HAAGA-HELIA
ammattikorkeakoulu

Middleware-tuotteen versionvaihtoprojektin suunnittelu jatkuvaa palvelun parantamista toteuttaen

Sanna Lassila

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2013

HAAGA-HELIA ammattikorkeakoulu

Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

12 April 2013

Tekijä tai tekijät Sanna Lassila	**Ryhmätunnus tai aloitusvuosi** 2010
Raportin nimi Middleware-tuotteen versionvaihtoprojektin suunnittelu jatkuvaa palvelun parantamista toteuttaen	**Sivu- ja liitesivumäärä** 41 + 3
Opettajat tai ohjaajat Immo Hahtola, Heikki Hietala	

Opinnäytetyön tavoitteena oli selvittää tietoteknisten kehitys- ja ylläpitoprojektien välisiä eroja suunnitteluvaiheessa ja perehtyä erityisesti Middleware-järjestelmien (suom. väliohjelmisto) versionvaihtoprojektien suunnitteluun. Työssä pyritään kuvaamaan parhaita käytäntöjä liittyen Middleware-järjestelmien versionvaihtoprojektien suunnittelun jatkuvan palvelun parantamisen prosessin mukaan. Työssä keskitytään projektin suunnitteluun, mm. toteutustavan suunnitteluun, mutta itse projektin toteutuksen hallinta on jätetty pois työn tavoitteista.

Työssä perehdytään ensin projektin suunnittelun perusteisiin, sekä kuvataan eroja tietoteknisten kehitys- ja ylläpitoprojektien välisessä suunnittelussa. Seuraavaksi perehdytään Middleware-järjestelmien sijoittautumiseen palvelukeskeistä arkkitehtuuria toteuttavassa ympäristössä, sekä versionhallintatyökalujen käyttöön versionvaihtoprojekteissa. Esimerkkitapauksena kuvataan WebSphere Message Broker-tuotteen versionvaihtoprojektin suunnittelu Asiakkaan integraatiokeskuksen tiedonvälityspalvelussa.

Opinnäytetyö toteutettiin kvalitatiivisena tapaustutkimuksena ja kirjoitettiin helmimaaliskuussa 2013. Esimerkkitapausprojektin suunnittelu tehtiin syksyllä 2012 ja projektia toteutettiin samanaikaisesti työn kirjoituksen aikaan.

Opinnäytetyön tuloksena syntyi jatkuvan palvelun parantamisen mukainen malli ylläpitoprojektien suunnitteluun. Työn viimeisissä kappaleissa esitetään toimenpide-ehdotuksia tulevien versionnostoprojektien suunnitteluun ja pohditaan työn merkitystä tekijälle ja hänen organisaatiolleen.

Asiasanat
Projektin suunnittelu, versionnosto, jatkuva palvelun parantaminen, integraatio, versionhallinta

HAAGA-HELIA
University of Applied Sciences

Abstract

12.4.2013

Information technology

Authors Sanna Lassila	**Group or year of entry** 2010
The title of thesis Middleware Product Migration Project Planning Implementing the Continual Service Improvement Process	**Number of report pages and attachment pages** 41 + 3
Advisor(s) Immo Hahtola, Heikki Hietala	

The aim of the thesis was to clarify the differences in planning phase between development and maintenance IT-projects and to especially focus on the planning of middleware migration projects. This thesis aims to describe best practices relating to planning of middleware's migration projects by the process of continual service improvement. This thesis focuses on project planning, including the planning of implementation, but the implementation itself is left out of the thesis scope.

This thesis first familiarizes with the basics of project planning and describes the differences in the planning of development and maintenance projects. Next this thesis describes how middleware systems are situated in the IT-environment implementing service orientated architecture and how version control tools are to be used in migration projects. An example case is described about WebSphere Message Broker product's migration project's planning in a customer's integration centers data exchange service.

This thesis was carried out as a qualitative case study and written in February and March 2013. The case projects' planning was done in the autumn of 2012 and the migration project itself was in the making while writing this thesis.

As an outcome of the thesis a model implementing the process of continual service improvement for maintenance projects planning was developed. In the last chapters of this thesis some proposals for future project planning are presented as well as the value of this thesis to the author and her organization is considered.

Key words
Project planning, migration, continual service improvement, integration, version control

Sisällys

1 Johdanto 1

2 Projekti ja sen suunnittelu 3

2.1 Projektin määritelmä 3

2.2 Projektin suunnittelu 4

2.3 Projektin suunnittelun tehtävä 5

2.4 Projektin suunnittelun vaiheet 5

2.4.1 Projektin tulostavoitteiden määrittely 6

2.4.2 Projektin organisaatio 6

2.4.3 Projektin ositus 8

2.4.4 Projektin aikataulutus 8

2.4.5 Projektin kustannussuunnittelu 9

2.4.6 Ohjausjärjestelmän suunnittelu 9

2.5 Erilaisia IT-projekteja 10

2.6 Ylläpitoprojektin erot kehitysprojektiin verrattuna 11

2.6.1 Ylläpitoprojektin määrittelyt 12

2.6.2 Todennus ylläpitoprojektissa 12

2.7 Miksi ylläpitoa? 13

3 Middleware-järjestelmät palvelukeskeisen integraatio-alustan perustana 14

3.1 Middleware-järjestelmät SOA-palveluiden tarjoajina 14

3.2 Middleware-järjestelmien ylläpito 14

3.3 Versionhallintajärjestelmä ylläpidon työkaluna 15

4 Case: Digia jatkuvan palvelun toimittajana Asiakkaan integraatioympäristössä 17

4.1 Asiakkaan tiedonvälityspalvelu 17

4.2 Middleware-tuotteet tiedonvälityspalvelua tukemassa 18

4.3 Jatkuva palvelu Asiakkaan tiedonvälityspalvelussa 19

4.3.1 Jatkuvan palvelun käytännöt integraatiokeskuksen ylläpidossa 19

4.3.2 Integraatioiden hallintamalli 22

4.4 Jatkuvan palvelun parantamisen mallin mukainen projektisuunnittelu integraatiokeskuksen ylläpidossa 23

4.5 WebSphere Message Broker-tuotteen versionvaihto-projekti Asiakkaan integraatioympäristössä 23
5 Middleware-järjestelmän versionvaihdon suunnittelu palvelua jatkuvasti parantaen 27
5.1 Ylläpitoprojektin suunnittelumalli jatkuvan palvelun parantamisen prosessin mukaan 27
5.2 Middleware-järjestelmän versionvaihto-projektin suunnittelu 28
5.2.1 Projektin tavoitteen asettaminen 28
5.2.2 Nykytilan kartoittaminen 29
5.2.3 Toteutustavan, resurssien ja aikataulun suunnittelu 29
5.2.4 Projektisuunnitelma Middleware-järjestelmän versionvaihtoprojektissa 32
5.2.5 Projektin seuranta projektisuunnitelman perusteella 32
6 Yhteenveto ja toimenpidesuositukset 33
7 Johtopäätökset ja jatkokehitysehdotukset 36
7.1 Johtopäätökset 36
7.2 Jatkokehitysehdotukset 37
Lähteet 39
Liitteet 42
Liite 1. WebSphere Message Broker-versionvaihtoprojektin sisällysluettelo. 42

1 Johdanto

Tietoteknisissä organisaatioissa kehitystä ja ylläpitoa toteutetaan yleensä projektimuotoisesti. Varsinkin suurten yritysten liiketoiminnan hallinta nojaa vahvasti tietotekniikkaan, joka vaatii sovellusten ja järjestelmien kehitystä ja ylläpitoa. Tietotekniset kehitys- ja ylläpitoprojektit noudattavat projektin suunnittelun osalta pääosin samaa kaavaa, mutta suunnittelun ja toteutuksen painopisteet eroavat hieman. Tämän opinnäytetyön tavoitteena on kuvata Middleware-tuotteen versiopäivityksen suunnitteluprosessi jatkuvan palvelun parantamisen mukaan samalla vastaten seuraaviin kysymyksiin:

- Mitä asioita tulee ottaa huomioon integraatiotuotteen versionvaihtoprojektin suunnittelussa?
- Miten integraatiotuotteen versionvaihtoprojekti eroaa käyttöönottoprojektista?
- Miten versionvaihtoprojektit kannattaa viedä läpi?
- Miten versionvaihtoprojekti kannattaa aikatauluttaa?
- Miten versionvaihtoprojekti kannattaa jakaa osiin?

Tämän tutkimuksen ”Projekti ja sen suunnittelu” kappaleessa kuvataan projektisuunnittelun keskeisiä vaiheita ylläpitoprojektien erityispiirteitä.

Kappaleessa 3 kuvataan Middleware-järjestelmien (suom. väliohjelmisto) sijoittautumista, toimintaa ja ylläpitoa palvelukeskeistä arkkitehtuuria toteuttavassa ympäristössä. Palvelukeskeinen arkkitehtuuri on tällä hetkellä suurissa sovellusympäristöissä laajasti käytetty sovellusten arkkitehtuuriratkaisu. Karkeasti voidaan kuvata, että palvelukeskeinen arkkitehtuuri kerrostaa käyttöliittymätoteutukset omaksi kerroksekseen, kuten myös sovelluslogiikan (palvelut) ja tietovarastot. Sovelluskeskeisessä arkkitehtuurissa palveluita tarjotaan käyttöliittymäkerrokselle yhtenäisen rajapinnan kautta. Middleware-järjestelmät ovat tämän palvelukerroksen keskiössä olevia palvelukeskeistä arkkitehtuuria toteuttavia tuotteita, jotka tarjoavat sovelluksille mm. yhtenäisen ajoympäristön ja rajapinnat kutsuja varten.

Middleware-järjestelmiä, kuten muitakin sovelluksia ja järjestelmiä tulee päivittää ja ylläpitää, jotta ne pysyvät suorituskykyisinä ja tuen piirissä. Middleware-järjestelmiin liit-

tyy paljon integraatioita muihin sovelluksiin, joka vaatii mm. kattavan todennuksen. Middleware-järjestelmien ylläpitoprojektit mielletään tästä syystä usein raskaiksi ja kalliiksi rupeamiksi, minkä takia niiden versiot pyritään pitämään mahdollisimman pitkään ennallaan. Tässä tutkimuksessa pyritään löytämään keinot Middleware-järjestelmien mahdollisimman tehokkaaseen ja näkymättömään ylläpitoon suunnittelemalla ylläpidon toteutus mahdollisimman vähän muuta ympäristön kehitystä häiritseväksi.

Tutkimuksen kappale 4 kuvaa Digian prosesseja ja toimintaa Asiakkaan integraatiokeskuksen tiedonvälityspalvelun ylläpitäjänä. Asiakkaan integraatiokeskus koostuu erilaisista integraatiojärjestelmistä, jotka muuntavat ja välittävät eri sovellusten tuottaman informaation lopulliselle vastaanottajalle. Digia kehittää ja ylläpitää Asiakkaan integraatiokeskusta jatkuvan palvelun parantamisen menetelmillä. Jatkuvan palvelun parantamista toteutetaan määrittelemällä palvelulle mittareita, seuraamalla näitä mittareita, toteuttamalla palveluun muutoksia mittausten tuottamien tulosten pohjalta, mittaamalla uudelleen jne. Jatkuvan palvelun parantamiseen kuuluvat myös ennalta määritellyt ja määrämuotoiset kokoukset.

Neljännen kappaleen lopussa kerrotaan keväällä 2013 toteutettavan Broker-järjestelmän versionvaihto-projektin suunnittelusta. Tätä tutkimusta kirjoitettaessa projekti oli vielä kesken, mutta muutamia kehittämispaikkoja projektin suunnittelussa ja läpiviennissä oli jo havaittavissa. Kappaleessa 5 on kuvattu jatkuvan palvelun parantamisen mukainen ylläpitoprojektin suunnittelumalli ja avattu mallia Middleware-järjestelmän versionnostoprojektia silmällä pitäen.

Lopuksi esitellään toimenpidesuositukset Asiakkaan integraatiokeskukselle tulevia ylläpitoprojekteja varten, pohditaan tämän tutkimuksen merkitystä sekä ehdotetaan jatkotutkimusaiheita.

2 Projekti ja sen suunnittelu

Projektimallinen toiminta ulottuu kaikkeen tuottavaan liiketoimintaan. Projektimallisen toiminnan etuna on mm. suunnitelmallisuus, seurattavuus ja projektiryhmän sitouttaminen. Yritysten tietotekniikkahankkeissa lähes kaikki kehitys on projektimallista. Tietotekniikka-alan projektit eroavat muiden alojen projekteista mm. siten, että tietotekniikan alalla liiketoiminnan vaatimukset ja teknologiat muuttuvat jatkuvasti. Myös sidosryhmien odotukset kasvavat jatkuvasti teknologian kehittyessä. Tietotekniikkaprojektien tuotokset ovat usein joustavampia muutokselle, sekä myös näkymättömämpiä loppukäyttäjille kuin esim. rakennusalan projektit. (Kelkar 2013, 47.)

2.1 Projektin määritelmä

Projekti määritellään kokonaisuudeksi yhteistä päämäärää kohti tähtääviä tehtäviä, jotka tulee suorittaa tiettyjen raamien puitteissa. Tällaisia raameja ovat mm. aikataulu ja budjetti. Projekti voidaan karkealla tasolla jaotella kolmeen vaiheeseen: projektin aloitukseen, toteutukseen ja lopetukseen. (Malmi & Åkerlund 2013.)

PMBOK® Guide määrittelee projektin tilapäiseksi haasteeksi, joka luo yksilöidyn tuotteen, palvelun tai tuloksen. Määritelmä tilapäisyydestä korostaa, että haasteella on alku ja loppu. Projekti on päättynyt kun tarvetta projektille ei enää ole tai sen tavoitteita ei voi enää saavuttaa. (Project Management Institute 2008, 5.)

Ohjelmisto- ja ei-ohjelmistoprojektin välille on vaikeata määritellä tarkkaa rajaa. Yleisesti ohjelmistoprojektiksi mielletään koodia tuottava tai ylläpitävä projekti, mutta ohjelmistoprojekteihin on laskettavissa myös projektit jotka kohdistuvat vaatimusmäärittelyihin, suunnitteludokumentteihin, testitapauksiin, ohjeisiin ym. ohjelmistoon liittyviin dokumentteihin. (Tsui 2004, 2.)

Tässä työssä käsitellään projektin suunnittelua, joka on osa projektin aloitusta ja huipentuu ohjausryhmän hyväksymään projektisuunnitelmaan.

2.2 Projektin suunnittelu

Projektin suunnittelu on osa projektin hallintaa. PMBOK® Guide on tiivistänyt projektin hallinnan määritelmän seuraavasti:

> "Project management is the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements."

Suomennettuna projektin hallinta on tiedon, taidon, työkalujen ja tekniikoiden hyödyntämistä projektin tavoitteiden saavuttamiseksi. (Project Management Institute 2008, 6.)

Projektin huolellinen suunnittelu on tärkeää, koska sillä voidaan ehkäistä ongelmia, joilla voi olla suuria vaikutuksia projektin valmistumiseen ja aikatauluun. Heikko suunnittelu aiheuttaa myöhästymisiä aikatauluista, ylitöitä, keskeneräisiä lopputuloksia, ongelmia resurssien käytössä ja pahimmassa tapauksessa koko projektin keskeytymisen. (Pelin 2008, 83–84.)

Harold Kerzner esittää kirjassaan "Project management", että projektin kokonaistyömäärästä noin 45 % kuluu suunnitteluun. Projektin suunnittelu on projektipäällikön vastuulla. Koko projektiryhmän, tai ainakin projektin avainhenkilöiden, tulee osallistua suunnitteluun. Suunnittelussa käydään läpi erilaisia vaihtoehtoja toteutukselle, joista ohjausryhmä, projektipäällikön esityksen pohjalta, valitsee sopivimman suhteessa aikatauluun ja kustannuksiin. Tehokas suunnittelu on verrattain nopeaa ja määrämuotoista. Projektin suunnittelun tulisi olla koko projektin ajan kestävä iteratiivinen prosessi. (Kerzner 2009, 411–420.)

Pahimmassa tapauksessa projektin huono suunnittelu ja sitä kautta projektin keskeyttäminen saattaa aiheuttaa jopa koko yrityksen toiminnan loppumisen. Usein keskeytyneen projektin taustalla on projektin heikko ja puutteellinen suunnittelu ja projektin keskeytymisen syyt ovat olleet olemassa jo ennen projektin varsinaista aloitusta. Ahosen ja Savolaisen tutkimus paljastaa, että keskeytyneen projektin taustalla on usein puutteita erityisesti projektin tavoitteiden määrittelemisessä ja toisaalta asetettujen tavoitteiden ymmärtämisessä. (Ahonen & Savolainen 2010.)

2.3 Projektin suunnittelun tehtävä

Projektin suunnittelulla etsitään parasta toteutustapaa projektille. Projektin suunnittelun lopputuloksena syntyy projektisuunnitelma, joka on keskeinen työkalu koko projektin ajan. Projektisuunnitelmaa päivitetään muutoshallinnan prosessien kautta aina kun projektisuunnitelmassa kuvattuihin asioihin kohdistuu muutoksia. Projektisuunnitelmassa tulee kuvata ainakin projektin tavoite, realistinen toteutussuunnitelma, projektin toimeksianto ja osapuolten valtuudet. (Pelin 2008, 83–84.)

Projektin edetessä projektisuunnitelmaa käytetään viitteenä siitä mitä projektissa on sovittu tehtävän. Hyvä projektisuunnitelma vastaa kysymyksiin: Mitä tehdään? Miksi tehdään? Miten tehdään? Missä tehdään? Milloin tehdään? Kuka tekee? Näihin kysymyksiin vastaaminen selvittää mm. projektin vaatimukset, aikataulun, kustannukset, resursoinnin ja johtamisen. Projektisuunnitelmassa on hyvä kuvata myös projektissa käytettävät menetelmät ja prosessit niin työskentelymenetelmien, muutoksen hallinnan, kuin tiedotuksen osalta. Projektin aikana syntyviä konfliktitilanteita ja väärinkäsityksiä voidaan ehkäistä, kun projektin käytännöt ja prosessit ovat kaikille osapuolille selvillä. (Kerzner 2009, 461.)

Kaikissa ohjelmistokehitykseen liittyvissä projektisuunnitelmissa tulisi myös kuvata minkä tyyppisestä projektista on kyse ja mitkä ovat projektin halutut tuotokset liittyen toiminnallisuuksiin ja dokumentteihin. Projektisuunnitelmassa tulee esittää projektin aikataulu ja projektin seurantapisteet, sekä projektin tarvitsemat resurssit ja niiden kustannukset. Projektisuunnitelmassa kuvataan myös projektiin liittyvät riskit mahdollisimman tarkalla tasolla. (Tsui 2004, 18–19.)

2.4 Projektin suunnittelun vaiheet

Projektin suunnitteluvaihe on osa projektin aloitusta, joka huipentuu ohjausryhmän hyväksymään projektisuunnitelmaan. Toteutusvaiheessa suunnitelmia joudutaan usein täsmentämään tiedon karttuessa. Projektien suunnittelussa tulisi etsiä vastauksia kysymyksiin: Kuka? Mitä? Milloin? Miten? Minkä verran? Usein jo projektin suunnitteluvai-

heessa havaitaan useita potentiaalisia ongelmia, jolloin niille etsitään vaihtoehtoisia ratkaisuja ja näistä vaihtoehdoista valitaan paras toteutustapa (Pelin 2008, 85-88.)

Projektien suunnittelussa voidaan tunnistaa seuraavat yleiset vaiheet, jotka on kuvattu kappaleissa 2.4.1–2.4.6.

2.4.1 Projektin tulostavoitteiden määrittely

Projektin lähtökohtana on sen tavoitteiden ja tehtävän määrittely. Ilman tavoitetta ei ole projektia. Ennen projektin tehtävän asettamista on hyvä selvittää taustoja, mitä on mahdollisesti jo tehty ja miksi projektiin pitää ryhtyä. Tavoitteiden määrittelyvaiheessa projektipäällikkö kokoaa projektin tehtäväluettelon yhdessä projektiryhmän jäsenten kanssa. Jokaiselle tehtävälle on nimettävä vastuuhenkilö. Projektin tavoitteita määriteltäessä asetetaan projektille selkeät ja mitattavat tavoitteet, kuten aikataulutavoite, kustannustavoite, tulostavoite ja tekniset tavoitteet. Projektia on hyvä mitata myös esim. toteutuksen edistymisellä ja laadulla, testauksen edistymisellä sekä määrittelyiden kattavuudella. Projektipäällikkö on vastuussa asetettujen mittareiden seurannasta. (Pelin 2008, 88.)

Projektin määrittelyvaiheessa kuvataan sovelluksen tai järjestelmän halutut piirteet ja toiminnallisuudet. Määrittelyiden pohjalta projektille arvioidaan tavoiteaikataulu. Määrittelyiden oikeellisuudesta vastaa loppukädessä projektin omistaja. Määrittelyt voivat olla joko toiminnallisia tai ei-toiminnallisia. Projektin määrittelyvaihe eroaa sovelluksen tai järjestelmän määrittelystä, jossa sovelluksen tai järjestelmän toiminnallisuudet kuvataan tarkalla tasolla ja joka on osa projektin toteutusvaihetta. (Kerzner 2004, 611.)

Projektin tavoitetta ja tehtävää määriteltäessä otetaan kantaa myös siihen, mitä projektissa ei tehdä. Projektista rajataan ulkopuolelle projektia rajoittavat ulkoiset tekijät, sekä kohdealueen ulkopuolelle jäävät tehtävät. (Pelin 2008, 88.)

2.4.2 Projektin organisaatio

Projektisuunnitelmassa kuvataan projektin organisaatio. Projektin käynnistäjänä toimii projektin asettaja tai omistaja, joka päättää projektin aloittamisesta ja toimii projektin

maksajana. Projektin omistaja myös nimeää projektin ohjausryhmän ja vastaa viime kädessä siitä, että projektilla on tarvittavat resurssit käytössä. Ohjausryhmä on projektin korkein päättävä elin, joka koostuu projektiin liittyvien eri organisaatioiden edustajista. Projektin ohjausryhmä pyritään pitämään kooltaan mahdollisimman pienenä ja tehokkaana. Ohjausryhmä mm. nimeää projektille projektipäällikön, hyväksyy projektisuunnitelman ja projektin tuloksen, sekä päättää projektin lopettamisesta. (Malmi & Åkerlund 2013.)

Projektipäällikkö on kokonaisvastuussa projektissa. Projektipäällikkö laatii projektisuunnitelman, ohjaa projektiryhmän työskentelyä ja raportoi ohjausryhmälle projektin edistymisestä. Projektipäällikkö huolehtii projektin dokumentoinnista ja laatii loppuraportin, sekä päättää projektin. Projektiryhmän jäsenet ovat yleensä omien alojensa asiantuntijoita tai sellaisiksi tähtääviä. Projektiryhmän jäsenet osallistuvat projektisuunnitelman laadintaan omien työtehtäviensä osalta ja raportoivat edistymisestään projektipäällikölle. Projektissa voi olla myös projektiassistentti, joka hoitaa sovitun osan projektipäällikön tehtävistä. (Pelin 2008, 68-70.)

Projektin tehtäväluettelon perusteella suunnitellaan projektissa tarvittavat roolit. Ensin määritellään roolit, joihin resursseja tarvitaan ja roolien tehtäväkuvauksien pohjalta haetaan tehtäville sopivat tekijät. Projektin omistajan vastuulla yhdessä ohjausryhmän kanssa on varmistaa tarvittava resurssien saatavuus projektille. Moniprojektiympäristössä samat resurssit ovat useiden projektien käytössä, jolloin muutokset yhden projektin aikataulussa heijastuvat myös muihin projekteihin. Resurssitarpeen suunnittelussa on kaksi vaihetta: resurssilaskenta ja resurssitasaus. Resurssilaskenta näyttää työmääräarvioiden pohjalta tehdyn kuormituksen, joka harvoin kuitenkaan toteutuu. Resurssitasaus perustuu siihen, että kuormitus aloitetaan tärkeimmistä resurssilajeista ja muita tehtäviä ja resursseja säädetään sen perusteella miten resursseja on käytettävissä muussa toimintaverkossa. (Pelin 2008, 145–149.)

Erään (Fleurey ym. 2007) ranskalaispankin koko sovelluskehitysympäristön migraatiota käsittelevässä artikkelissa todetaan, että migraatioprojektissa avainresursseina ovat lähtö- ja tavoitealustan tai –sovellusten asiantuntijat, sekä migraatiotyökalujen asiantuntijat. Ympäristöjen migraatioprojektien organisaation suunnittelussa tulee suunnitella ja ku-

vata asiantuntijaroolit lähtö- ja tavoitealustojen hallinnointiin, sekä itse migraation ja sen työkalujen hallinnointiin, kehittämiseen ja todennukseen.

2.4.3 Projektin ositus

Projekti tulee vaiheistaa pienempiin, itsenäisesti suunniteltaviin ja toteutettaviin kokonaisuuksiin eli projekti tulee osittaa. Projektin ositus helpottaa projektin hallittavuutta sekä tulosten ja kustannusten seurantaa. Projektin osat aikataulutetaan erillisiksi osa-aikatauluiksi ja niiden keskinäiset riippuvuussuhteet kuvataan. Projekti voidaan jakaa edelleen pienempiin osaprojekteihin, joista jokainen tuottaa oman tuloksensa. Projektit voidaan osittaa eri vaiheiden perusteella (suunnittelu, toteutus, käyttöönotto), systeemeittäin eri järjestelmien perusteella tai eri työlajien perusteella (suunnittelu, toteutus, hallinto, testaus, käyttöönotto). (Pelin 2008, 93–95.)

Jokaisen projektin vaiheen jälkeen katselmoidaan vaiheen aikaansaannokset projektiryhmän ja ohjausryhmän toimesta. Katselmoinneissa voidaan vaiheen tuotoksien läpikäynnissä käyttää apuna tarkistuslistoja. Ohjausryhmä hyväksyy vaiheen tuotokset ja antaa luvan jatkaa seuraavaan vaiheeseen. (Kerzner 2009, 418.)

2.4.4 Projektin aikataulutus

Projektin aikataulu laaditaan projektin tehtävien pohjalta. Projektipäällikkö laatii tehtäväluettelon perusteella aikataulun yhdessä jokaisen alueen asiantuntijan kanssa. Projektin edetessä tulee usein muutoksia ja aikataulua joudutaan muuttamaan. Aikatauluohjaus on projektin loppuun asti kestävä prosessi. Projektin tehtäväluettelon laatimiseen kannattaa panostaa, sillä suurimmat aikatauluvirheet johtuvat unohdetuista tehtävistä. (Pelin 2008, 110–115.)

Tehtävien työmäärien arviointi vaatii aikaa, työtä ja kokemusta. Työmäärien arvioinnissa tulee noudattaa todennäköisyysajattelua, jolloin arviointivirheet tehtävien aikatauluissa kumoavat toisensa. Arvioihin ei tule sisällyttää virhemarginaaleja, eikä niitä tule tehdä liian optimistisiksi. Työmääräarviot tehdään aina yhdessä tehtävän suorittajan kanssa. Mikäli suorittaja ei ole vielä tiedossa, annetaan arvio keskimääräisen suorittajan perusteella ja tarkennetaan myöhemmin. Vinkkejä työmääräarvioihin kannattaa ottaa edellis-

ten vastaavien projektien toteutuneista työmääristä. Tulevaisuuden vastaavia projekteja varten on tärkeätä kirjata ja tallettaa tiedot projektin toteutuneista työmääristä loppuraporttiin ja mahdolliseen kokemuskantaan. (Pelin 2008, 116–122.)

Ylläpitoprojekti ja sen varsinainen toteutusosuus kannattaa aikatauluttaa mahdollisimman lyhyeksi ajaksi. Järjestelmien versionvaihtoprojektit eivät yleensä tuota liiketoiminnalle näkyvää muutosta toiminnallisuuteen, jolloin pitkäkestoisen versionvaihtoprojektin uhkana on tällaisten näkyvien muutosten ottaminen osaksi versionvaihtoprojektia. Muutosten mukaan ottaminen puolestaan on omiaan viivästyttämään itse versionvaihdon toteutusta ja siten uhkaa koko projektin aikataulua. Mitä kauemmin projekti kestää, sitä vaikeampi on myös huolehtia projektin resursoinnista. (Teppe 2009.)

2.4.5 Projektin kustannussuunnittelu

Projektin kustannustavoite ohjaa usein projektin suunnittelua. Projektin kustannukset ovat myös yksi projektin tärkeistä seurattavista mittareista. Joskus ylläpitoprojekteissa kustannustavoite on projektin tärkein tavoite, mikäli ympäristöä lähdetään kustannussyistä päivittämään. Projektin aikataululla on suora yhteys projektin kustannuksiin. Mitä pidempi aikataulu on, sitä enemmän projektille tulee kustannuksia. Toisaalta mahdollisimman nopeasti suoritettava projekti vaatii enemmän kustannuksia useampien resurssien tarpeen vuoksi. Projektin kustannussuunnittelulla pyritään löytämään optimiratkaisu aikataulun ja kustannusten suhteen. Kustannusarviot annetaan tehtävittäin riittävällä tarkkuustasolla. Mikäli mahdollista, arvioinnin pohjana kannattaa käyttää edellisten vastaavien projektien tietoja toteutuneista kustannuksista. Projektin kustannusten suunnittelu päättyy aikaan sidottuun projektibudjettiin. Budjetin tekeminen on tärkeää, jotta projektin edistyessä voidaan valvoa kustannusten toteutumista suhteessa suunniteltuun. (Pelin 2008, 165–170.)

2.4.6 Ohjausjärjestelmän suunnittelu

Projektin ohjausjärjestelmän tehtävänä on valvoa ja ohjata projektia saavuttamaan projektille asetetut tavoitteet projektin budjetin ja aikataulusuunnitelman puitteissa. Ohjausjärjestelmän avulla kerätään tietoa, arvioidaan tilanne saadun tiedon perusteella, tehdään tarvittavat päätökset sekä välitetään toimenpideohjeet. Ohjausjärjestelmä pitää

sisällään niin projektin tiedotuksen kuin kokous- ja raportointikäytännöt. Ohjausjärjestelmän kunnollisen toiminnan kannalta on saadun informaation oltava luotettavaa. Ohjauskierroksen tulee myös sujua lyhyessä ajassa ja raportoinnin vaatiman työmäärän pysyä mahdollisimman pienenä. Ohjauksen on nostettava ongelmat selkeästi esille ja ne on pystyttävä ratkaisemaan ohjauksen piirissä. (Pelin 2008, 295–297.)

Erityisesti jatkuvan palvelun parantamisen prosessin kannalta tilanteen kartoitus suhteessa suunniteltuun on tärkeää. Tilannetta kartoitetaan erilaisilla ennalta määritellyillä mittareilla kuten projektin kustannukset tai projektin edistyminen suhteessa aikatauluun (esim. valmistuneet toiminnallisuudet tai hyväksytysti ajetut testitapaukset). Jatkuvan palvelun parantamisen prosessin mukaan ohjausjärjestelmän suunnitteluvaiheessa projektille asetetaan seurattavat mittarit sekä määritetään seurantapisteet ja -vastuut. (Office of Government Commerce 2007, 66–67.)

2.5 Erilaisia IT-projekteja

Tietotekniikkaprojekteja voidaan kategorisoida monella tavalla. Ne voidaan jakaa toistettaviin projekteihin tai vain kerran suoritettaviin ongelmanselvitys- tms. projekteihin. Projekteja voidaan jaotella massa- tai uniikkiprojekteihin. (Wysocki 2010, 5). Tietotekniikkaprojektit voidaan jakaa myös kehitys-, ylläpito- tai kolmannen osapuolen sovelluksen integrointiprojekteihin. Omana lukunaan ovat vielä erilaiset dokumentaatioprojektit, joissa pääpaino on dokumentaation tuottamisella, ei niinkään sovellusten toiminnan kehittämisellä.

Kehitysprojekteissa pääpaino on uusien sovellusten tuottamisessa. Ylläpitoprojekteissa tavoitteena on kehittää olemassa olevaa sovellusta luomalla siihen uusia ominaisuuksia tai päivittää ajossa olevaa sovellusympäristöä. Ylläpitoon ei tässä yhteydessä lasketa ns. sisällön ylläpitoa, jossa tavoitteena on tuottaa sisältöä esim. verkkosivustolle. Kolmannen osapuolen sovelluksen integraatio-projektit ovat projekteja, joissa jokin ulkoisen toimijan tuottama järjestelmä integroidaan osaksi yrityksen muuta ympäristöä esim. SAP tai muu vastaava toiminnanohjausjärjestelmä.

2.6 Ylläpitoprojektin erot kehitysprojektiin verrattuna

Projektin suunnittelun vaiheet ja sisältö ovat sekä uusien ohjelmistojen kehitysprojekteissa että jo olemassa olevien sovellusten ja ympäristöjen ylläpitoprojekteissa lähtökohtaisesti samanlaiset. Molemmissa projektityypeissä on tärkeää asettaa tavoitteet ja toisaalta rajata ulos sellaiset tekemiset, jotka eivät ole aloitettavan projektin vastuulla. Sovellusten ajoympäristön ylläpitoprojektien tavoitteena on jonkin sovelluksen tai järjestelmän teknisen version kohottaminen tai mahdollisesti koko sovelluskehitysalustan (Legacy -> J2EE) tai käyttöjärjestelmän vaihtaminen kokonaan uuteen (Windows -> AIX).

Ylläpitoprojekteissa ja erityisesti versionnostoprojekteissa keskitytään yleensä olemassa olevan toiminnallisuuden säilyttämiseen uudella teknisellä alustalla, samalla mahdollisesti lisäten uusia ominaisuuksia. Tällöin erityisen paljon painoarvoa tulee antaa tuotosten todennukselle, joka tulee huomioida projektin työmäärien suunnittelussa. Ylläpitoprojekteissa tulee huolellisesti suunnitella myös projektin toteutustapa päällekkäisen kehitystyön tai täydellisen kehitystyön tauon vaatiman ajanjakson minimoimiseksi. Utrechtin yliopiston tekemässä tutkimuksessa eräs haastateltu migraatioprojektin asiantuntija toteaa, että varsinaista migraatiotyötä tulisi toteuttaa iteroiden sillä migraatio on usein aikaa vievää toimintaa ja migraatioprosessin aikana alkuperäiseen versioon tehdyt muutokset tulee toteuttaa myös migroituun versioon. Projektin aikataulutuksessa tulee huomioida sovellusten keskinäiset suhteet ja niiden kehitysaikataulut, sekä pyrkiä osittamaan projekti loogisiin ja aikataulullisesti luonteviin kokonaisuuksiin. Menestyksekäs sovellusten ja ympäristön ylläpito vaatii teknisten edellytysten lisäksi myös yhteistyökykyvykkyyttä projektien ja sovellusten välillä. (Bennett & Rajlich 2000; Khadka, Reijnders, Saeidi, Jansen & Hage 2011.)

Ranskalainen Sodifrance-yhtiö on migraatioprojekteissaan todennut erityisesti pilottivaiheen tärkeyden. Pilottivaiheen tavoitteena on todentaa ja virittää migraatioprosessia ja migraatiotyökaluja, sekä toimia liiketoiminnalle tai asiakkaalle demonstraationa ja vertailukohteena migraation tuotoksille. Pilotoitavaksi osa-alueeksi kannattaa valita tarpeeksi laaja ja kattava osa migroitavaa aluetta. (Fleurey, Breton, Baudry, Nicolas & Jézéquel 2007.)

2.6.1 Ylläpitoprojektin määrittelyt

Ylläpitoprojektien suunnittelussa tulee antaa erityisen paljon painoarvoa toteutuksen ja todennuksen suunnittelulle. Ylläpitoprojektien suunnittelussa lähtötilanteen kartoitus on olennainen vaihe suunniteltaessa projektin lopullista toteutustapaa. Sovellukset, niiden toiminta ja ajoympäristö kuvataan ja mahdollisesti mallinnetaan. Tavoitteena on yleensä toimittaa vastaava toiminnallisuus uudessa ympäristössä kuin on ollut vanhassa ympäristössä. Fleureyn ym. artikkelissa koskien Sodifrancen suurelle ranskalaiselle pankille toimittamaa sovelluskehitysympäristön migraatiota esitetään toimivaksi kartoitusmalliksi ensin sovelluskoodien jäsentämistä ja seuraavaksi niiden takaisinmallinnusta esim. UML-malleiksi. (Fleurey ym. 2007).

2.6.2 Todennus ylläpitoprojektissa

Todennus on tärkeä osa projektia, jossa varmistetaan projektin tuotosten laatu suhteessa määrityksiin. Kustannusten vähentämiseksi projektissa pyritään löytämään automaation optimiaste, tämä koskee myös todennusta. Kehitysprojekteissa uutta toiminnallisuutta luotaessa testaus toteutetaan yleensä nimettyjen testaajien toimesta, jotka tekevät testitapaukset annettujen määritysten perusteella. Ylläpitoprojekteissa, kuten järjestelmäversioiden migraatioprojekteissa tehokkain tapa hoitaa todennus on ajaa automatisoituja testitapauksia. Testitapaukset luodaan olemassa olevassa lähtöympäristössä, niitä mahdollisesti muokataan uuteen ympäristöön sopiviksi ja ajetaan uudessa ympäristössä toiminnallisuuden varmistamiseksi. Automatisoitujen testitapausten nauhoittaminen vie aikaa, mutta ne ovat uudelleenkäytettäviä ja ylläpidettyinä palvelevat myös tulevaisuudessa tehtävää sovelluskehitystä ja ylläpitoa. (Teppe 2009.)

Sodifrancen toimittamassa ranskalaisen pankin sovelluskehitysalustan ja –ympäristön migraatiossa testauksen osuus koko projektin kustannuksista oli jopa 45 %. Tämä johtui pääosin siitä, että sovellusten migraatio hoidettiin automatisoidusti, mutta testaus manuaalisesti. Testitapausten automatisaatiolla migraation kustannuksia olisi saatu laskettua entisestään. (Fleurey ym. 2007.)

2.7 Miksi ylläpitoa?

Teknologia kehittyy ja uusia, entistä tehokkaampia ja parempia ohjelmistokehityksen ratkaisuja ja malleja tuodaan markkinoille jatkuvasti. Aina ei kuitenkaan ole tarpeen kehittää kaikkea uudestaan, vaan vanhoja sovelluksia muokataan uusiin tarpeisiin sopivammiksi. IEEE Standardi 1219 määrittelee ohjelmistojen ylläpidon seuraavasti:

> ”Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.”

Suomennettuna ohjelmistojen ylläpito on tuotantoon siirretyn ohjelmistotuotteen muokkaamista virheen korjaamiseksi, suorituskyvyn tai muiden tekijöiden parantamista tai ympäristöön liittyvien muutosten sopeuttamista. Ylläpidon tavoitteena on siis muokata jo tuotantoon siirrettyä ohjelmistoa samalla säilyttäen sen eheys suhteessa muihin sovelluksiin. Ylläpito mielletään usein hitaaksi ja kalliiksi, joskin tarpeelliseksi prosessiksi. Tarve ylläpidolle tulee noin 75 prosentissa tapauksista joko käyttäjien toiveista/vaatimuksista tai muuttuneesta ympäristöstä. Ohjelmistojen ylläpito kuluttaa myös suurimman osan ohjelmiston koko elinkaaren kustannuksista ja toisaalta mikäli ohjelmisto ei pysty vastaamaan kyllin nopeasti muuttuneisiin tarpeisiin tarkoittaa se liiketoimintahyötyjen laskemista ja sitä kautta ohjelmiston arvon alenemista. (Bennett & Rajlich 2000.)

3 Middleware-järjestelmät palvelukeskeisen integraatio-alustan perustana

Middleware-järjestelmällä tarkoitetaan järjestelmää, joka helpottaa sovellusten tai palveluiden suunnittelua, kehitystä, hallinnointia ja integraatiota osana järjestelmäympäristöä. Middleware-järjestelmät tarjoavat mm. ajoalustan joukolle sovelluksia, jossa niitä voidaan hallita ajonaikaisesti keskitetysti ja/tai esimerkiksi rajapinnan alemman tason sovellusten ja järjestelmien käyttöön palveluina tai liiketoimintaprosesseina. Ohjelmistokehitykseen tarkoitetuissa Middleware-tuotteissa on yleensä myös rajapintoja eri ohjelmointikielille (API = Application Programming Interface). Middleware-järjestelmillä saavutetaan ympäristöön myös ei-toiminnallisia ominaisuuksia kuten skaalautuvuutta, hallittavuutta ja tietoturvaa. (Al-Jaroodi & Mohamed 2012.)

3.1 Middleware-järjestelmät SOA-palveluiden tarjoajina

Palvelukeskeisen ohjelmistoarkkitehtuurin (SOA=Service Oriented Architecture) yleistyessä myös Middleware-järjestelmien käyttö on laajentunut. SOAa toteuttavissa ympäristöissä palveluita tarjotaan standardien rajapintojen kautta ja Middleware-järjestelmät toimivat tämän rajapinnan tarjoajina. Middleware-tuotteet ovat usein kolmannen osapuolen ohjelmistoyrityksen tuottamia ratkaisuja palvelujen keskitettyyn hallintaan. Middleware-järjestelmät palvelurajapinnan tarjoajina tukevat SOA-pohjaista ohjelmistoarkkitehtuurimallia ja siten myös palveluiden uudelleenkäytettävyyttä. Palvelukeskeinen arkkitehtuurimalli perustuu lähtökohtaisesti kolmeen toimijaan: palvelun tarjoaja, palvelun välittäjä ja palvelun kuluttaja. Middleware-tuotteet ovat palvelun välittäjiä, jotka tarjoavat rajapinnan sovelluksen käyttöön esim. Web Servicen tai REST-rajapinnan kautta. (Al-Jaroodi & Mohamed 2012.)

3.2 Middleware-järjestelmien ylläpito

Kuten muitakin järjestelmiä, myös Middleware-järjestelmiä tulee hallita ja ylläpitää. Middleware-tuotteet ovat yleensä ohjelmistoyritysten toimittamia ja kehittämiä järjestelmiä, jotka ovat erillisen lisenssimaksuin ostettavan tuen piirissä. Tuen piirissä pysyäkseen tulee Middleware-tuotteita päivittää. Tuotteen toimittanut yritys tarjoaa yleensä

tukea tuotteelle tietystä versiosta ylöspäin ja uusien versioiden ilmestyessä vanhempien versioiden tuki päättyy. Middleware-järjestelmien toimiessa ympäristön keskiössä muiden järjestelmien solmukohtana niihin kohdistuu paljon integraatioita ja siten niiden ylläpito on paljon aikaa ja testausta vaativa prosessi. Middleware-järjestelmiä voidaan ylläpitää ja hallinnoida joko osana sovellusten hallintaprosessia, jolloin ylläpito ja hallinnointi tehdään jonkun sovelluksen tarpeita vastaamaan tai osana ympäristöjen keskitettyä hallintaprosessia. Middleware-järjestelmien versionvaihtoa tai muuta ylläpitoa suunniteltaessa lähtötason kartoitus on tärkeää. Kaikki tuotteen integraatiot muihin järjestelmiin on selvitettävä ja kuvattava sekä huolehdittava, että kaikki rajapinnat tulevat todennetuiksi. (Bennett & Rajlich 2000; Office of Government Commerce 2007, 99)

Middleware-järjestelmien konfiguraatioiden laadukkaassa ylläpidossa versionhallinnalla on yhtä suuri merkitys kuin muidenkin sovellusten ylläpidossa. Jo valittaessa Middleware-tuotetta kannattaa huolehtia siitä, että tuote on mahdollista asentaa ”hiljaisesti”, eli automatisoidusti skripteillä. Skripteillä tehtävien asennusten konfiguraatiot on mahdollista tallentaa konfiguraationhallinta-järjestelmään ja siten ne ovat helposti uudelleenkäytettävissä. Manuaaliasennuksiin verrattuna skripteillä tehtävät automatisoidut asennukset maksavat itsensä nopeasti takaisin työmäärien ja virheiden vähentyessä. Skripteillä automatisoidut asennukset ovat yleensä manuaali-asennuksia virheettömämpiä kirjoitusvirheiden vähentyessä samaa asennusmallipohjaa kerta toisensa jälkeen käytettäessä. Ideaali-tilanteessa tuotteen asennus-binäärit ja konfiguraatio-tiedostot on molemmat talletettu versionhallintaan eri hakemistoihin, joista asennusohjelma tai -skripti ne hakee ja asentaa haluttuun ympäristöön. Mikäli ohjelmisto tallentaa konfiguraatiotiedostonsa tietokantaan, on huolehdittava tietokannan automatisoidusta varmentamisesta. (Humble & Farley 2011, 295-299.)

3.3 Versionhallintajärjestelmä ylläpidon työkaluna

Versionhallinta on osa ympäristön laadukasta ylläpitoa. Versionhallinnalla tarkoitetaan prosessia, jolla projektiin liittyviä artefakteja ja niiden välisiä riippuvuuksia hallitaan. Versionhallinta määrittää miten projektiin kohdistuvia muutoksia hallitaan. Versionhallintajärjestelmä tukee versionhallinnan prosessia. Versionhallintajärjestelmä tuo ympäristön hallintaan laatua ja vakautta automatisoiduilla prosessilla. Versionhallintajärjes-

telmän kautta voidaan palauttaa aikaisempi versio sovelluksesta tai ympäristön konfiguraatioista. Versionhallintajärjestelmän kautta voidaan myös havaita sovellukseen tai ympäristöön kullakin ajanhetkellä kohdistuneet muutokset. (Humble & Farley 2011, 31-33.)

Sovelluksen toiminnan kannalta ympäristön konfiguraatiolla on yhtä suuri merkitys kuin itse sovelluksen konfiguraatiolla. Ympäristöjen manuaalinen hallinta saattaa vaikuttaa helpolta ja nopealta tavalta tehdä muutoksia ympäristön konfiguraatioon, mutta manuaalisten muutosten hallinta ja siirto sellaisenaan uuteen ympäristöön on suurissa ympäristöissä äärimmäisen hankalaa. Ympäristön konfiguraatioiden kustannustehokkaan hallinnoinnin kannalta ensiarvoisen tärkeää on hallita konfiguraatioita helposti ja nopeasti uudelleenluotavana massana. Ympäristön konfiguraatiot pitäisi aina olla kustannustehokkaampaa luoda uudelleen, kuin korjata vanhaa. Paras käytäntö tähän on toimittaa konfiguraatiot täysin automatisoituna prosessina. (Humble & Farley 2011, 49-50.)

4 Case: Digia jatkuvan palvelun toimittajana Asiakkaan integraatioympäristössä

Digia Oyj on suomalainen ohjelmistoalan yritys, jolla on toimintaa maailmanlaajuisesti. Digiassa työskentelee yli 1000 henkilöä seitsemässä eri maassa. Digian liikevoitto ennen veroja vuonna 2012 oli 19,9 miljoonaa euroa. Digia tuottaa IT-palveluita usealle eri liiketoiminnan alalle keskittyen erityisesti finanssiliiketoimintaan, julkiseen sektoriin, kaupan ja palveluiden sekä telekommunikaation alueeseen. Digian tavoitteena on auttaa asiakkaitaan löytämään liiketoimintahyötyjä ymmärtämällä asiakkaan tarpeita. (Digia Oyj 2013a.)

Vuonna 2006 Asiakkaalla otettiin käyttöön Sentera Oyj:n (nykyinen Digia Oyj) toimittama IBM-teknologiaan perustuva integraatiokeskus. Integraatiokeskus toimitettiin kolmessa päävaiheessa: käyttöönottovaihetta edelsivät evaluointi- ja arkkitehtuurimäärittelyvaiheet, joissa kaikissa Sentera oli mukana. (Virtanen, T. 20.2.2013.)

Integraatiokeskus on kokonaisvaltainen integraatioratkaisu lähes ajantasaiseen tiedonvälitykseen Asiakkaan sovellusten ja kumppaneiden välillä. Ratkaisu toi läpinäkyvyyttä ja hallittavuutta sanomaliikenteen käsittelyyn, sekä antoi arvokasta tietoa johtamisen tueksi. (Liite 2.)

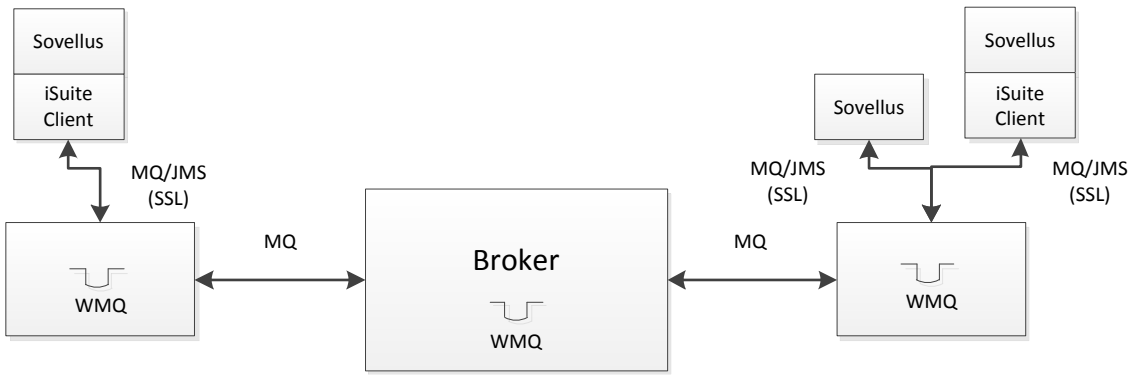
4.1 Asiakkaan tiedonvälityspalvelu

Asiakkaan tiedonvälityspalvelu huolehtii sekä Asiakkaan sovellusten että Asiakkaan ja sen kumppaneiden välisestä aineistoliikenteestä. Tiedonvälityspalvelu hoitaa keskitetysti ja uudelleenkäytettävyyden periaatteita noudattaen sovellusten ja järjestelmien väliset integraatiot. Tiedonvälityspalvelu tukee sekä vanhempia eräajopohjaisia tiedostosiirtoja, että uudempaa sanomapohjaista liikennettä. Tiedonvälityspalvelu tukee myös erilaisia synkronisia ja asynkronisia (lähes ajantasaisia) protokollia, sekä viestiformaatteja ja -standardeja. Palvelu tarjoaa työkalut aineistoliikenteen käsittelyyn, seurantaan, operointiin ja arkistointiin. Aineistosiirtojen käsittelyvaiheesta jää lokimerkinnät palvelun seurantakantaan ja virheellisistä tai puutteellisista siirroista generoituu hälytys operointiin. Tiedonvälityspalvelun kautta kulkevaa aineistoa säilytetään oletuksena kahden viikon

ajan arkistossa. Tiedonvälityspalvelun kautta kulkevaa aineistoa voidaan tarvittaessa rikastaa tai muuntaa palvelun toimesta vastaanottajan ymmärtämään muotoon. Palvelu tuo myös tietoturvaa salaamalla ja purkamalla aineiston salauksen. Asiakkaan tiedonvälityspalveluun on päivittäin yhteydessä pitkälti toista tuhatta sovellusta lähettäen tiedonvälityspalvelun kautta miljoonia sanomia. (Asiakkaan IT-palvelut 2012a.)

4.2 Middleware-tuotteet tiedonvälityspalvelua tukemassa

Asiakkaan tiedonvälityspalvelu on rakennettu keskenään integroiduista Digian iSuite-tuotteista, sekä yhtenäistä arkkitehtuuria toteuttavista IBM:n WebSphere-perheen Middleware-tuotteista. Tiedonvälityspalvelun sanomanvälitys perustuu WebSphere Message Queue-järjestelmään (myöh. WMQ), joka on asynkroniseen sanomanvälitykseen erikoistunut sanomajonoihin perustuva tuote. WebSphere Message Broker (myöh. Broker) on IBM:n sanomien ja tiedostojen muunnokseen ja reititykseen erikoistunut ESB-ohjelmisto (ESB = Enterprise Service Bus), joka toimii tiedonvälityspalvelun keskiössä. Sovellukset lähettävät keskitetylle WMQ:n jonomanagerille sanomia, jotka WMQ ohjaa Brokerin käsittelyyn. Sanoman tunnisteen tai sisällön perusteella Broker poimii sanomajonoista sanomat käsittelyynsä, muuntaa, rikastaa ja/tai reitittää sanomat edelleen lopulliselle vastaanottajalle. Kuviossa 1 havainnollistetaan Asiakkaan integraatiokeskuksen komponenttien keskinäisiä suhteita ja Broker-järjestelmän sijaintia integraatioiden keskipisteenä. Broker käsittelee sanomia tietovirta-toteutuksissaan, jotka ovat graafisella työkalulla toteutettuja sovelluksia, joihin on myös Java- tai ESQL-rajapinnat. (Digia Oyj 2013b.)



Kuvio 1: Middleware-tuotteet Asiakkaan integraatiokeskuksen tiedonvälityspalvelun ytimessä

Asiakkaan integraatiokeskuksen Tiedonvälityspalvelun komponenttien koodeja ja konfiguraatioita säilytetään IBM:n toimittamassa ClearCase-versionhallintajärjestelmässä. Versionhallintajärjestelmästä voidaan eriyttää haarat kuhunkin kehitysympäristöön ja ohjelmallisesti yhdistää tehdyt muutokset toisiinsa.

4.3 Jatkuva palvelu Asiakkaan tiedonvälityspalvelussa

Digia toimittaa Asiakkaalle integraatiopalvelua jatkuvan palvelun parantamisen menetelmien mukaisesti. Jatkuvan palvelun parantamisen prosessissa määritetään visio, kartoitetaan nykytilanne, asetetaan tavoitteet, määritetään etenemistapa, sekä kehitystyön tai projektin päätyttyä mitataan saavutetut tulokset ja kehitetään palvelua edelleen tulosten pohjalta. (Office of Government Commerce 2007, 35.)

Yhteistyötä Digian ja Asiakkaan välillä on tehty useiden vuosien ajan ja sitä pyritään edelleen jatkuvasti parantamaan vähentämällä virheitä, tehostamalla prosesseja, sekä etsimällä uusia ratkaisuja liiketoimintahyötyjen saavuttamiseen. Jatkuvaa palvelua pyritään kehittämään mm. uudelleenkäytettävyyden periaatteiden mukaisesti määrittämällä ja kuvaamalla keskeisiä integraatioiden hallintaan liittyviä prosesseja.

4.3.1 Jatkuvan palvelun käytännöt integraatiokeskuksen ylläpidossa

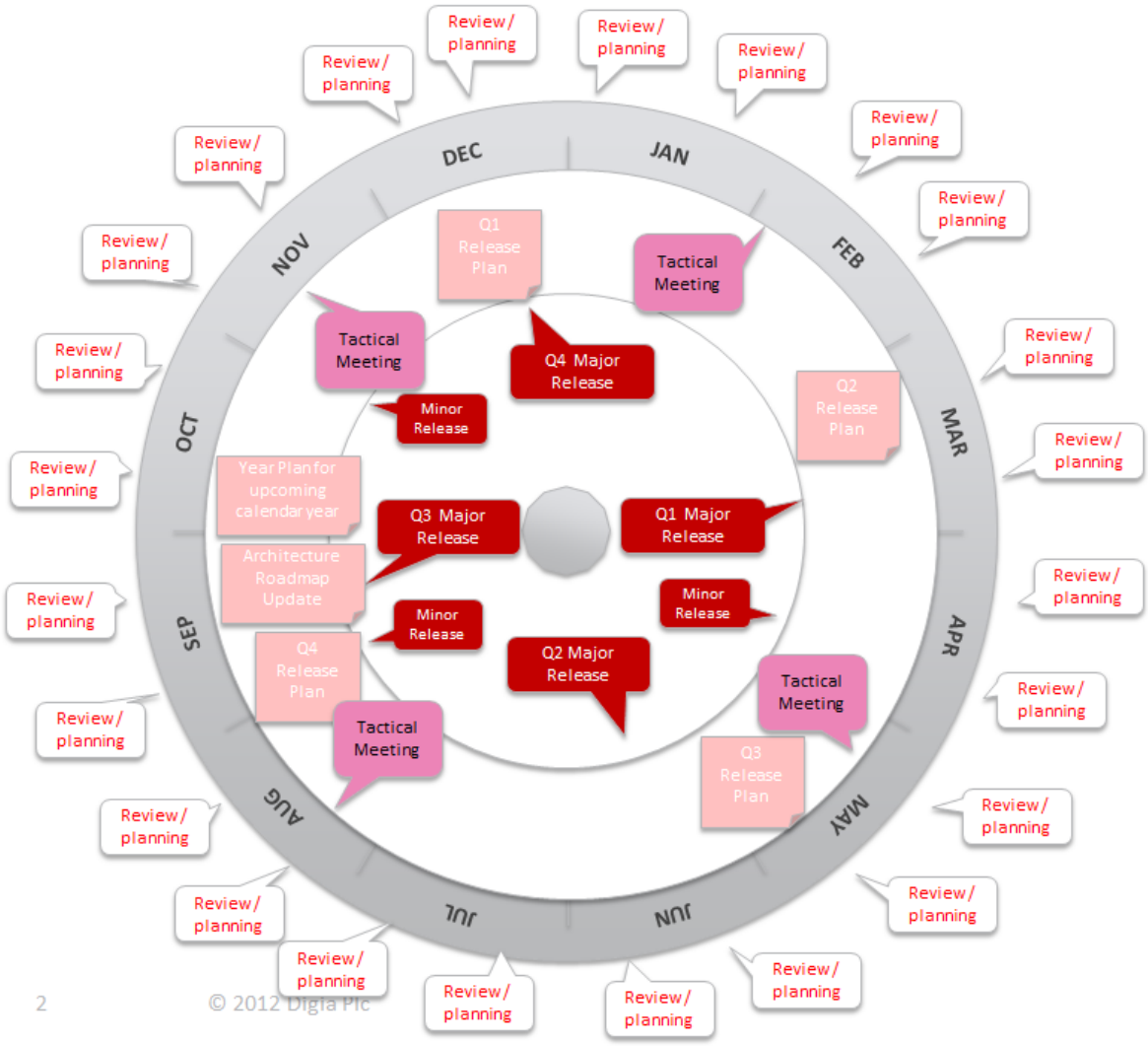
Palvelun jatkuva parantaminen on osa palvelun hallintaa. Palvelun jatkuvalla parantamisella pyritään vastaamaan liiketoiminnan muuttuviin vaatimuksiin tuottamalla asiakkaalle lisäarvoa IT-palveluiden prosesseja tehostamalla. Jatkuvan parantamisen mallin mukaisesti palvelua seurataan ja mitataan jatkuvasti. Seurannan perusteella palvelua suunnitellaan ja kehitetään entisestään. Olemassa olevan palvelun kehittäminen on jatkuvaa. Palvelua ohjaavat erilaiset suunnittelu- ja seurantapalaverit, joissa suunnitellaan tulevia hankkeita ja projekteja, sekä seurataan niiden etenemistä jatkuvan palvelun prosessien mukaisesti. (Spalding 2007, 14.)

Digian toimittamaa palvelua Asiakkaan integraatiokeskuksen ylläpidossa ohjataan ITIL:in jatkuvan palvelun parantamisen prosesseihin perustuvilla käytännöillä. Asiakkaan integraatiokeskuksen tiedonvälityspalveluun kohdistuvia kehitys- ja ylläpitotöitä

sekä -hankkeita hallitaan Digian toimesta määrämuotoista projektityömallia noudattaen. Tärkeimpiä ympäristöön kohdistuvia hankkeita ja projekteja seurataan myös asiakkaan projektitoimistossa.

Työskentelymenetelmänä Asiakkaan integraatiokeskukseen kohdistuvassa työssä Digia käyttää Digia Project- ja CPM (Core Process Model)-malleja, jotka perustuvat ohjelmistokehityksen ketterien menetelmien osalta Scrum- ja XP-menetelmiin. Työtä tehdään 3-8 hengen tiimeissä, kahden viikon sprinteissä. Kehitystyö suunnitellaan luomalla automatisoidut testitapaukset käyttötapausten pohjalta. Kaikki sovelluskoodit ja konfiguraatiot talletetaan versionhallintajärjestelmään.

Integraatiokeskuksen jatkuvaa palvelua ohjaa vuosisuunnittelumallin perusteella kuvattu vuosikellon mukainen aikataulu (ks. Kuvio 2), jossa on kuvattuna alustavat ajankohdat tuotantoon siirroille, sekä suunnittelu- ja tarkistuspisteille. Tärkeimmät jatkuvan palvelun ohjauksen käytännöt ovat erilaiset asiakkaan kanssa yhdessä pidettävät, suunnittelua ohjaavat palaverit. (Digia Oyj 2013c.)



Kuvio 2. Vuosikello ohjaa palvelun vuosisuunnittelua (Digia Oyj 2013c).

Vuoden loppupuoliskolla vuosisuunnittelu-työpajoissa valmistellaan seuraavaa kalenterivuotta koskeva kehityssuunnitelma, jossa kuvataan seuraavan vuoden tärkeimmät kehityshankkeet. Kehityssuunnitelmaa päivitetään hankkeiden etenemisen mukaisesti. Vuosisuunnitteluun osallistuvat asiakkaan edustajat, kehitystiimit, palvelupäällikkö ja palvelusalkun haltija. Vuosisuunnitelma hyväksytään kalenterivuoden alussa pidettävässä katselmointi-palaverissa asiakkaan palvelualueen omistajan toimesta. (Digia Oyj 2013c.)

Kehityksen aikataulua ohjaavat neljännesvuosittaiset ”Major Releaset”, jotka ovat suunniteltuja sovelluskokonaisuuksien keskitettyjä käyttöönottoja. Pienempiä muutoksia voidaan viedä tuotantoon ns. ”Minor releaseina” varsinaisten Releasejen välissä. Releasejen jälkeen pidettävissä taktisissa palavereissa tarkastellaan kehityssuunnitelman

toteutumista. Taktisiin palavereihin asiakkaan edustuksen lisäksi osallistuu Digialta mm. asiakaskohtainen myyntipäällikkö, palvelusalkun haltija, palvelupäällikkö ja pääarkkitehti. Taktisten palavereiden tavoitteena on kartoittaa palvelutilanne asiakkaan näkökulmasta. Jatkuvan palvelun parantamisen menetelmien mukaisesti taktisissa palavereissa käydään läpi asetettuja mittareita tuotannon häiriöiden ja palvelupyyntöjen sekä hankkeiden kustannusten toteutumisen osalta. Palavereissa kartoitetaan myös hankkeiden tilanne suhteessa vuosisuunnitelmaan, käydään läpi laskutusta sekä ennakoidaan mahdollisia riskejä. (Holmala, R. 22.2.2013.)

Ylläpitoon liittyviä töitä seurataan ja läpikäydään yhdessä asiakkaan kanssa Sprint Review and Planning -kokouksissa kahden viikon välein pidettävien viikkopalaverien yhteydessä. Viikkopalavereissa asiakkaan edustajalla on mahdollisuus priorisoida työn alle otettavia tehtäviä ja nostaa uusia kehitettäviä töitä työlistalle. Sprint Review and Planning-kokouksiin osallistuvat asiakkaan ja palvelupäällikön lisäksi kehitystyötä tekevät tiimit. (Digia Oyj 2013c.)

4.3.2 Integraatioiden hallintamalli

Digian ylläpitämän Asiakkaan integraatiokeskuksen kautta hoidetaan tiedonvälityspalvelun integraatiot muihin järjestelmiin ja yhteistyökumppaneille. Integraatioiden hallintaa ja uusien integraatioiden toimitusta on tehostettu määrittelemällä ja kuvaamalla integraatioihin liittyvien prosessien parhaat käytännöt integraatioiden hallintamallilla. Hallintamallissa kuvataan integraatioiden suunnitteluun ja toteutukseen liittyvät prosessit ja niiden sisältämät työvaiheet. Hallintamalli on joustava ja sovellettavissa eri järjestelmähankkeiden eri tilanteisiin ja näkyy asiakasorganisaatioille Asiakkaan Tietohallinnon tiedonvälityspalveluiden tarjoamina IT-palveluina. Integraatioiden hallintamallissa määritetään ne prosessit, jotka varmistavat mm. integraatioiden tarvemäärittelyn, suunnittelun ja toteutuksen yhtenäiset ja tehokkaat käytännöt. Jokaiselle prosessille on kuvattu ja vastuutettu niihin liittyvät roolit. Hallintamallissa on kuvattu myös integraatioihin liittyvien keskeisten dokumenttien sisältö. Yhtenäinen integraatioiden hallintamalli tuo prosessiin laatua, hallittavuutta ja ennakoitavuutta. (Asiakkaan IT-palvelut 2012b.)

4.4 Jatkuvan palvelun parantamisen mallin mukainen projektisuunnittelu integraatiokeskuksen ylläpidossa

Digia pyrkii toimittamaan Asiakkaan integraatiokeskuksen ylläpito- ja kehityspalvelua jatkuvasti palvelua parantaen. Integraatiokeskus koostuu erilaisista sanomia reitittävistä ja käsittelevistä IBM:n WebSphere-perheen Middleware-tuotteista, joita ylläpidetään tuetuissa versioissa. Versiotasonnostoja tehdään noin 3-5 vuoden välein ja ne toteutetaan kohtalaisen laajoina projekteina.

Asiakkaan integraatiokeskuksen Middleware-tuotteiden, eli lähinnä Brokerin ja WMQ:n, versionvaihtoprojektit ovat Asiakkaan integraatiokeskuksessa laajoja projekteja, koska erilaisia Broker-sovelluskomponentteja on muutamia satoja ja uusia toteutuksia kehitetään aktiivisesti. Yhteensä integraatioita eri sovelluksiin on toista tuhatta. Migraatiot hoituvat Broker-tuotteella automatisoidusti, mutta toteutuksiin ja ympäristöön joudutaan kuitenkin tekemään jonkin verran manuaalisesti muutoksia. Testitapausten ajaminen uusia koodeja vasten on työläs prosessi, sillä tietokannan ja ympäristön tilan tulee olla kullekin testitapauksen kannalta oikealla mallilla.

Asiakkaan integraatiokeskuksen kehityksessä ja ylläpidossa pyritään Digian toimesta mahdollisimman ketterään ja laadukkaaseen toimintamalliin. Keväällä 2013 tehtävä versionvaihto-projekti toteutetaan normaalia Digian projektimallia noudattaen.

4.5 WebSphere Message Broker-tuotteen versionvaihto-projekti Asiakkaan integraatioympäristössä

Syksyllä 2012 aloitettiin tiedonvälityspalvelun perustana olevan WebSphere Message Broker-sovelluksen migraatio-projekti (Brokerv8). Projektin tavoitteena on päivittää tuotannossa käytössä oleva Brokerin 6.1-versio uusimpaan 8-versioon kevään 2013 aikana. Samassa yhteydessä päivitetään Brokerin käyttämät WMQ-jonomanagerit uusimpaan 7.5-versioon, sekä vaihdetaan palvelinlaitteisto uudempaan ja nostetaan AIX-käyttöjärjestelmätasoa. (Digia Oyj 2012.)

Edellinen vastaava tiedonvälityspalvelun Middleware-järjestelmien versionnosto-projekti toteutettiin kolmea vuotta aikaisemmin syksyllä 2009, jolloin onnistuneesti päi-

vitettiin koko integraatiokeskuksen palvelimet, käyttöjärjestelmäversiot, sekä Brokerin ja WMQ:n versiot. Edellisen versionvaihdon jälkeen integraatiokeskuksen välittämän sanomaliikenteen volyymi ja käsittelyä toteuttavien komponenttien määrä ovat kasvaneet huomattavasti sovellusten siirryttyä käyttämään asynkronista sanomaliikennettä perinteisen FTP–liikenteen sijaan. (Digia Oyj 2012; Virtanen, T. 22.2.2013.)

Tarve WebSphere Message Brokerin version vaihdolle lähti edellisen 6.1-version tuen loppumisen päivämäärän julkaisusta. IBM:n tuki 6.1-versiolle loppuu syksyllä 2013. Jotta Asiakkaan integraatiokeskus pysyisi IBM:n tuen piirissä, on Brokerin versio päivitettävä. Brokerin nojatessa vahvasti WebSphere Message Queue-tuotteeseen sanomanvälityksen osalta, samassa yhteydessä päivitetään WMQ:sta uusin Major Release-versio 7.5. Brokerv8 versionnosto-projekti toteutetaan yhteistyössä ympäristöjen ylläpitoa ja tuotantopalvelua hoitavan kolmannen osapuolen kanssa. (Digia Oyj 2012.)

Projekti käynnistettiin suunnittelulla syksyllä 2012 ja kehitysympäristön asennukset aloitettiin tammikuussa 2013. Ympäristön WMQ-jonomanagereista päivitetään Brokerv8 migraatio-projektin yhteydessä vain Brokerin käyttämät jonomanagerit. Tiedonvälityspalvelun muut WMQ-jonomanagerit ja niiden palvelimet säilytetään nykyisessä ympäristössä nykyisissä versioissaan ja päivitetään uuteen ympäristöön Brokerv8 migraatio-projektin jälkeen omana projektinaan. (Digia Oyj 2012.)

Brokerv8-migraatio suunniteltiin edellisen Broker-versionnostoprojektin kokemusten ja Digian projektinhallintaprosessien pohjalta. Brokerv8-versionnostoprojektin tavoitteena on siirtää kehitys ja tuotanto uuteen Brokerv8-ympäristöön mahdollisimman näkymättömästi ja mahdollisimman lyhyessä ajassa aiheuttaen mahdollisimman vähän rinnakkaisen kehitystyön tarvetta.

Brokerv8 migraatio-projektin suunnittelussa tärkeintä on kehityksen mahdollistaminen Broker-toteutuksille myös migraatio-vaiheessa. Sovelluksia on pystyttävä kehittämään ja tuotantoon pitää pystyä viemään mahdollisia korjauspaketteja myös migraation aikana. Rinnakkainen kehitys kahden eri ympäristön sovelluskoodien välillä on mahdollista toteuttaa versionhallintajärjestelmän avulla. Versionhallintajärjestelmän avulla on mahdollista eriyttää tietyn hetkistä tuotantoa vastaavat koodit migroitavista koodeista. Toi-

saalta rinnakkaisen kehityksen tarve kahteen eri ympäristöön halutaan pitää minimissään.

Brokerv8-migraation pohjaksi otettiin vuoden 2013 ensimmäisessä Releasessa tuotantoon siirretyt koodit. Brokerv8-ympäristön tuotantoon siirto hoidetaan vuoden 2013 toisessa Major Releasessa toukokuussa. Brokerv8-sovellusten toiminta ja koodit vastaavat ensimmäisessä Releasessa tuotantoon siirrettyjä koodeja. Tällä halutaan varautua mahdolliseen paluuseen tulevassa tuotantoon siirrossa; mikäli tuotantoon siirto uuteen ympäristöön jostain syystä epäonnistuu, on sanomaliikenne mahdollista ohjata takaisin vanhaan ympäristöön ja säilyttää sama toiminnallisuus. Uusiin Brokerv8-koodeihin ei siis oteta mukaan mitään sellaista toiminnallisuutta, jota siihen asti ajossa olevassa tuotantoympäristössä ei ole. Mikäli v6.1 tuotantoympäristöön joudutaan näiden Releasejen väliaikana viemään jokin uusi toiminnallisuus, sama muutos tulee myös tehdä v8-migroituihin koodeihin. Releasejen väliajalla tehdyistä muutoksista koodeihin pidetään kirjaa. (Digia Oyj 2012.)

Hallittavuuden maksimoimiseksi kaikki ympäristön konfiguroinnit ja asennukset tehtiin uudestaan käytettävillä skripteillä. Skriptit tallennettiin versionhallintaan, jossa ne toimivat samalla myös ympäristön dokumentaationa. Skriptit kehiteltiin lokaalissa työasemaympäristössä ja viimeisteltiin kehitysympäristön asennusten yhteydessä. Samoilla skripteillä hoidetaan testi- ja tuotantoympäristön asennukset ja konfiguraatiot. (Digia Oyj 2012.)

Broker-toteutusten koodien migraatio hoituu automaattisesti Broker-tuotteen toimesta asentamalla vanhat v6.1-koodit uuteen v8-ympäristöön. Broker muokkaa itse sovelluskoodit uuteen muotoon. Muutamissa poikkeustapauksissa joudutaan manuaalisesti hoitamaan jokin osa migraatiosta. Migraation ollessa automaattinen, migraatioprosessin tärkein tehtävä on migroidun sovelluskokonaisuuden todennus uudessa ympäristössä. Digia käyttää kaikessa Asiakkaan integraatiokeskukseen kohdistuvassa kehitystyössään ennalta suunniteltuja ja toteutettuja testitapauksia. Testitapaukset ovat uudelleenkäytettäviä versionhallintajärjestelmässä säilytettäviä skriptejä. Kaikki testitapaukset ajetaan kehitysympäristössä öisin, jolla varmistetaan, etteivät muutokset ympäristössä tai muissa toteutuksissa vaikuta ei-toivotulla tavalla muihin toteutuksiin. Testitapaukset käyn-

nistävät tapahtumasarjan, jossa Brokerin käsittelyyn ohjataan kunkin testitapauksen tarvitsemaa aineistoa. Mikäli testitapaus menee onnistuneesti läpi, katsotaan testitapauksen testaamaan koodin toimivan. (Digia Oyj 2012.)

Kaikkien testitapaus-ajojen onnistuttua suoritetaan vielä muutamia suorituskyky- sekä vikasietotestiajoja uutta ympäristöä vasten ja verrataan niitä vanhan ympäristön vastaaviin tuloksiin. Suorituskykytesteillä varmistetaan, että uusi ympäristö on vähintään yhtä suorituskykyinen kuin vanha ympäristö. Vikasietotesteillä varmistetaan, ettei tarpeellisia sanomia häviä ja että ympäristö palautuu vikatilanteesta toimintakuntoon. (Digia Oyj 2012.)

Brokerv8-projektin organisaatio on pyritty pitämään pienenä ja tehokkaana. Digiasta projektin toteutukseen osallistuu kaksi asiantuntijaa sekä projektipäällikkö. Asiantuntijat keskittyvät ympäristön asennus- ja konfiguraatioskriptien luomiseen, migroitavien sovellusten asentamiseen uuteen ympäristöön, sekä testien ajamiseen ja seurantaan. Tuotantopalveluja hoitavalta kolmannelta osapuolelta projektiin osallistuu AIX-asiantuntija, verkkoasiantuntija, sekä projektipäällikkö. (Digia Oyj 2012.)

Brokerv8-migraatioprojektia ohjataan päivittäisillä scrum-palavereilla, viikoittaisilla tuotantopalveluita hoitavan yrityksen edustajien kanssa pidettävillä koko projektiryhmän palavereilla, sekä kerran kuukaudessa järjestettävillä ohjausryhmän kokouksilla. Ohjausryhmän kokouksiin osallistuu asiakkaan (Asiakkaan IT Infra) edustajien lisäksi Digialta palvelusalkun haltija, palvelupäällikkö ja projektipäällikkö, sekä tuotantopalveluja hoitavan yrityksen projektipäällikkö. Lisäksi projektin tilannetta käydään asiakkaan kanssa läpi joka toinen viikko järjestettävässä ”Sprint review & planning”-kokouksessa. Viikkopalavereita ja ohjausryhmän kokouksia varten Digian projektipäällikkö valmistelee materiaalin, jossa käydään läpi tärkeimmät mittarit projektin edistymisestä ja kustannuksista. (Digia Oyj 2012.)

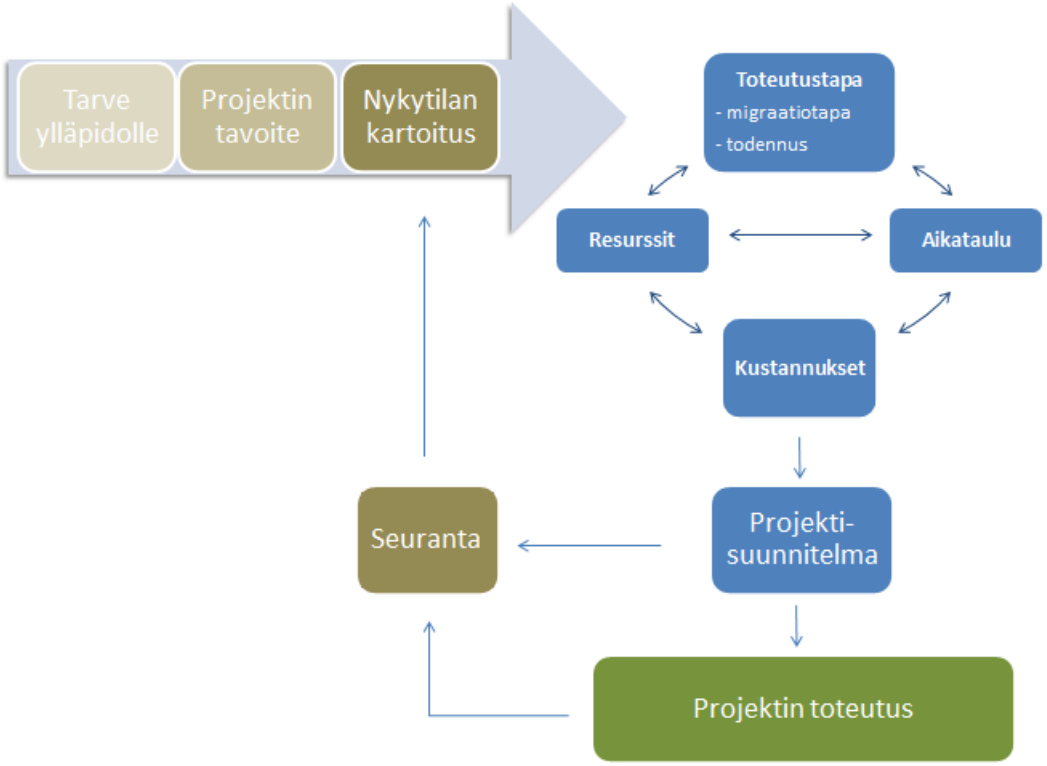
5 Middleware-järjestelmän versionvaihdon suunnittelu palvelua jatkuvasti parantaen

Tässä tutkimuksessa tavoitteena on löytää malli Middleware-järjestelmien ylläpitoprojektien suunnittelulle palvelun jatkuvan parantamisen menetelmien mukaan. Tavoitetilassa Middleware-järjestelmän ylläpito voitaisiin hoitaa mahdollisimman näkymättömästi ja pienin kustannuksin samalla minimoiden rinnakkaisiin ympäristöihin tehtävä kaksinkertainen sovelluskoodien ylläpito. Yleisesti sovellusten migraatiosta ja migraatiomenetelmistä löytyy joitakin tutkimuksia, mutta Middleware-järjestelmien ylläpidosta ja niiden migraatio-menetelmistä tutkimustietoa on vain vähän.

5.1 Ylläpitoprojektin suunnittelumalli jatkuvan palvelun parantamisen prosessin mukaan

Ylläpitoprojektin suunnitteluprosessi jatkuvan palvelun parantamisen menetelmien mukaisesti eroaa hieman perinteisestä kehitysprojektin suunnittelusta. Koska kyseessä on ylläpidettävä ympäristö ja toiminnallisuus, seurantaa toteutetaan jatkuvasti ja tarve ylläpidolle laukaisee suunnittelun. Tarve ylläpidolle tulee usein välillisesti ulkopuolelta, esim. ohjelmiston tai käyttöjärjestelmän tuki päättyy, tuotteelta vaaditaan uusia ominaisuuksia tai koko ajoympäristö halutaan vaihtaa toiseen (esim. COBOL -> J2EE). Tarpeen ilmettyä kartoitetaan ja kuvataan nykytila, määritetään tavoite ja suunnitellaan miten tavoitteeseen päästään, missä ajassa ja keiden toimesta. Toteutustapaa suunniteltaessa tulee samalla suunnitella myös laadunvarmistusprosessi, miten ylläpitoprojektin tuotokset todennetaan, sekä hahmotella tuotantoon siirtotapaa. Projektisuunnitelman päivitys on koko projektin ajan tapahtuva iteratiivinen prosessi, jossa projektin ja palvelun seurannan perusteella tehdään projektisuunnitelmaan muutoksia määrämuotoisen muutoshallinnan prosessin kautta.

Kuviossa 3 olevassa mallissa on havainnollistettu ylläpitoprojektin suunnitteluprosessia perustuen jatkuvan palvelun parantamisen menetelmään.



Kuvio 3. Ylläpitoprojektin suunnittelumalli

5.2 Middleware-järjestelmän versionvaihto-projektin suunnittelu

Middleware-järjestelmät ovat keskeinen elementti suurten ympäristöjen SOA-mallisissa arkkitehtuuriratkaisuissa. Middleware-järjestelmien versionvaihtoprojektit ovat laajoja, montaa sovellusta ja liittymää koskettavia projekteja, joiden suunnittelussa pätee edellä kuvattu ylläpito-projektin suunnittelumalli. Suunnittelumallia on avattu Middleware-järjestelmän versionvaihtoprojektin suunnittelun kannalta kappaleissa 5.2.1–5.2.5.

5.2.1 Projektin tavoitteen asettaminen

Projektin tavoite asetetaan tarpeen perusteella. Middleware-järjestelmien versionvaihtoprojekteissa keskeinen tarve on päivittää tuotteen versio tuoreempaan (esim. päivittää WebSphere Message Broker versiosta 6.1 versioon v8.0). Tämän päivitystavoitteen lisäksi tulee projektille asettaa selkeitä ja mahdollisuuksien mukaan mitattavia tavoitteita (aikataulu- ja kustannustavoitteet, ympäristön suorituskyky jne). Tavoitteeksi ei riitä maininta ”Ympäristön tai sovelluksen tulee toimia kuten ennenkin”. Versionvaihtoprojektien tavoitteiden asettamisessa kannattaa pyrkiä selkeisiin kokonaisuuksiin ja raja-

ta ulos versionvaihtoon suoraan liittymättömät työt, kuten uusien toiminnallisuuksien lisääminen palveluihin.

Middleware-tuotteiden versionvaihto-projekteissa ongelmia aiheutuu usein prioriteetin puutteesta. Tällaiset ”pellin alla” tapahtuvat teknisten versioiden vaihto-projektit eivät tuo liiketoiminnalle uusia näkyviä toiminnallisuuksia ja siksi ne koetaan usein jopa turhiksi, mikäli vanha versio toimii hyvin. Prioriteetin puute versionvaihtoprojektille hankaloittaa resurssien saamista projektin käyttöön, jolloin projekti pitkittyy. Projektin aikataulu venyy myös, mikäli versionvaihto-projektin vaatiman koodien migraation yhteydessä halutaan ottaa käyttöön uusia toiminnallisuuksia. Projektinhallinnan keinoin, tavoitteiden selkeällä asettamisella, realistisella aikataulu-, resurssi- ja kustannussuunnittelulla sekä seurantapisteiden asettamisella ja raportoinnilla projektille on helpompi saada resursseja.

5.2.2 Nykytilan kartoittaminen

Nykytilan kartoittaminen kertoo lähtötason. Nykytilan kartoituksessa ja kuvauksessa on hyvä listata erilaisten ympäristön komponenttien versiotietojen lisäksi selvästi ja riittävällä tarkkuudella myös ko. ylläpidettävän tuotteen tai ympäristön integraatiot muihin järjestelmiin. Integraatioiden listauksen perusteella voidaan myöhemmin tarkistaa testitapausten kattavuus. Suunnitelmaan on hyvä liittää myös ympäristökuvaukset nykyisestä ja tulevasta ympäristöstä.

5.2.3 Toteutustavan, resurssien ja aikataulun suunnittelu

Kun nykytila on kuvattu, suunnitellaan versionvaihtoprojektin toteutustapa. Toteutustavan suunnittelussa kartoitetaan erilaisia toteutustapoja (esim. manuaalinen tai skripteillä automatisoitu migraatio) sekä suunnitellaan tuotosten todennustapa. Jokaiselle toteutusvaihtoehdolle laaditaan tehtäväluettelo, jonka perusteella arvioidaan alustava aikataulu ja toteutustavan vaatimat resurssit sekä lasketaan alustava kustannuslaskelma.

Toteutustapa on siis hyvin kiinteässä suhteessa resursseihin ja aikatauluun. Kartoitettaessa erilaisia toteutustapoja punnitaan samalla eri vaihtoehtojen hyödyt ja haitat, sekä suunnitellaan kunkin toteutustavan tarvitsemat roolit ja resurssit. Resurssien määrä ja

osaaminen on suhteessa aikatauluun. Mitä enemmän ja/tai osaavampia tekijöitä projektilla on käytössä, sitä vähemmän projektiin kuluu aikaa. Toisaalta resurssien määrä ja laatu sekä aikataulun pituus vaikuttavat suoraan kustannuksiin. Projekti on mahdollista suorittaa lyhyemmässä ajassa useammilla resursseilla, jolloin kustannukset nousevat korkeiksi ja resurssien käyttö on poissa muusta kehityksestä. Projektin toteutustapaa suunniteltaessa tulee löytää optimi-arvot resurssien ja ajan käytölle suhteessa kustannuksiin.

Automatisoitu ja ohjelmallisesti toteutettu migraatio on suurissa sovelluskokonaisuuksien migraatioissa tehokkain ja virheettömin tapa, sillä migraation vaatimat koodimuutokset ovat usein ennalta määritettyjä ja samanlaisia kaikille toteutuksille. Ohjelmallisesti toteutetun migraation etuna on myös se, ettei se ole henkilöriippuvaista. Toisaalta automaattisen migraatiotyökalun kehittämisestä aiheutuu kustannuksia, jolloin pienissä ympäristöissä tai sovelluskokonaisuuksissa ei välttämättä ole kustannustehokasta kehittää migraatiolle omaa työkalua.

Middleware-järjestelmän päällä ajossa olevien sovelluskoodien migraatio kannattaa suunnitella ja aikatauluttaa mahdollisimman lyhyelle ajanjaksolle, jotta minimoidaan kahden eri koodiversion rinnakkaisen kehittämisen vaatima työ. Ennen varsinaista migraatio-prosessia kannattaa suorittaa pilotti-migraatio. Pilotoitavaksi sovellukseksi kannattaa valita tarpeeksi kattava sovellus, migroida se uuteen ympäristöön ja kirjata ylös mahdolliset virheet ja toimenpiteet. Sovellusten migraatioaikataulun suunnittelussa tulee priorisoida yleiskäyttöisten, sekä aktiivisesti kehitettävien palveluiden migraatiot uudelle alustalle.

Versionvaihtoprojekteissa voidaan varusohjelmistot päivittää joko migroimalla vanha versio suoraan uuteen tai perustamalla täysin uusi rinnakkainen asennusympäristö uusilla ohjelmistoversioilla. Migroimalla vanha versio suoraan uuteen versioon tulee kaikki Middleware-järjestelmässä ajossa olevat sovellukset ja liittymät ottaa kerralla käyttöön. Uuden ympäristön perustamisen etuna on se, että asennukset ja mahdollinen pilottitestaus voidaan suorittaa rauhassa etukäteen muuta kehitystyötä häiritsemättä. Uutta ympäristöä perustettaessa on samalla tilaisuus päivittää käyttöjärjestelmätasoa. Erillisen rinnakkaisen ympäristön pystyttämisen etuna on myös se, että sovelluksia voidaan tes-

tata ja ottaa tuotantokäyttöön erillisinä kokonaisuuksina ja virhetilanteessa voidaan liikenne ohjata takaisin vanhaan ympäristöön. Rinnakkaista ympäristöä perustettaessa tulee huomioida kahden erillisen ympäristön vaatimat lisenssit. Rinnakkaisen ympäristön pystytys kannattaa tehdä silloin, kun versionvaihto vaatii sovelluksiin koodimuutoksia. Mikäli kyseessä on pienempi versionnosto, joka ei vaadi muutoksia Middleware-järjestelmää käyttäviin muihin sovelluksiin, versionnoston voi hoitaa migroimalla ohjelmiston.

Middleware-järjestelmän migraatio-projektin suunnittelussa myös todennustavan suunnittelu on osa toteutustavan suunnittelua. Versionvaihtoprojekteissa todennus on usein projektin aikaa vievin vaihe. Todennuksen tulee olla kattava ja kaikki integraatiot muihin järjestelmiin tulee todentaa. Todennusta helpottaa, mikäli sovellusten toiminnallisuutta uudessa ympäristössä voidaan verrata alkuperäiseen, eli koodeihin ei ole tehty muita kuin ympäristön ylläpidon vaatimia muutoksia. Mikäli jotakin toiminnallisuutta ei ollut alkuperäisessä koodi-versiossa vanhassa ympäristössä, ei sitä tule olla myöskään uudessa. Vertaaminen vanhaan ympäristöön tarkoittaa myös sitä, että ylläpitoprojektin aikana tulee uuden ja vanhan ympäristön olla rinnakkain ajossa tai että testitapaukset ovat niin tarkasti dokumentoituja, että toiminnallisuus selviää niistä.

Automatisoidut regressiotesti-skriptit ovat ympäristöjen ylläpito-projekteissa erittäin hyödyllisiä. Regressiotestiskriptejä ajamalla voidaan varmistua, että sovellusten toiminnallisuus on säilynyt ennallaan ympäristön ylläpidosta ja/tai migraatiosta huolimatta. Kuormitustesteillä varmistetaan uuden ympäristön riittävä suorituskyky. Mikäli tarkkoja vaateita uuden ympäristön suorituskyvylle ei ole, kannattaa kuormitustestiajot ajaa vanhaa ja uutta ympäristöä vasten ja verrata, ettei uuden ympäristön suorituskyky ole ainakaan vanhaa ympäristöä heikompi. Jos ympäristöllä on vikasietovaatimuksia tulee suorittaa vaadittavat vikasietotestit ja varmistua ympäristön palautumisesta virhetilanteessa.

Tuotantoon siirtoa on hyvä alustavasti suunnitella etukäteen projektin toteutustapaa ja aikataulua suunniteltaessa. Eri sovelluskokonaisuudet kannattaa mahdollisuuksien mukaan siirtää tuotantoon omissa erissään, jolloin tuotantoon siirrossa mahdollisesti tapahtuvien virheiden ja toimimattomuuksien aiheuttaja on helpommin rajattavissa.

5.2.4 Projektisuunnitelma Middleware-järjestelmän versionvaihtoprojektissa

Middleware-järjestelmän versionvaihtoprojektissa projektisuunnitelma on tärkeä työkalu, johon voidaan palata projektin aikana tarkistamaan sovittuja asioita ja jota tulee päivittää muutoshallinnan menetelmin aina tarpeen vaatiessa. Middleware-järjestelmän versionvaihdon projektisuunnitelmassa tulee tuoda selkeästi ilmi projektin tavoite, ympäristön nykytila, toteutustapa, jolla tavoitteeseen päästään, sekä sen vaatimat resurssit, aikataulu ja kustannukset. Nykytilan kuvauksessa on hyvä kuvata Middleware-tuotteen integraatiot muihin sovelluksiin tai viitata dokumenttiin, jossa integraatiot on kuvattu. Toteutustavan kuvauksessa määritetään millä tavalla ja millä työkaluilla versionvaihto suoritetaan ja miten se todennetaan. Projektisuunnitelmassa tulee myös kuvata projektin ohjaus- ja seurantamenetelmät. Liitteessä 1 esitetään Asiakkaan integraatiokeskuksen WebSphere Message Broker-versionvaihtoprojektin projektisuunnitelman sisällysluettelo.

5.2.5 Projektin seuranta projektisuunnitelman perusteella

Kun versionvaihto-projektille on löytynyt sopiva toteutustapa ja resurssit, aikataulu sekä kustannukset on asetettu, kirjataan ne projektisuunnitelmaan ohjausryhmän tai muun päättävän elimen hyväksyttäviksi. Projektin edetessä projektisuunnitelman toteutumista seurataan ennalta määriteltyjen menettelyjen mukaan ohjausryhmän toimesta projektipäällikön raportoinnin perusteella. Projektin edetessä ja tilanteiden päivittyessä projektisuunnitelmaan tehdään tarvittaessa muutoksia muutoshallintaprosessia noudattaen ja arvioidaan resurssien tarve, aikataulu ja kustannukset uudelleen suhteessa muutoksiin ja siihenastisiin saavutuksiin. Projektipäällikkö on vastuussa projektisuunnitelman päivittämisestä. Projektisuunnitelman päivitys jatkuvan palvelun parantamisen mallilla on jatkuva iteratiivinen prosessi aina projektin päättämiseen asti.

6 Yhteenveto ja toimenpidesuositukset

Middleware-järjestelmien ylläpito- ja migraatioprojektien suunnittelu vaatii ympäristön ja sen integraatioiden hyvää tuntemusta. Ylläpitoprojekteissa keskeistä on nykytilan kartoittaminen ja kuvaaminen, migraation toteutustavan optimaalinen suunnittelu sekä kattava todennus. Palvelun jatkuvan parantamisen periaatteiden mukaisesti projektin suunnittelu on jatkuvasti tapahtuva iteratiivinen prosessi, joka tapahtuu projektin seurannan perusteella. Projektille asetetaan seurantapisteitä, joissa projektin etenemistä verrataan projektisuunnitelmassa asetettuihin tavoitteisiin.

Middleware-järjestelmien migraatioissa ympäristön keskeinen integraatiopiste päivitetään tuoreempaan versioon, mikä tarkoittaa kaikkien liittymien kattavaa todennusta migraation jälkeen. Tällaisissa sovellusten ajoympäristöön kohdistuvissa projekteissa konfiguraationhallinta on keskeisessä asemassa. Konfiguraationhallinnan kautta on mahdollista tarkistaa ympäristöön tehdyt muutokset ja tarvittaessa palauttaa ne ennalleen. Migraatioprojektien toteutus kannattaa suunnitella käyttämällä hyväksi mahdollisimman paljon automaatiota ja konfiguraationhallintaa. Ympäristöjen ylläpitoprojekteissa kaikki mahdollinen asennus-, migraatio-, konfigurointi- ja todennustyö tulisi tehdä mahdollisimman automatisoidusti. Automatisoidut asennus-, migraatio- ja konfiguraatioskriptit toimivat samalla myös eräänlaisena dokumentaationa, ovat talletettavissa konfiguraationhallintajärjestelmään ja sieltä hallittavissa keskitetysti.

Asiakkaan integraatiokeskuksen Brokerin versionvaihto-projektimalli noudattaa jo monilta osin jatkuvan palvelun parantamisen parhaita käytäntöjä. Projektin suunnitteluvaiheessa pyrittiin löytämään mahdollisimman tehokas, dokumentoitu ja uudelleenkäytettävä asennusmalli. Broker-sovellusten vaatimat migraatiot ja ympäristön asennukset toteutetaan automatisoidusti. Kaikki asennukset ja konfiguraatiot suoritetaan keskitetysti hallittavilla ja uudelleenkäytettävillä skripteillä. Migraatioprojektin tuotokset todennetaan myös automatisoiduilla konfiguraationhallintajärjestelmässä säilytettävillä regressiotestiskripteillä.

Projektin tuotantoon siirrot suunniteltiin myös toteuttaviksi ennalta määritellyn vuosisuunnittelukellon Release-ajankohtaan. Projektin etenemistä seurattiin tuntikirjausten

perusteella. Tuntikirjausjärjestelmään perustettiin omat tehtävät mm. skriptien suunnittelulle, ympäristön asennuksille, projektin hallinnalle sekä eri migraation vaatimille tehtäville per testitapaus. Projektiryhmän jäsenet kirjasivat projektille tehdyt tunnit tuntikirjausjärjestelmään, jonka perusteella raportoitiin projektin kustannuksista asiakkaalle ohjausryhmän kokouksissa ja viikkopalavereissa. Projektista rajattiin ulos muiden integraatiokeskuksen palvelinten ja jonomanagereiden versionnostot ja määriteltiin ne toteutettavaksi myöhemmin omana projektinaan. Rajaus aiheutti sen, että Broker-tuotteen migraatio-projektia ei suunnitelmien vastaisesti pystyttykään hoitamaan täysin muulta kehitykseltä näkymättömissä, sillä niin kehityksen tekemien muutosten kuin migraationkin testaus vaatii koko sanomaketjun todennusta, jolloin Broker-käsittelyä tuli vaihdella uuden ja vanhan ympäristön välillä. Tehokkaampaan ja näkymättömämpään migraatioon olisi päästy toteuttamalla koko integraatiokeskuksen ympäristö rinnakkaiseksi, eikä vain Brokereiden osalta. Toisaalta kokonaan rinnakkaisen ympäristön toteutus olisi tuonut projektille enemmän työmäärä ja siten pitkittänyt tätä pelkästään Brokereiden versionvaihtoon tähtäävää projektia. Projektin etenemistä seurataan kustannusten toteutumisena ja onnistuneesti ajettujen testicasejen määrällä. Mikäli Broker-sovelluskomponentit ja testicaset eivät olisi toisistaan riippuvaisia, voitaisiin mitata myös uudessa ympäristössä ajossa olevien Broker-toteutusten määrää.

Toimiva versionhallintajärjestelmä automatisoitune konfiguraatio-skripteineen mahdollistaisi myös entistä ketterämmän mallin versionvaihdon toteuttamiseen. Projektin toteutuksen suunnittelussa voisi hyödyntää jatkuvan integraation toimintatapoja, jolloin itse projekti olisi hallinnollisesti kevyt. Hyvällä suunnittelulla, versioiden hallinnalla ja raportoinnilla versionvaihdot olisi mahdollista suorittaa jatkuvan kehityksen rinnalla ilman perinteistä projektiorganisaatio-mallia. Ideaali-tilanteessa Middleware-tuotteiden versiopäivitykset hoidetaan muutaman hengen projektiryhmän toimesta osana jatkuvaa prosessia, jolloin versiopäivitys ei aiheuttaisi lainkaan katkoja kehitykselle. Pienemmissä versiopäivityksissä uudet versiot päivitettäisiin ympäristöön yöllisissä automatisoiduissa paketointi-, ympäristön uudelleenasennus- ja testiajoissa, joista raportoidaan mahdolliset toimimattomuudet. Suuremmissa versiopäivityksissä, jotka vaativat muutoksia toteutuskoodeihin, Middleware-tuotteista ja koko integraatiokeskuksesta tulisi perustaa rinnakkainen ympäristö, jossa jatkuvan integraation malliin automatisoidusti koottaisiin ja asennettaisiin migroiduista komponenteista paketteja testattavaksi uudessa ympäris-

tössä. Projektin seuranta ja päätöksenteko olisi keskitetty jatkuvan palvelun viikkopalavereihin.

Ympäristöjen hallittavuuden, sekä joustavamman ja tehokkaamman todennuksen osalta erityisesti testitapausten alustuksessa on vielä kehitettävää. Osa testitapauksista on riippuvaisia edellisten testitapausten tuotoksista. Testitapausskiptejä tulisi vielä kehittää siten, että jokainen testitapaus tai vähintäänkin testitapaus-kokonaisuus alustaisi ympäristön (tietokannan) vaatimaansa tilaan.

Projektin seuranta suunniteltiin toteutettavan asiakkaan (Asiakkaan IT Infra) kanssa yhteisissä viikkopalavereissa, sekä projektin omissa ohjausryhmän palavereissa. Koska Middleware-tuotteet ovat keskeinen osa integraatiokeskuksen ympäristöä, halutaan seurantaa toteuttaa tiiviisti, jotta mahdollisiin projektin aikana esiin tuleviin ongelmiin voidaan nopeasti puuttua.

7 Johtopäätökset ja jatkokehitysehdotukset

Olen työssäni usein projektipäällikkönä infrastruktuuri-projekteissa, joissa muutetaan olemassa olevaa ympäristöä joko kokonaan uuteen käyttöjärjestelmäympäristöön tai päivitetään jonkin olemassa olevan Middleware-tuotteen versiota. Yritysten prosesseihin perustuvat projektisuunnitelmapohjat on lähtökohtaisesti aina suunniteltu kehitysprojekteille, jonkin uuden sovelluksen tai toiminnallisuuden tuottamiseen.

7.1 Johtopäätökset

Perehdyin työni kirjoittamisen aikana useisiin projektinhallintaan liittyviin tutkimuksiin. Onnistuin löytämään muutamia ylläpitoprojekteihinkin liittyviä tutkimuksia ja vertasin tutkimusten tuloksia omiin kokemuksiini laajoista infrastruktuuriprojekteista. Kattava nykytilan kartoitus, mahdollisimman pitkälle viety automaatio ja huolellinen todennuksen suunnittelu tuli tutkimuksissa ja kirjoissa usein esiin. Versionvaihtoprojektien suunnittelussa myös projektin osituksella ja aikataulutuksella on tärkeä merkitys muun sovelluskehityksen kannalta.

Jatkuvan palvelun parantamisen kannalta ympäristön ylläpito- ja versionvaihtoprojektit tulisi suunnitella perustuen selkeästi mitattaviin arvoihin, kuten aikataulut, kustannukset, projektin eteneminen jne. Näiden arvojen toteutumista tulee seurata määrämuotoisissa kokouksissa ja projektisuunnitelmaa tulee päivittää seurannan perusteella koko projektin ajan.

Jatkuva palvelun parantamisen näkökulma ylläpitoprojekteissa tarkoittaa myös sitä, että ylläpitotyöt tulee suorittaa mahdollisimman näkymättömästi muulta kehitykseltä. Mahdolliset migraatiot aikataulutetaan siten, että kehityksen siirtäminen uudelle alustalle ei aiheuta huomattavia katkoja sovellusten kehitykselle, vaan sovelluskehitys ja varsinainen tuotanto siirtyvät joustavasti uudelle alustalle.

Voin hyödyntää tämän opinnäytetyöni tuloksia tulevaisuudessa työssäni projektipäällikkönä erilaisissa infrastruktuurin ylläpitoprojekteissa.

Projektinhallinnan perusteet ovat tulleet tutuksi vuosien varrella, mutta opinnäytetyöprosessi antoi myös paljon uutta tietoa ja muistutti vanhasta. Entuudestaan lähes täysin

uutta asiaa minulle olivat tutkimukset ja kirjoitukset konfiguraationhallinnan tuottamista mahdollisuuksista versionvaihtoprojekteissa.

7.2 Jatkokehitysehdotukset

Opinnäytetyössäni tutkin erityisesti versionnostoprojektin suunnittelua, mutta ylläpidon ja erilaisten versionnostojen ollessa suurissa sovellusympäristöissä jatkuvaa on niiden tutkimisessa ja tehostamisessa vielä paljon työsarkaa. Infrastruktuuriprojektit ovat laajoissa ympäristöissä todella suuria ja kalliita projekteja. Suunnittelun ohella niiden varsinaisessa toteutuksessa on paljon tutkittavaa ja mallinnettavaa. Erityisesti jatkuvan integraation menetelmät ja sen toteuttamistekniikat infrastruktuurin ylläpitoprojekteissa on kiinnostava ja erityisesti suuria sovellusympäristöjä hyödyttävä tutkimuskohde. Onnistuneen infrastruktuuriprojektin tavoitteena on olla niin kehittäjille kuin loppukäyttäjillekin mahdollisimman näkymätön ja nopea. Tämän tavoitteen saavuttaminen jatkuvan integraation menetelmin olisi erittäin hyödyllinen ja mielenkiintoinen tutkimuskohde. Jatkuvan integraation menetelmin toteutettu ympäristön varusohjelman versionnosto voitaisiin ideaalitilanteessa toteuttaa osana jatkuvaa kehitystä ilman erillistä projektia.

Myös automatisoitujen testitapausten hallinta ja niiden käyttäminen ylläpitoprojekteissa olisi hyvä tutkimusaihe, sillä monessa ympäristössä on käytössä automatisoituja testitapauksia, mutta niiden täysmittainen hyödyntäminen on vielä kesken. Automatisaation ulottaminen regressiotestien lisäksi myös yksikkötestien suoritukseen ylläpidon tai versionvaihdon aikana toisi merkittäviä kustannushyötyjä niin tulevaan kehittämiseen kuin regressiotestivaiheeseen, kun mahdolliset toimimattomuudet löydetään uudelleen käytettävillä testitapauksilla jo aikaisemmassa vaiheessa ja virheen syy on helpompi löytää pienemmästä kokonaisuudesta.

Kiinnostava tutkimuskohde voisi olla myös suurten sovellusympäristöjen varusohjelmien versionnostoprojektien osituksen ja aikataulutuksen suunnittelu. Minkälaisiin osakokonaisuuksiin projekti kannattaa osittaa ja miten tuotantoon siirrot tulee suunnitella suhteessa näihin osakokonaisuuksiin? Suurten migraatioprojektien organisaation ja erityisesti migraation toteuttavan projektiryhmän optimaalinen koko olisi hyvä tutkimuskohde. Tutkimuksessa voisi tunnistaa esim. keskitetyn migraatiotyöryhmän ja toisaalta

hajautetun, sovelluskohtaisen, migraatiotavan hyötyjä ja haittoja ja pyrkiä kuvaamaan optimaalinen organisaatio migraation ja todennuksen toteuttamiseen. Myös virtuaalisten ympäristöjen ja pilvipalveluiden hyödyntämistä ylläpitoprojekteissa olisi kiinnostavaa tutkia.

Lähteet

Ahonen, J. & Savolainen P. 2010. "Software Engineering Projects May Fail Before They Are Started: Post-mortem Analysis of Five Cancelled Projects." Journal of Systems and Software 83 (11): 2175–2187. doi:10.1016/j.jss.2010.06.023.

Al-Jaroodi, J. & Mohamed, N. 2012. "Service-oriented Middleware: A Survey." Journal of Network and Computer Applications 35. doi:10.1016/j.jnca.2011.07.013.

Asiakkaan IT-palvelut 2012a. Palvelukuvaus - Tiedonvälityspalvelu. IT-infrapalvelut - Tiedonvälityspalvelu v1.0.docx. Helsinki.

Asiakkaan IT-palvelut 2012b. Integraatioarkkitehtuuri - Integraation Hallintamalli. Integraatioarkkitehtuuri_Hallintamalli.docx. Helsinki.

Bennett, K, & Rajlich, V. 2000. "Software Maintenance and Evolution: a Roadmap." In Proceedings of the Conference on The Future of Software Engineering, 73–87. ICSE '00. New York, NY, USA: ACM. doi:10.1145/336512.336534. Luettavissa: http://doi.acm.org/10.1145/336512.336534. Luettu: 23.2.2013.

Digia Oyj 2012. Broker V8 Migration Plan. Projektisuunnitelma. Helsinki.

Digia Oyj 2013a. Digia in brief. Annual report 2012. Luettavissa: http://annualreport2012.digia.com/digia-now-and-in-the-future/digia-in-brief. Luettu: 19.2.2013.

Digia Oyj 2013b. Asiakkaan Integraatiokeskus arkkitehtuurikuvaus Broker. INT_Arkkitehtuurikuvaus_Broker.doc. Helsinki.

Digia Oyj 2013c. Asiakkaan Integraatiokeskus vuosisuunnitelma. EAI vuosisuunnitelma 2013_v0.6.doc. Helsinki.

Fleurey, F., Breton, E., Baudry, B., Nicolas, A & Jézéquel, J-M. 2007. "Model-Driven Engineering for Software Migration in a Large Industrial Context." In Model Driven Engineering Languages and Systems. Lecture Notes in Computer Science 4735. Springer Berlin Heidelberg. Luettavissa: http://link.springer.com/chapter/10.1007/978-3-540-75209-7_33. Luettu: 20.2.2013

Holmala, R. 23.2.2013. Palvelupäällikkö. Digia Oyj. Haastattelu. Helsinki.

Humble, J. & Farley, D. 2011. Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education. Indiana.

IEEE Std. 1219: Standard for Software Maintenance. Los Alamitos CA., USA. IEEE Computer Society Press, 1993.

Kelkar, S. A. 2013. Software Project Management: A Concise Study. 3. painos. PHI Learning Pvt. Ltd. Delhi.

Kerzner, H. 2004. Advanced Project Management: Best Practices on Implementation. John Wiley & Sons. New Jersey.

Kerzner, H. 2009. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. John Wiley & Sons. New Jersey.

Khadka, R., Reijnders, G., Saeidi, A., Jansen, S. & Hage, J. 2011. "A Method Engineering Based Legacy to SOA Migration Method." 27th IEEE International Conference on Software Maintenance (ICSM). doi:10.1109/ICSM.2011.6080783

Malmi, P. & Åkerlund K. 2013. Creative_Project_Management_v_1_0.pdf. Luettavissa: http://www.datavok.com/pdf/Creative_Project_Management_v_1_0.pdf. Luettu: 15.2.2010

Office of Government Commerce. Service Operation. 2007. ISBN 978 0 11 331046 3. 2. Painos. The Stationery Office. Lontoo.

Pelin, R. 2008. Projektihallinnan Käsikirja. 5. painos. Gummerus Kirjapaino Oy. Jyväskylä.

Project Management Institute, Inc. (PMBOK) 2008. A Guide to the Project Management Body of Knowledge. Project Management Institute, Inc. Pennsylvania.

Spalding, G. 2007. Continual Service Improvement Book. 1. painos. The Stationery Office. Lontoo.

Teppe, W. 2009. "The ARNO Project: Challenges and Experiences in a Large-Scale Industrial Software Migration Project." 13th European Conference on Software Maintenance and Reengineering, *2009. CSMR '09*, 149 –158. doi:10.1109/CSMR.2009.64.

Tsui, F. 2004. Managing Software Projects. Jones & Bartlett Learning. Lontoo.

Virtanen, T. 20.2.2013. Palvelusalkun haltija. Digia Oyj. Haastattelu. Helsinki.

Wysocki, R. 2010. Effective Software Project Management. John Wiley & Sons. Indianapolis.

Liitteet

Liite 1. WebSphere Message Broker-versionvaihtoprojektin sisällysluettelo.

1	Introduction
1.1	Description and project background
2	PROJECT CONTENTS
2.1	Vision
2.2	Key project stakeholders and user groups
2.3	Key project constraints
2.4	Project scope
2.5	Project outlines
2.6	Planned deliverables
3	IMPLEMENTATION PLAN
3.1	General issues
3.2	Applied processes
3.3	Project roadmap
3.4	Significant external dependencies
3.5	Use of Open Source libraries
3.6	Product liability
4	PROJECT RESOURCES
4.1	Project organization
4.2	Customer responsibilities
4.3	Tools
4.4	Environment
4.5	Resource usage
5	CONTROL PLAN
5.1	“Definition of Done”
5.2	Quality assurance and inspections
5.3	Testing
5.4	Configuration management
5.5	Communication
5.6	Key metrics
6	RISK MANAGEMENT
7	Version history, references and terminology