottaa liiketoimintayksiköiden vetäjät ja varsinaiset loppukäyttäjät aikaisessa vaiheessa mukaan päivitysprosessiin. Näin saadaan arvokasta palautetta, joka voi estää pienten ongelmien muuttumisen isoiksi ongelmiksi. (Andersson 2012; Sundström 2012.)

4.2.3 Päivitysprojektien epäonnistumisten seuraamukset

Tutkimuksessa paneuduttiin myös seurauksiin, joita toiminnanohjausjärjestelmän päivityksen epäonnistumisella voi olla. Haastateltavat mainitsivat ainakin seuraavia seurauksia voivan syntyä epäonnistumisista:

Päivitysprojektiin on valittu väärä teknologia tai rakennettu liian jäykkä järjestelmä, joka aiheuttaa taloudellisia menetyksiä, kun prosesseja joudutaan jälkikäteen muuttamaan ja työaika kuluu jäykkien toimintojen kanssa selviämiseen. (Andersson 2012.)

Käytännössä kaikki päivitysprojektit saadaan jollain tasolla ”maaliin”, mutta aikaa ja rahaa on voinut kulua suunniteltua enemmän ja toimintoja on voitu joutua korjaamaan jälkikäteen. Riskit rajoittuvatkin yleensä uuden version rakentamiseen käytettyyn aikaan, kuten myös järjestelmätoimittajan aikaan sekä asiakkaan työntekijöiden työaikaan ja rahaan. Joissain tapauksissa toteutuneina riskeinä on voinut olla menetettyjä tietoja, kun varmistukset eivät ole olleet ajan tasalla tai menetettyä liikevaihtoa, kun järjestelmä ei ole ollut käytössä ja on jouduttu palaamaan takaisin vanhempaan versioon. (Era 2012; Henriksson-Vainio 2012; Kestilä 2012; Laine 2012; Niemelä 2012; Sundström 2012.)

Laskutus voi keskeytyä ja niihin liittyviä asioita tulisikin priorisoida. Mikäli kassavirta pysähtyy, voi pieneltä tuntuva tekninen ongelma muuttua yritykselle kohtalokkaaksi. (Era 2012; Henriksson-Vainio 2012.)

Yrityksen asiakkaat voivat suuttua ja lakata luottamasta yritykseen, jos tilaukset eivät kulje ja laskutus ei toimi, tällöin he voivat lopulta päätyä vaihtamaan yritystä toiseen, jonka järjestelmät toimivat. (Henriksson-Vainio 2012.)

Loppukäyttäjien turhautuminen voi johtaa siihen, että he laiminlyövät järjestelmää ja pyrkivät toimimaan järjestelmän ulkopuolella tai jättävät tarkoituksella asioita tekemättä järjestelmässä. (Henriksson-Vainio 2012).

4.2.4 Aiemmin tehtyjen järjestelmämuutosten siirtäminen uuteen versioon

Hyvä tapa toimia on pystyttää rinnakkainen standardijärjestelmä uudella versiolla ja asiakkaan ja järjestelmätoimittajan yhdessä arvioida vertikaalien hyödyllisyys siinä. Eli uudessa versiossa toteutetaan ns. Proof of concept (POC). Tämä Proof of concept kuuluu osana SureStep –metodologiaan. Järjestelmätoimittajan olisi hyvä asettaa tätä arviointityötä varten oma kehittäjä, joka voi keskittyä vain tähän yhteen asiaan. Tämän jälkeen voidaan tarvittaessa arvioida järjestelmämuutoksia toiminto kerrallaan, peilaten niitä uuteen toiminnallisuuteen. Haasteena vanhojen järjestelmämuutosten arvioinnissa on usein vanhojen taustojen ja syiden puuttuminen eli miksi jokin muutos on alkujaan tehty. Kun ei tiedetä alkuperäistä syytä järjestelmämuutokselle, on paljon vaikeampaa arvioida sen nykyistä tarvetta. Jos ollaan hyppäämässä usean version yli, saattaa olla järkevää olla siirtämättä ainuttakaan järjestelmämuutosta vanhasta järjestelmästä ja katsoa vain vanhan järjestelmämuutoksen ideaa ja miettiä voiko sen tehdä toisin uudessa versiossa. (Andersson2012; Henriksson-Vainio 2012; Kestilä 2012; Laine 2012.)

Mahdollista on toimia myös niin, ettei siirretä ainuttakaan vanhaa järjestelmämuutosta uuteen versioon, ennen kuin tarve sille esiintyy. Weiss (2011) kannattaakin tätä toimintatapaa ja huomauttaa, että asioita voidaan muuttaa myös jälkikäteen. Tämä ei kuitenkaan ole mahdollista kaikissa toiminnoissa tai prosesseissa. Myös toiminta, jossa päivitetään kaikki järjestelmämuutokset, mutta ei oteta niitä vielä käyttöön, voi olla järkevää. Tässä mallissa pystytään reagoimaan käyttäjien esittämiin tarpeisiin huomattavasti nopeammin, kuin jos vasta tarpeen ilmennyttyä ryhdytään muutosta päivittämään. Muutosten tutkiminen ja arviointi yksitellen voi maksaa enemmän kuin niiden päivittäminen kerralla. Yhtenä nopeana arviointikriteerinä voi käyttää vanhan version kenttiä ja verra-

ta niitä uuteen standardiversioon. Mikäli käytössä on kenttiä, joita uudesta versiosta ei löydy, lienee selvää, että jonkin tasoinen järjestelmämuutos on tarpeen. (Niemelä 2012.)

Kun uusia vertikaaleja ja vanhoja järjestelmämuutoksia arvioidaan ja testataan, on syytä muistaa arvioida koko prosessin toimivuutta. Käytännössä joudutaan aika ajoin pitämään prosessitestausta, jossa käyttäjät testaavat prosessin alusta loppuun. Prosessitestauksen aikana tehdyt huomiot kirjataan ja niiden perusteella tehdään muutoksia, joko itse prosessiin tai aikaisempiin järjestelmämuutoksiin. (Sundström 2012.)

Testiympäristössä tulee olla runsas määrä ihmisiä koekäyttämässä ja arvioimassa toimintoja. Tämä on toimivin tapa pyrkiä varmistumaan siitä, että tehdyt asiat toimivat ja toisaalta myös tehdään oikeita asioita. Lisäksi yrityksen prosesseja tulisi arvioida kriittisesti. Prosessien arvioinnissa voi olla hyödyllistä käyttää ulkopuolista asiantuntijaa apuna ja saada näin tuore näkökulma asiaan. (Andersson 2012.)

Toistaiseksi vielä vähemmän käytetty mahdollisuus on käyttää ulkoistuspalvelua, eli asiantuntijoita, jotka tekevät vain järjestelmäpäivityksiä työkseen järjestelmämuutosten siirtämiseen uuteen versioon. Esimerkkinä on 1ClickFactoryn käyttäminen päivityksissä teknisenä toimijana. (Kestilä 2012.)

Toimintatapoja ja standardeja järjestelmämuutosten siirtämiseen on lähinnä Microsoft Dynamics tuotteiden osalta Sure Step metodologia, missä otetaan kantaa päivitysten tekemiseen sekä Microsoftin MSDN:n tekniset ohjeet. MSDN:n teknisissä ohjeissa annetaan neuvoja ja suosituksia yksittäisiin asioihin, mutta myös laajempiin toiminnallisiin kokonaisuuksiin. Hyväksi toimintatavaksi on myös havaittu työpajat, joissa käydään läpi halutut järjestelmämuutokset käytännön tasolla. (Era 2012; Henriksson-Vainio 2012; Kestilä 2012; Laine 2012; Niemelä 2012; Sundström 2012.)

4.2.5 Järjestelmämuutosten hallitseminen ja työvälineet siihen

Järjestelmämuutosten kokonaisuus pitää ensin tuntea, jotta sitä voi hallita ja päivittää. Tämä on mahdollista suorittamalla tekninen esikartoitus järjestelmämuutoksista. Teknisessä esikartoituksessa arvioidaan muutosten suuruus ja sitä mihin järjestelmän osiin ne kohdistuvat. Yhtenä mahdollisuutena on, että 1ClickFactory tekee alustavan arvioinnin kooditasolla ja arvioi siinä muutoksien merkittävyyttä ja kokoluokkaa. Tämän arvioinnin jälkeen on järjestelmätoimittajan tehtävä antaa asiakkaalle arvio järjestelmämuutosten kokonaisuudesta. (Henriksson-Vainio 2012; Kestilä 2012; Laine 2012.)

Järjestelmämuutoksia kannattaa pyrkiä hallinnoimaan prosessi kerrallaan, ei yksitellen. Käyttäjien tulisikin arvioida prosessi kerrallaan omaan työhönsä verraten. Näin käyttäjien ammattitaitoa saataisiin arviointityöhön mukaan. (Sundström 2012.)

Yksittäisten järjestelmämuutosten hallintaan voidaan hyvin käyttää Microsoftin SharePointia, mutta sen kuten minkä tahansa muunkin työkalun käyttö riippuu siitä, kuinka hyvin ja täydellisesti järjestelmää käytetään. Jos SharePointia käytetään niin, että myös asiakkaalla on pääsy sinne, tuo se lisää tehoa järjestelmämuutosten hallintaan. Tällöin asiakas ja järjestelmätoimittaja voivat järjestelmämuutosten osalta käyttää SharePointia niiden hyväksyntään ja testaus- ja muutoskommentointeihin. (Era 2012; Sundström 2012.)

Pienissä päivitysprojekteissa on mahdollista tehdä järjestelmämuutosten hallintaa jopa Excelissä. Excelin vahvuutena on sen yleisyys ja se, että käyttäjät osaavat käyttää sitä ilman erillistä koulutusta. (Sundström 2012.)

Ei ole väliä mikä työväline on käytössä, kunhan siinä yksilöllinen tunniste jokaiselle järjestelmämuutokselle. Tuon yksilöllisen tunnisteen tulee löytyä myös itse muutoksen koodista. (Laine 2012; Niemelä 2012.)

AXP:tä käyttämällä voidaan jokaiselle järjestelmämuutokselle antaa oma yksilöllinen tunniste, jota voidaan käyttää viittauksena ohjelmistokoodissa ja keskusteluissa. AXP:ssä järjestelmämuutoksille annetaan kuvaus ja niiden valmistumisen ja testauksen edistymistä seurataan omina resursseille osoitettuina riveinään. AXP mahdollistaa myös asiakkaan osallistumisen hallinnointiin ja testausprosessiin. AXP on Microsoft Dynamics AX:n sisään kehitetty ohjelmisto, joka auttaa hallitsemaan järjestelmämuutoksia. (Henriksson-Vainio 2012; Sundström 2012.)

Microsoftin Sure Step on metodologia, joka ottaa huomioon myös järjestelmämuutosten tarpeen. Sure Stepiä käyttämällä pystytään paremmin arvioimaan järjestelmämuutosten tarvetta ja vaikutuksia. (Andersson 2012; Kestilä 2012.)

5 Toimenpidesuositukset

Tässä luvussa annetaan toimenpidesuosituksia päivitysprojektien onnistumisen edistämiseksi. Toimenpidesuositukset perustuvat tutkimuksessa löydettyihin esimerkkeihin onnistumisista ja toisaalta myös epäonnistumista. Toimenpidesuositukset toimivat sekä päivitysprojekteissa että käyttöönottoprojekteissa. Ne eivät myöskään ole toiminnanohjausjärjestelmäsidonnaisia, vaan niitä voidaan käyttää toiminnanohjausjärjestelmästä riippumatta.

Toiminnanohjausjärjestelmän päivitysprojekti kannattaa käynnistää, kun ilmiselvästi on sen aika. Näin riippumatta siitä tukevatko laskelmat sitä tai eivät. Järjestelmästä tulisikin pitää tietoisesti huolta ja sen päivittämisessä kannattaa käyttää maalaisjärkeä.

Päivitysprojektia tulisi seurata yhden mittarin sijasta useammalla mittarilla, joille voidaan antaa eri painoarvoja. Näin aikakriittisen projektin aikataulussa pysymiselle voidaan antaa suurempi painoarvo kuin esimerkiksi budjetissa tai alkuperäisissä projektin rajauksissa pysymisille.

Päivitysprojektille on hyvä laittaa useita seurantapisteitä, joissa arvioidaan koko projektia. Näiden seurantapisteiden avulla voidaan tarvittaessa ohjata projektia oikeaan suuntaan tai jopa keskeyttää se, ennen kuin suurempia menetyksiä syntyy.

On syytä arvioida erittäin kriittisesti kannattaako lähteä päivittämään järjestelmää sellaisenaan vai aloittaa uudestaan uudella versiolla. Jos päätetään aloittaa alusta, tulee samalla mahdolliseksi arvioida uudestaan aikaisemmin tehdyn ohjelmistovalinnan toimivuus ja nykyisen järjestelmätoimittajan sopivuus. Mikäli päätetään vain vaihtaa uuteen versioon, niin pystytään silti hyödyntämään aikaisemmin hankittua kokemusta järjestelmästä. Toisaalta tällöin myös toimittajalla on paljon suuremmat mahdollisuudet onnistua projektissa, koska toimittaja tuntee sekä uuden että vanhan järjestelmän, josta ollaan päivittämässä.

Jokaisesta järjestelmämuutoksesta tulisi löytyä liiketaloudellinen peruste, miksi se tehdään. Siinä hyvänä apuna voi olla Beaubouefin (2012) kaavio (kaavio 1), joka auttaa muodostamaan järjestelmämuutokselle liiketoiminnallisia perusteluita ja hylkäämään ne järjestelmämuutokset, joille perusteluita ei löydy.

Selkeän kustannustehokas malli järjestelmämuutosten arviointiin on jakaa järjestelmämuutokset ensin niiden päivittämisen vaatiman työmäärän mukaan kolmiportaisesti pieniin, keskisuuriin ja suuriin. Sitten samat järjestelmämuutokset kannattaa jakaa niiden liiketoimintakriittisyyden perusteella samalla tavalla kolmeen ryhmään.
Näin muutoksista on helpompi karsia pois ne, jotka eivät ole liiketoimintakriittisiä ja jotka toisaalta aiheuttaisivat runsaasti kustannuksia päivityksessä.

On syytä käyttää järjestelmämuutosten seuraamiseen jotain järjestelmää. Mikä tahansa järjestelmä tai ohjelmisto valitaankaan, tulee sen mahdollistaa asiakkaan ja järjestelmätoimittajan yhteistyö järjestelmämuutosten hallinnan, määrittelyn ja testausten osalta. Asiakkaalle on tärkeää päästä seuraamaan mitä on tehty ja missä vaiheessa järjestelmämuutokset ovat. Yhtä tärkeää on järjestelmätoimittajan pystyä seuraamaan asiakkaan suorittamia testauksia ja niiden tuloksia.

Kannattaa käyttää jotain projektimetodologiaa, kuten esimerkiksi Sure Step metodologiaa, joka määrittelee, mitä kukin tekee ja missä järjestyksessä, sekä kuka vastaa mistäkin. Projektimetodologiat tarjoavat yleensä useamman vaihtoehtoehdon projektin tyypiksi, joista valitaan parhaiten sopiva.

Muutosten testaamisessa kannattaa muistaa testata koko prosessia aika ajoin. Prosessitestauksen aikana tehdyt huomiot kirjataan ja niiden perusteella tehdään muutoksia joko itse prosessiin tai tehtyihin järjestelmämuutoksiin. Kannattaa muistaa myös käyttää testaamisessa riittävää määrää ihmisiä, jotta saadaan mukaan riittävästi erilaisia tapoja tehdä asioita ja myös mahdollisia virhetilanteita.

Tulee muistaa sisäinen viestintä sekä päivitysprojektin aikana, mutta myös ennen sitä. Tämä on erityisen tärkeää siksi, että näin saadaan lievitettyä vanhan järjestelmän käyttäjien tuntemaa luontaista muutosvastarintaa ja jopa käännytettyä heidät uuden järjestelmän puolelle. Päivitysprojektia ei ole mahdollista pilata tiedottamalla liikaa. Parasta olisikin, jos viestintää varten olisi oma henkilö, eikä projektipäällikkö hoitaisi viestintää oman toimensa ohella.

Kannattaa pyrkiä sitouttamaan päivitysprojektin avainresurssit sopimalla kirjallisesti heidän käytöstään, mutta myös hyväksymällä se tosiasia, että kokeneemmat osaajat maksavat enemmän. Kokeneempien senioriosaajien käyttö voi kuitenkin nopeasti maksaa ylimääräiset kustannukset takaisin säästyneenä aikana ja laadukkaampana määrittelynä ja ohjelmointina.

6 Yhteenveto ja loppusanat

Olen työskennellyt Microsoft Dynamics AX:n parissa vuodesta 2002 lähtien, ensin pääkäyttäjänä Väinö Korpinen Oy:ssä, sitten iVersum Oy:ssä johtavana konsulttina kahdeksan vuotta ja sen jälkeen reilun vuoden ajan takaisin asiakkaalla Are Oy:ssä kehityspäällikkönä. Nyt olen taas järjestelmätoimittajan puolella Digia Oyj:ssä ratkaisuarkkitehtinä. Tänä aikana olen päässyt tutustumaan ohjelmistoon perusteellisesti ja monesta eri näkökulmasta. Tutuiksi ovat tulleet niin käyttöönotot kuin päivitysprojektitkin. Halusinkin tehdä tutkimuksen nimenomaan tästä aiheesta, koska se liittyy niin voimakkaasti omaan työhöni ja tutkimus tarjosi myös mahdollisuuden kehittyä ammatillisesti. Tutkimusta ei tehty minkään yrityksen toimeksiannosta, joten olen voinut tehdä sitä täysin riippumattomasti ja itsenäisesti.

Tutkimukseen saatiin varsin kattava otos haasteista ja toisaalta myös onnistumisista, joihin suomalaiset yritykset toiminnanohjausjärjestelmiä päivittäessään joutuvat. Vaikka haastatteluissa keskityttiin Microsoft Dynamics AX:aan, niin tulokset ja toimenpidesuositukset ovat suoraan käytettävissä myös muiden toiminnanohjausjärjestelmien parissa. Tuloksia ja toimenpidesuosituksia voi käyttää hyväksi päivitysprojektien kanssa ja ne soveltuvat varsin hyvin myös varsinaisen käyttöönottoprojektin ohjeiksi.

Haastateltujen arvovaltaisuutta korostavat heidän kaikkien hyvin pitkät uransa toiminnanohjausjärjestelmien parissa. He kykenivät kertomaan useiden päivitysprojektien tuomalla kokemuksella asioita, joita myös tietoperustassa käsiteltiin. Tietoperustan pyrin saamaan mahdollisimman kattavaksi ja luotettavaksi käyttämällä monipuolisia, kansainvälisiä ja tuoreita lähteitä, mutta myös jo hyväksi tunnustettuja kotimaisia lähteitä. Mahdollisuuksien mukaan lähteitä on käytetty niin, että useampi lähde vahvistaa väitteen. Näin toimien on pystytty vähentämään yksittäiseen lähteeseen liittyvää epäluotettavuutta. Tutkimusta kokonaisuudessaan voitaneen pitää luotettavana, sillä myös tietoperusta tukee haastateltuja. Ainoastaan ulkopuolisen projektipäällikön käytössä oli haasteltujen ja tietoperustan välistä erimielisyyttä.

Kaksi haastateltavista nosti esille sen, ettei asiakkaan palkkaamalla ulkopuolisella projektipäälliköllä välttämättä ole hyviä mahdollisuuksia onnistua tehtävässään. Toisaalta Herrig ja Scanlon (2012), kuten myös Berg (2013) huomauttavat, että kustannuksista huolimatta on järkevää palkata ammattimainen projektipäällikkö, joka on keskittynyt tekemään projektipäällikön tehtäviä ja pystyy ehkä näin luovimaan ajoissa projektin pahimpien karikoiden ohi. Voi olla, että tämä ero johtuu sattumasta ja haastateltaville oli vain sattunut kohdalle mieleenpainuvia tapauksia. On myös mahdollista, ettei suomalainen yrityskulttuuri, etenkään pienemmissä yrityksissä, suosi ulkopuolisten projektipäälliköiden käyttämistä.

Yleisesti ottaen sekä teoria että käytäntö kohtasivat erinomaisesti. Haastatelluilla olikin jo käytössään päivitysprojektin onnistumista tukevia tapoja ja järjestelmiä. Lisäksi he nostivat haastatteluissa esille hyvin paljon sellaisia asioita, jotka huomioimalla päivitysprojektin onnistumisen todennäköisyys kasvaa huomattavasti. Näistä asioista on helppo poimia projektipäällikön osuus päivitysprojektin onnistumisessa, joka on vähintäänkin huomattava. Samalla tavalla projektipäällikön roolia korostettiin myös Bergin (2013) ja Herrigin ja Scanlonin (2012) toimesta. Projektipäälliköt lienevätkin hyvässä ja pahassa päivitysprojektin näkyvimpiä henkilöitä. Erityisen tärkeää on asiakkaan ja toimittajan projektipäälliköiden keskinäinen yhteistyö.

Minulla oli ennakko-oletuksena tätä tutkimusta aloittaessani, että päivitysprojekteja on pakko käsitellä samalla tavalla kuin varsinaisia käyttöönottoja. Tämä osoittautuikin paikkansapitäväksi. Sen sijaan erilaisten päivitysstrategioiden käyttö oli odottamaani runsaampaa sekä kirjallisuudessa että myös käytännössä. Tutkimusta tehdessäni oli mielenkiintoista huomata, miten tutkimustyö vei mennessään ja uppouduin kaikenlaiseen aiheeseen liittyvään materiaaliin. Myös oma ansiotyöni sai uutta perspektiiviä, kun selvitin erilaisia teorioita, käsityksiä, suosituksia ja esimerkkitapauksia tutkimusta varten. Pystynkin soveltamaan tätä tutkimusta omassa työssäni ja kehittämään omaa tekemistäni sen avulla. Tämä tutkimus ei vain onnistunut saavuttamaan sille asetettuja tavoitteita, vaan myös antamaan konkreettisia toimenpidesuosituksia toiminnanohjausjärjestelmien päivittämiseen.

Haluankin kiittää kaikkia haastateltavia, jotka uhrasivat useita tunteja aikaansa tätä tutkimusta varten. Ilman niitä haastatteluja ja niistä saatua erinomaista ja kattavaa materiaalia ei tämä tutkimus olisi valmistunut. Lämmin kiitos teille kaikille!

Lopuksi haluan jättää lukijalle ja kenties jollekin opinnäytetyön tekijälle, ajatuksen siitä, miten aihetta voisi vielä työstää jatkossa. Päivitysprojektien riskienhallintaa ei nimittäin tässä tutkimuksessa käsitelty ja mielestäni se ansaitsisi asiaan paneutumista. Riskienhallinnasta on olemassa kirjallisuutta ja projekteissa on yleisesti ottaen hallittu riskejä, ainakin jollain tasolla. Olisi kuitenkin mielenkiintoista selvittää, mitkä ovat toimivimmat riskienhallintatavat ja miten niitä Suomessa käytetään, jos käytetään.

Lähteet

1ClickFactory. Microsoft Dynamics AX Upgrade. Luettavissa: http://www.1clickfactory.com/microsoft-dynamics/upgrade/microsoft-dynamics-ax-upgrade. Luettu: 24.9.2013

Andersson, J. 6.6.2012. Toimitusjohtaja. iVersum Oy. Haastattelu. Espoo.

Atlassian. JIRA. Luettavissa: https://www.atlassian.com/software/jira. Luettu: 24.9.2013

Babic, V. 2009. Top 7 reasons why to avoid (much) customization. Luettavissa: http://vjeko.com/blog/top-7-reasons-why-to-avoid-much-customization. Luettu: 24.9.2013

Beaubouef, B. 2012. Developing a Business Case for ERP Customizations. Luettavissa: http://gbeaubouef.wordpress.com/2012/01/29/business-case-for-erp. Luettu: 24.9.2013

Berg, T. 2013. Viisi askelta onnistumiseen. Luettavissa: http://www.tietoviikko.fi/cio/blogit/CIO_100_blogi/viisi+askelta+onnistumiseen/a900569. Luettu 24.9.2013

Era, H. 21.8.2012. Johtava konsultti. eCraft Oy. Haastattelu. Espoo.

Feldman, J., 2012. Project Management Gets Lean. InformationWeek. Luettavissa: http://www.informationweek.com/software/project-management-gets-lean/232600005?ct=1022. Luettu: 24.9.2013

Gibbons, L. 2012. Manufacturing IT Best Practices: ERP Upgrades and Migrations. Luettavissa: http://docs.media.bitpipe.com/io_10x/io_104090/item_516116/manufacturing%20IT%20Best%20Practices_v4.pdf. Luettu 24.9.2013

Henriksson-Vainio, T. 20.6.2012. Vanhempi konsultti. Evry Business Solutions Oy. Haastattelu. Vantaa.

Herrig, K. & Scanlon, S. 2012. 10 critical ERP upgrade mistakes. Luettavissa: http://www.techrepublic.com/blog/10things/10-critical-erp-upgrade-mistakes/3202. Luettu: 24.9.2013

Kestilä, J. 12.6.2012. Toimitusjohtaja. 1ClicFactory Finland Oy. Haastattelu. Vantaa.

Laine, T. 26.6.2012. Vanhempi konsultti. Tmi Lainetar. Haastattelu. Helsinki

Lech, P. 2004. 80/20 Rule in ERP System Implementation – A case Study on Maximizing ROI. Tieteellinen julkaisu. University of Gdańsk. Gdańsk. Luettavissa: http://www.przemyslawlech.info.pl/index_pliki/Lech_80_20_rule_maximizing_ROI.pdf. Luettu: 24.9.2013

Microsoft a. Sure Step Online. https://mbs2.microsoft.com/Surestep/default.aspx. Luettu 20.10.2013

Microsoft b. Team Foundation Server. http://www.microsoft.com/visualstudio/eng/products/visual-studio-team-foundation-server-2012. Luettu: 24.9.2013

Myllymäki, R., Hintikka, T., Dahlberg, T. & Uimonen, B. 2010. Miksi tietojärjestelmäprojekti epäonnistuu? Helsinki.

Niemelä, J. 13.6.2012. Partner Account Manager. Microsoft Finland Oy. Haastattelu. Espoo.

Nigel Frank International. Microsoft Dynamics Salary Survey 2012. Luettavissa: http://www.nigelfrank.com/en/salarySurvey. Luettu 24.9.2013

Oracle. 2012. Upgrading or Rip and Replace you ERP: Business Consideration for JD Edwards EnterpriseOne Customers. Luettavissa: http://www.oracle.com/webfolder/technetwork/tutorials/jdedwards/White%20Papers/JDE%20E1%20WP-1101%20ERP%20Upgrade%20or%20Reimplementation%20Analysis.pdf. Luettu 24.9.2013

Phillips, S. 2010. ERP Software: Are Modification Always a Bad Idea? Luettavissa: http://www.r3now.com/erp-software-are-modifications-always-a-bad-idea. Luettu: 24.9.2013

Robertson, J. 2011 Make the right decision – apply strategy to ERP. Luettavissa: http://www.theskillsportal.com/human-resources/articles/631-make-the-right-decision-apply-strategy-to-erp.html. Luettu: 24.9.2013.

SearchManufacturingERP. 2010. FAQ: ERP upgrade best practices. Luettavissa: http://searchmanufacturingerp.techtarget.com/news/2240020891/FAQ-ERP-upgrade-best-practices. Luettu: 24.9.2013

Shepherd, J. 2007. Reduce the Pain of ERP Upgrades With Better Planning. AMR Research. Boston.

Stackpole, B. 2008. Five signs it's time for an ERP system upgrade. Luettavissa: http://searchmanufacturingerp.techtarget.com/news/1342896/Five-signs-its-time-for-an-ERP-system-upgrade. Luettu: 24.9.2013

Sundström, J. 12.6.2012. Vanhempi konsultti. Daxmate Oy. Haastattelu. Vantaa.

Talentum, 2005. Tietojärjestelmän hankinta, ohjelmistotoimittajan ja -ratkaisun valinta. 2. uudistettu painos. Helsinki.

Weiss, T. 2011. Customize You ERP or Adapt To It? What's Your Strategy? Luettavissa: http://www.cio.com/article/686202/Customize_Your_ERP_or_Adapt_To_It_What_s_Your_Strategy_. Luettu: 24.9.2013

Liitteet

Liite 1. Keskeiset käsitteet

Business case

Kertoo mitä hyötyjä tai riskejä jonkin asian tai projektin tekemisestä on liiketoiminnan kannalta.

Dynamics AX

Microsoft Dynamics AX on Microsoftin tekemä toiminnanohjausjärjestelmä.

Fit-gap

Vertailu halutun ja vakiotoiminnallisuuden välillä.

Integraatio

Integraatio tarkoittaa kahden erillisen järjestelmän yhdistämistä tai keräämistä yhdeksi kokonaisuudeksi.

JIRA

JIRA on Atlassianin tekemä projektien seurantaohjelmisto.

Järjestelmätoimittaja

Yhtiö joka edustaa toiminnanohjausjärjestelmän tehnyttä yhtiötä asiakkaalle. Tekee asiakaskohtaisia järjestelmämuutoksia.

Järjestelmävalmistaja

Toiminnanohjausjärjestelmän tehnyt yhtiö, kuten Microsoft, SAP tai Oracle.

Kick-off

Kick-off tilaisuus on yleensä projektin aloituspalaveri tai juhlatilaisuus jossa projekti julistetaan alkaneeksi.

MSDN (Microsoft Developer network)

Microsoftin kehittäjäverkko jota Microsoft ylläpitää ja hallinnoi. Se antaa teknisiä suosituksia ja neuvoja.

Sertifiointi

Toimenpide jossa valvotussa tilanteessa annetaan näyttö osaamisesta ja josta onnistuneen suorituksen jälkeen saadaan todistus eli sertifikaatti.

Standardi

Standardi on jonkin organisaation esittämä määritelmä siitä, miten jokin asia tulisi tehdä.

Sure Step -metodologia

Sure Step on virallinen metodologia Microsoft Dynamics-tuotteiden käyttöönottoon ja päivittämiseen. Sure Step kertoo, mitä kukin tekee ja missä järjestyksessä, sekä kuka vastaa mistäkin. Sure Step tukee useita eri projektityyppejä, joita voi tarvittaessa yhdistellä.

Toiminnanohjausjärjestelmä (ERP)

Toiminnanohjausjärjestelmä on yrityksen keskeisin tietojärjestelmä, joka yleensä koskettaa yrityksen kaikkia toimintoja. Sillä voidaan ohjata mm. kirjapitoa, reskontraa, varastohallintaa, tuotantoa, huoltoa, laskutusta ja projekteja. Toiminnanohjausjärjestelmille on tyypillistä kaikkien siinä olevien moduulien saumaton ja reaaliaikainen yhteistoiminta. Ne pyrkivätkin tehostamaan yritysten toiminta poistamalla manuaalisia työvaiheita.

Regressiotestaus

Regressiotestaus on mitä tahansa ohjelmistotestausta, jossa pyritään löytämään piileviä ohjelmistovirheitä, joita on voinut syntyä, kun järjestelmää on tuotu jokin uusi ominaisuus, muutos tai korjaus. Siinä testataan myös niitä ohjelmiston osia, joita uuden koodin ei pitäisi koskettaa, mutta virheellisesti saattaa niin tehdä. Yleinen tapa tehdä regressiotestausta on tehdä vanhoja testejä uudestaan, jolloin epätoivotut muutokset on helpompi löytää.

Räätälöinti

Räätälöinti on ohjelmiston muokkaamista asiakaskohtaisesti, yleensä järjestelmätoimittajan toimesta.

Parametrointi

Järjestelmän asetusten laittaminen haluttuun tilaan. Tehtävä, jonka yleensä järjestelmätoimittaja suorittaa asiakkaan kanssa tehtyjen määrittelyjen pohjalta.

POC (Proof of concept)

Tapa todistaa ratkaisun käyttökelpoisuus, esittelemällä etukäteen sovittua käyttötapausta vakioympäristössä ja kertomalla lisäksi mihin kohtiin prosessia tarvitaan asiakaskohtaisia muutoksia. On myös osa Sure Step –metodologiaa.

Pääomantuotto (ROI)

Pääomantuottolaskelma on tapa mitata investoinnin kannattavuutta. Pääomantuotto lasketaan kaavalla: pääomantuotto (%) = nettovoitto (eur) / investointi (eur) * 100. Osa tuotosta voi olla myös muuta kuin rahallisesti mitattavaa, kuten käyttäjätyytyväisyys.

Team Foundation Server (TFS)

Microsoftin Team Foundation Server eli TFS on järjestelmä, joka auttaa hallitsemaan ohjelmistojen kehitystä.

Vertikaali

Toimialaratkaisu. Kolmannen osapuolen tietylle toimialalle tekemä tuotteistettu komponentti tai järjestelmämuutos, jolle annetaan käyttäjätukea ja josta maksetaan lisenssimaksuja.

Liite 2. Haastattelukysymykset

Kerro oman kokemuksesi perusteella, miksi toiminnanohjausjärjestelmien päivitysprojektit epäonnistuvat tai onnistuvat?

Millaisia seuraamuksia päivitysprojektien epäonnistumisilla on ollut taloudellisesti tai muuten?

Miten olette saaneet vanhaan versioon tehdyt järjestelmämuutokset siirrettyä järkevimmin uuteen versioon?

Onko Teillä käyttökelpoisia toimintatapoja / standardeja järjestelmämuutosten siirtämiseen ja jos on, niin mitä ne ovat?

Järjestelmämuutoksia ei kannata siirtää kokonaisuutena, vaan yksitellen arvioiden. Kuinka olette hallinneet monitahoista järjestelmämuutosten kokonaisuutta?

Onko Teillä käytössä järjestelmämuutosten hallintaan työvälineitä ja jos on, niin mitä ne ovat?