SOVELLUKSEN KÄYTTÖÖNOTTOPROSESSIN VAIHEET

Case YIS / MDM SC

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, Hämeenlinna, työn hyväksymispäivä

Juha Ässälä

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Visamäki, Hämeenlinna
Tietojenkäsittelyn Koulutusohjelma

Tekijä Juha Ässälä **Vuosi** 2012

Työn nimi Sovelluksen käyttöönottoprosessin vaiheet

TIIVISTELMÄ

Opinnäytetyön toimeksiantajana toimii YIT Information Services Oy. Työn case-sovelluksena on MDM SC –sovellus. Työ keskittyy pääasiassa kyseisen sovelluksen ja toimeksiantajan muiden sovellusten käyttöönotto-prosesseihin ja –menetelmiin. Aiheen valinta perustui pitkälti meneillä olleeseen sovelluksen käyttöönottoon.

Opinnäytetyön tavoitteena oli tutkia, millaisia sovelluksen käyttöönottoon tarkoitettuja menetelmiä on yleisesti olemassa. Case-tutkimuksen tarkoituksena oli selvittää YIS:llä käytössä olevat ja MDM SC –projektissa toteutuneet menetelmät.

Opinnäytetyön teoriaosuuden pääpaino on kolmella valitulla käyttöönottomenetelmällä, jotka esitellään työssä vaiheistuksineen. Työn tutkimusmenetelmä pohjautuu sovellusasiantuntijoiden haastatteluihin. Niiden lisäksi tutkimusmenetelmänä on käytetty osallistuvaa havainnointia. Työn aineistona on käytetty alan kirjallisuutta ja luotettavia internet-lähteitä.

Tutkimuksen tuloksena saatiin selville, että YIS:llä on virallisesti käytössä vain yrityksen omiin tarpeisiin muokattu ASAP-menetelmä. Case-sovelluksen käyttöönotossa ei tietoisesti ole käytetty mitään tiettyä menetelmää, mutta lähtökohtaisesti sen vaiheistus pohjautui ASAP-menetelmään. Käytetty prosessimalli muuttui projektin edetessä vastaamaan enemmän XP-menetelmän vaiheistusta ja toimintatapoja.

Johtopäätöksenä voidaan todeta, että sovelluksen käyttöönottomenetelmä tulee aina valita kyseisen projektin mukaan riippuen käyttöönotettavan sovelluksen koosta ja siitä, kuinka paljon asiakas osallistuu prosessiin. Kehitysehdotuksena YIS:lle suositellaan tutkielmaan liitetyn menetelmäluokitustaulun käyttämistä projektikohtaisesti.

Avainsanat käyttöönottoprosessi, käyttöönottomenetelmä, lineaarinen, iteratiivinen

Sivut 41 s. + liitteet 13 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Visamäki, Hämeenlinna
Degree Programme in Business Information Technology

Author Juha Ässälä **Year** 2012

Subject of Bachelor's thesis Phases of the Software Introduction Process

ABSTRACT

The present thesis is commissioned by YIT Information Services Oy. The thesis treats an application called MDM SC as a case-application. The thesis concentrates mainly on the application and besides that it handles other application introduction processes and methods used at YIS. The subject of this thesis was decided to be based on the application introduction which was going on at the time when the doing of this thesis was initiated.

The target of this thesis is to find out what kind of methods of application introduction exist in general. Still the main goal is to research all the introduction methods used in YIS and what kind of methods were actualized in the MDM SC project during the introduction process.

The theory of this thesis is outlined on three different introduction methods. Those methods are introduced with all the phases included. The main research method used in the thesis is the interviews. Experiences of the thesis author are used as a research method as well. The material used in the thesis is mostly gathered from literature of Information Technology and from reliable web sites.

As a result of the thesis it was found out that there is no other introduction method than a YIS-modified ASAP used in YIS as an official method. It was also found out that there were no other methods used in the case-application project. Basically the phases of the process leaned on the YIS ASAP-method, but it was turned to XP-method during the process.

As a conclusion of the thesis it can be mentioned that the method should be chosen based on the project aspect. It depends on the size of the application to be implemented and how much the commissioner of the application is involved in the process. As a development proposal for YIS it is recommended that a method classification attached in the thesis should be used every time when a new introduction project is started.

Keywords introduction process, -method, linear, iterative

Pages 41 p. + appendices 13 p.

Tutkimuksessa käytettävät termit ja lyhennekäsitteet	
YIS	YIT Information Services Oy
MDM SC	Lyhenteellä viitataan käyttöönotettavaan case-sovellukseen (tutkimuksen kannalta sovelluksen nimi ei ole oleellinen)
BizTalk	Integraatio- ja tiedostonsiirtotyökalu
Lineaarinen	Sarjallinen, järjestyksessä etenevä
Iteratiivinen	Pieninä palasina kierroksittain etenevä
Inkrementaalinen	Vaiheistettu toimitus, toimitus tehdään pienemmissä osissa
Refaktorointikäytäntö	Koodin muuttaminen niin, että varsinainen toiminnallisuus ei muutu, koodista tulee selkeämpää ja helpommin luettavaa
Iteraatio	Yhden pienen kokonaisuuden valmistuskierros
AM (Agile Modeling)	Mallintamiseen keskittyvä malli
Crystal	Menetelmägeneraattori, josta voidaan valita palasia tarpeen mukaan
Lean	Toyotan kehittämä prosessimalli, jota on myöhemmin kehitetty erikseen ohjelmistosuunnittelua varten
ASD (Adaptive Software Development)	Ketterä prosessimalli
SAP	Systems, Applications and Products in Data Processing, toiminnanohjausjärjestelmä
SAP Template	Toiminnanohjausjärjestelmälle tehty Euroopan laajuinen malli eri asetuksista ja säännöistä.
Delta-ajo	Ohjelmallinen muuttuneiden tietojen poiminta siirtotiedostolle

SISÄLLYS

1 JOHDANTO

Sovelluksien käyttöönottojen onnistuminen edellyttää aina prosessien vaiheistamista. Prosessille saadaan näin selkeät suuntaviivat ja tavoitteet, joiden mukaan prosessia viedään eteenpäin. Projektien onnistumiseen tarvitaan aina selkeät määrittelyt siitä, mitä tullaan tekemään missäkin prosessin vaiheessa. Käyttöönotoissa on erityisen tärkeää tehdä aina projektin alussa tarkat määrittelyt itse prosessin eteenpäin viemisestä ja siitä saatavasta lopputuloksesta. Jokaiselle vaiheelle tulee antaa rajapuitteet, joiden perusteella projektia viedään eteenpäin. Näin kaikki prosessiin osallistuvat tiedostavat parhaiten koko projektin kokonaiskuvan ja näin päästään kokonaisuuden kannalta parhaaseen lopputulokseen.

Opinnäytetyön aihe valittiin töissä ajankohtaisen sovelluksen käyttöönoton perusteella. Näin opinnäytetyön tekemisessä pystytään hyödyntämään työelämässä saatuja kokemuksia sovelluksen käyttöönottoprosesseista. Samalla opinnäytetutkimuksesta saadaan vuorovaikutteisesti oppia toimintatapoihin sekä pystytään kehittämään ja tuomaan uusia näkökulmia tuleviin sovelluksen käyttöönottoprosesseihin.

Työn keskeisenä ideana on selvittää, millaisia sovelluksen käyttöönoton prosessimalleja on olemassa. Erilaisista vaihtoehtoista valitaan parhaiten yrityksen käytäntöihin sopiva malli. Tarkemmassa käsittelyssä on kolme erilaista prosessimallia, jotka toimivat yhtä hyvin niin sovellusten käyttöönottoprosesseissa kuin sovellusten kehitysprojekteissa.

Sovellusten käyttöönoton prosessimalli nivoutuu hyvin läheisesti sovelluksen kehitysmalleihin, koska kehitysprojekteissa käytetään yleisesti ottaen samaa kaavaa. Näin ollen prosessin vaiheet ovat hyvin samankaltaiset kuin itse käyttöönottoprosessissa. Niiden välinen käsitteistö ja vaiheistus vastaavat käytännössä toisiaan, joten siksi työssä käytetään paljon viittauksia suoraan sovellusten kehitysprojektimalleihin.

Tutkimustyön empiriaosuudessa peilataan tulosten perusteella valittua prosessimallia yrityksen käytössä oleviin toimintatapoihin käyttöönotettavan sovelluksen käyttöönottoprosessia painottaen. Tehdyllä tutkimuksella pyritään löytämään yrityksen käytössä olevista prosesseista heikkoudet, joiden perusteella annetaan kehitysideoita yrityksen käyttöön.

Tutkimuksessa pyritään vastaamaan seuraaviin tutkimuskysymyksiin:
Mitä vaiheita yleisesti ottaen sovelluksen käyttöönottoprosessi sisältää?
Millainen käyttöönottomalli on toteutunut case-tapauksessa?
Millaisia käyttöönottomalleja case-yrityksessä on yleisesti käytössä?
Miten yrityksen käyttöönottoprosesseja pystytään parantamaan?

Työn tilaajana toimii YIT Information Services Oy, jonka pääasiallisena toimintaperiaatteena on palvella ja tukea YIT:n kaikkia yhtiöitä ympäri Eurooppaa tietoteknisissä haasteissa.

2 TUTKIMUKSEN AIHEEN RAJAUS JA TAVOITTEET

Tutkimustyön rajauksen perustana pidetään käyttöönotetun sovelluksen käyttöönottoprosessia. Analyyttista tutkimustyötä tehdään lähinnä kyseisen MDM SC –sovelluksen käyttöönoton prosessista. Tutkimustyön tueksi haastatellaan myös muita henkilöitä, jotka ovat työskennelleet YIS:n käyttöönottoprojektien parissa. Heiltä saadaan laajempaa tietoa yrityksessä käytettävistä sovelluksen käyttöönottoprosessin malleista.

Rajauksen ulkopuolelle jäävät sovellusten käyttöönottoprosessiin kuuluvat taloudellisten lukujen toteuma ja seuranta. Työssä ei myöskään käsitellä sovellusten osalta teknisiä yksityiskohtia. Lisäksi rajauksen ulkopuolelle jäävät muutosjohtaminen ja sovellusten käyttöönottoon liittyvä koulutus. Työssä ei käsitellä sovellusten käyttöönottoon liittyviä tukipalveluita. SAP-toiminnanohjausjärjestelmän roll-out projektit on myös rajattu pois, koska työssä keskitytään lähinnä pienkehityksen sovellushankkeisiin.

Tutkimustyön tavoitteena on saada mahdollisimman laaja näkemys yleisesti käytössä olevista sovelluksen käyttöönottoprosessien vaiheistuksista, ja niihin perustuen valitaan parhaimmat ja soveltuvimmat mallit YIS:n käyttötarkoituksiin. Tutkimustyön empiriaosuudessa selvitetään yrityksessä käytössä olevien sovellusten käyttöönottoprosessien toteutuksia haastattelemalla yrityksen eri henkilöitä. Haastatteluiden avulla pyritään saamaan mahdollisimman laaja ymmärrys YIS:llä käytössä olevista vaiheistusmalleista. Tutkimustyön päätavoitteena on kuitenkin analysoida käyttöönotettavan sovelluksen käyttöönottoprosessin vaiheistusta ja saada vastaaviin käyttöönottoprojekteihin parannus- ja kehitysideoita yrityksen käyttöön.

Käyttöönoton prosessimalli nivoutuu hyvin läheisesti sovelluksen kehitysmalleihin ja niihin liittyviin vaiheistuksiin. Tutkimuksen tuloksena tulleita johtopäätöksiä voidaan siis käyttää myös sovellusten kehitystöihin, koska sovelluksen kehityksessä suoritetaan samat vaiheet kuin uutta sovellusta käyttöönotettaessa. Työn tuloksena valittavaa mallia voidaan siten hyödyntää jatkossa myös MDM SC:n jatkokehityksessä.

Opinnäytetyön tavoitteena on myös tuottaa yritykselle uutta tietoa sovellusten käyttöönottoprosessien erilaisista vaiheistusmenetelmistä ja siten tukea yritystä käyttöönottoprosessien onnistumisessa tulevaisuudessa.

3 TUTKIMUKSEN TIETOPERUSTA

Teoriaosuudessa käytetään erilaisia lähteitä tietoperustan saavuttamiseksi. Pääasiallisena tietoperustana käytetään alan kirjallisuutta ja internetistä löytyviä luotettavia lähteitä. Käytännön empiriaosuuden tietoperustana käytetään osallistuvaa havainnointia meneillään olevassa sovelluksen käyttöönottoprojektissa. Lisäksi tutkimuksen tukena käytetään haastatteluita.

Opinnäytetyötutkimuksen taustateoria perustuu lähinnä ohjelmistojen käyttöönotto- ja kehitysprosesseissa yleisesti käytettyihin vaiheistusmalleihin. Erilaisia malleja vertailemalla ja mahdollisesti yhdistelemällä pyritään löytämään yrityksen käyttöön paras mahdollinen vaiheistusmenetelmä erilaisiin käyttötarkoituksiin.

Tutkimuksen teoreettinen viitekehys muodostuu kolmesta valitusta sovelluksen käyttöönottoprosessimenetelmässä. Näiden kolmen valitun menetelmän lisäksi esitellään myös SAP:n lanseeraama ASAP-menetelmä. Vertailemalla kolmea valittua menetelmää yrityksen käytössä oleviin prosessimalleihin ja MDM SC –sovelluksen käyttöönottoprosessin ratkaisuihin pystytään menetelmiä keskenään peilaamalla saamaan parempaa näkökulmaa yrityksen tämän hetken tilanteesta käytössä olevien menetelmien suhteen. Samalla saadaan näkökulmia käytössä olevien käyttöönottoprosessien heikoista kohdista sekä pystytään osoittamaan kehityskohdat yrityksen tarpeisiin.

4 TUTKIMUKSEN TEOREETTINEN TAUSTA

Lähtökohtaisesti vertailuun otettiin tutkittavaksi yleisimmin tunnustettuja ja yleisimmin käytössä olevia prosessimallien menetelmiä. Vertailtavina menetelminä tutkimuksessa käytetään vesiputousmallia, Scrum- ja XP –menetelmiä. Lisäksi esitellään SAP:n kehittelemä ASAP-menetelmä, jonka perusteella YIS:lle on tehty omaan käyttöön paremmin soveltuva menetelmä. Tästä menetelmästä kerrotaan tarkemmin kappaleessa 7.2.

Vesiputousmalli edustaa ns. perinteistä mallia, joka pohjautuu pitkälti tarkkaan suunnitteluun, ja sen järjestelmälliseen toteuttamiseen. Kaksi muuta tarkasteltavaa mallia ovat ns. ketteriä malleja, joiden toimintaperiaate perustuu lähinnä siihen, että missä tahansa prosessin vaiheessa voidaan tehdä asioita aina uudestaan, jolloin asioita ei suunnitella kovinkaan pitkälle eteenpäin.

4.1 Vesiputousmalli

Vesiputousmalli lienee kaikkein tunnetuin suunnitteluvetoisista ohjelmistokehitysmalleista. Sen ensimmäisen virallisen määritelmän esitti Winston W. Royce vuonna 1970 artikkelissa ”Managing the Development of Large Software Systems”. Sana vesiputous tulee mallin tavasta edetä vaiheesta toiseen erittäin suunnitelmallisesti projektin alusta loppuun, ikään kuin vesiputous valuessaan vuorenrinnettä alas. Edellinen vaihe tulee aina saada päätökseen ennen siirtymistä seuraavaan vaiheeseen, sillä paluuta eri vaiheiden välillä ei ole (Royce, 1977).

Alkuperäinen vesiputousmalli koostui viidestä eri vaiheesta, jotka ovat vaatimusmäärittely, suunnittelu, toteutus, verifiointi sekä validointi ja ylläpito (Royce 1977). Tämän jälkeen vesiputousmallista on tehty monta erilaista versiota, joista esitellään kaksi eri vaihtoehtoa tässä luvussa – perinteinen lineaarinen malli ja iteratiivinen vesiputousmalli.

4.1.1 Vesiputousmalli yleisesti

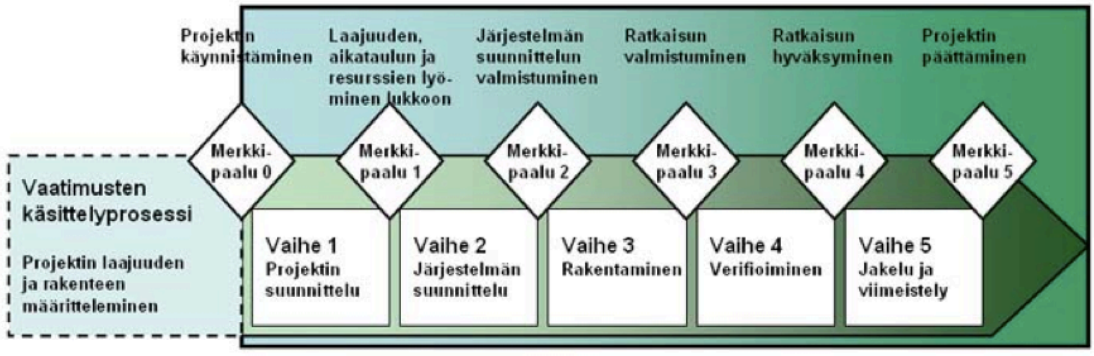
Suunnittelu on vesiputousmallissa selvästi tärkein osa-alue. Vesiputousmallin käyttämiseen sisältyy paljon dokumenttien yksityiskohtaisia ohjeistuksia ja malleja, joita tarvitaan erilaisten prosessien, ohjelmien ja projektien toteutuksessa. Mallien avulla pystytään määrittelemään prosessien rajapinnat tai projektiin osallistuvien roolit ja vastuut. Näin kustannusarvio voidaan toimittaa asiakkaalle jo projektin alussa. Mallin selkeyden edesauttamana projektin eri vaiheet seuraavat toisiaan linjakkaasti. Tarkistuspisteiden läpäisy ja niihin liittyvät dokumentit ovat aina edellytys seuraavan vaiheen aloittamiselle siten, että edellisen vaiheen tarkistuspisteen dokumentin hyväksymisen jälkeen voidaan siirtyä seuraavaan vaiheeseen. Tarkka dokumentoiminen prosessien aikana syntyvistä tiedoista varmistaa esimerkiksi, että tiettyjen avainhenkilön poistuminen projektista ei kaada koko projektia ja, että tarvittaessa projektista poistuvat henkilöt voidaan korvata ilman ylimääräisen koulutuksen tarvetta. Joka tapauksessa ison ja

monimutkaisen projektin loppuun vieminen täsmälleen suunnitelman mukaisesti on yleisesti ottaen mahdotonta. Jokaiseen kehitysprojektiin kuuluu aina suuri määrä epävarmuustekijöitä. Erityisesti vesiputousmallia käyttämällä projektin vaiheissa peruuttaminen on työlästä ja tulee kalliiksi, sillä todellisuudessa kehitysprojektien vaiheet ovat harvoin toisistaan riippumattomia. Tästä johtuen voidaan joutua uusimaan jopa kaikki jo kertaalleen suoritetut vaiheet. Tyypillisenä vesiputousmallin voidaan myös pitää sitä, että asiakkaalle voidaan esitellä tuloksia vasta projektin lopussa. (Hybridimenetelmä, 2009.)

Monien lähteiden mukaan varsinkin lineaarinen vesiputousmalli on liian kankea pienten sovelluksen kehittämiseen sekä sovellusten käyttöönottoprosessien mallina käytettäväksi. Tämä perustuu lähinnä siihen, että jo suunnitteluvaiheessa sovelluksesta tehdään lopullinen määritelmä, eikä tämän jälkeen ole enää mahdollista ottaa prosessissa taka-askelia. Vesiputousmalli soveltuu siis useimmiten suurempien projektien käyttöön, joissa annetaan kunkin prosessin vaiheen toteutukselle enemmän ylimääräistä aikaa verrattuna pienempien projektien aikataulutukseen. Näin ollen kunkin vaiheen toteutus aikataulussa mahdollistetaan. Ylimääräistä aikaa ei ole varaa antaa pienemmissä projekteissa, ja juuri se tekee lineaarisen vesiputousmallin käyttämisen pienemmissä projekteissa tehottomaksi. (Hybridimenetelmä, 2009.)

Ohjelmistokehitysprojektit ovat edelleenkin kuitenkin mallin yleisiä käyttökohteita. Mallille tyypilliset pitkät etukäteissuunnittelut ja muutoksissa joustamattomuus ovat joka tapauksessa vaikeasti sovitettavissa ohjelmistotekniikan jatkuvasti vaihtelevaan ympäristöön. Nykyaikaisissa ohjelmistokehitysprojekteissa on tyypillistä, että niiltä vaaditaan yhä enemmän joustavuutta ja tulokset halutaan näkyville nopealla aikataululla ja lisäksi prosessin sisällä vaaditaan nopeaa reagointikykyä toistuviin muutostarpeisiin. Tavoitteita asettamalla saadaan projektissa toimimiselle yleiset suuntaviivat, mutta projektisuunnitelman tarkka toteutus voi edelleen olla vaikeaa. Yleensä suunnitelmaa pystytään tarkentamaan vasta prosessin edetessä. (Hybridimenetelmä, 2009.)

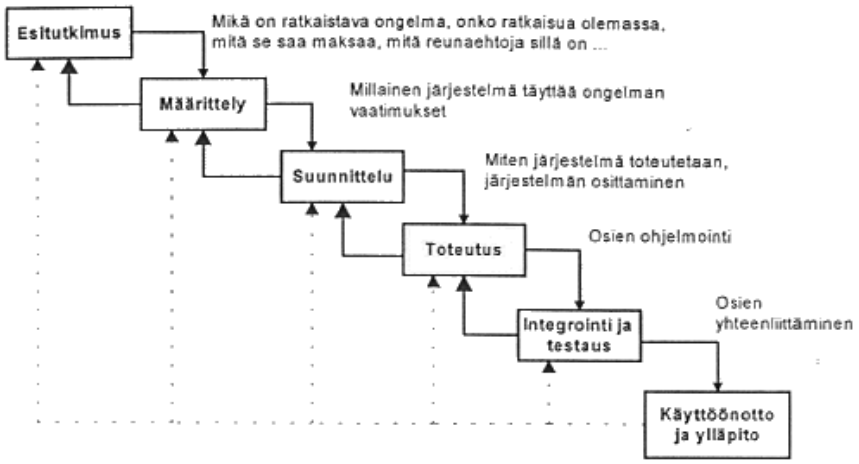
4.1.2 Lineaarisen vesiputousmallin vaiheet



Kuva 1. Lineaarisen vesiputousmallin vaiheet (Hybridimenetelmä: Vesiputousmalli ja Scrum 2009)

Lineaarisen vesiputousmallin prosessiin kuuluvia vaiheita kuvataan yleisesti monessa eri muodossa. Yleisimmin prosessin vaiheiden eteneminen kuvataan portaittaisesti ja silloin prosessin kuvaus muistuttaa vesiputousta. Prosessin etenemisen perusperiaate ja vaiheistus on jokaisessa vesiputousmallin versiossa käytännössä sama. Prosessin eteneminen jaetaan viiteen eri kuvassa 1 näkyvään vaiheeseen. Jokainen erillinen vaihe lopetetaan aina samaisessa kuvassa mainittuihin merkkipaaluihin. (Hybridimenetelmä, 2009.)

4.1.3 Iteroivan vesiputousmallin vaiheet



Kuva 2. Iteroiva vesiputousmalli (Haikala & Märijärvi 2002)

Iteroivassa mallissa vaiheet pysyvät koko prosessin elinkaaren ajan sinällään samanlaisina kuin perinteisessä lineaarisessa mallissa, mutta sen iteratiivisuuden takia on mahdollista palata aina joko edelliseen tai jopa aiemmin suoritettuun vaiheeseen.

Haikalan ja Märijärven mukaan sovelluksen kehitysprosessi alkaa määrittelyvaiheesta, josta se etenee eri vaiheiden kautta sovelluksen lopulliseen

käyttöönottoon ja ylläpitoon. Tarvittaessa ennen määrittelyä voidaan tehdä esitutkimus, jos ongelman ratkaisumahdollisuuksista halutaan varmistua ennen käyttöönottoprojektin aloittamista. Kunkin elinkaaren vaiheen (lukuun ottamatta ylläpitovaihetta) päätteeksi suoritetaan laadunvarmistus, joka on yleensä vaihetuotosten virallinen katselmointi. Laadunvarmistuksen jälkeen siirrytään seuraavaan vaiheeseen ja edellisen vaiheen tuotokset toimivat seuraavan vaiheen syötteinä.

Esitutkimusvaiheen tarkoituksena on selvittää järjestelmän käyttöönottamisen syyt tai mahdolliset esteet järjestelmän käyttöönotolle. Esitutkimusvaihetta kutsutaan usein myös tarvekartoitukseksi, jossa selvitetään asiakkaan yleisen tason vaatimukset järjestelmälle. Lisäksi esitutkimusvaiheessa arvioidaan järjestelmän teknistä ja liiketaloudellista toteutettavuutta käytettävissä olevien teknologioiden sekä arvioitujen työmäärien ja kustannusten perusteella.

Määrittelyvaiheessa kerätään asiakkaalta vaatimukset käyttöönotettavalle järjestelmälle, jos sitä ei jo esitutkimusvaiheessa aiemmin ole tehty. Määrittelyvaiheessa näistä asiakkaan yleisistä vaatimuksista johdetaan ohjelmistovaatimuksia, jotka ovat käyttöönotettavan järjestelmän ominaisuuksia. Tässä vaiheessa määritellään kaikki ohjelmiston toiminnot ja toteutukselle asetettavat ei-toiminnalliset vaatimukset sekä rajoitukset. Periaatteena on, että määrittelyvaihe tuottaa toiminnallisen määrittelyn, jossa on kuvattu kaikki käyttöönotettavalle ohjelmistolle asetetut vaatimukset. Nämä vaatimukset täytettyään ohjelmisto on valmis täyttämään asiakkaan esittämät asiakasvaatimukset. Lisäksi määrittelyvaiheessa tuotetaan testaussuunnitelmaan testitapauksia, joilla ominaisuuksien toiminta testataan testausvaiheessa.

Suunnitteluvaiheessa suunnitellaan sellaisen järjestelmän arkkitehtuuri ja tekninen toteutus, joka sisältää kaikki toiminnallisen määrittelyn toiminnot. Tällä tavoin käyttöönotettu järjestelmä täyttää alkuperäiset asiakasvaatimukset. Suunnitteluvaihe voidaan jakaa kahteen tai useampaan alivaiheeseen. Yleisimmin se jaetaan arkkitehtuurisuunnitteluun ja moduulisuunnitteluun. Arkkitehtuurisuunnittelussa suunnitellaan järjestelmän tekninen yleisarkkitehtuuri ja moduulijako. Moduulisuunnittelussa suunnitellaan yksittäiset moduulit ja niiden tekninen toteutustapa. Suunnitteluvaihe tuottaa teknisen määrittelyn, joka kuvaa, millaisella teknisellä ratkaisulla toiminnallisen määrittelyn mukainen järjestelmä voidaan toteuttaa. Lisäksi suunnitteluvaiheessa tuotetaan teknisiä testitapauksia testaussuunnitelmaan.

Toteutusvaiheessa suoritetaan varsinainen ohjelmointityö. Siinä järjestelmä toteutetaan aiemmin valmistuneiden teknisten ja toiminnallisten määritysten pohjalta. Tämän vaiheen päättyessä järjestelmästä on olemassa kaikki asiakasvaatimukset täyttävä, ja kaikki määritetyt ominaisuudet sisältävä versio.

Integrointi- ja testausvaiheessa järjestelmä integroidaan ja testataan kokonaisuutena. Testausvaihe jaetaan yleensä moduuli-, integrointi- ja järjes-

telmätestaukseen. Moduulitestauksella varmistetaan, että yksittäiset moduulit täyttävät vaatimuksensa ja toimivat virheettömästi. Integrointitestauksessa testataan kokonaisuudeksi integroitujen moduulien yhteistoimintaa. Kun kaikki moduulit on integroitu yhdeksi kokonaisuudeksi ja integraatiotestaus on suoritettu, suoritetaan järjestelmätestaus, jossa järjestelmän kaikki ominaisuudet varmistetaan ja testataan. Kaikki testaukset suoritetaan testaussuunnitelman testitapauksien mukaisesti ja testien tulokset kirjataan testausraportteihin.

Valmis ja testattu järjestelmä toimitetaan asiakkaalle ja asennetaan käyttöympäristöönsä. Asennuksen ja käyttöönoton jälkeen aloitetaan ylläpitovaihe, jossa järjestelmätoimittaja tarjoaa asiakkaalle ylläpitopalvelua. Ylläpito sisältää yleensä tukipalvelut, ohjelmistovirheiden korjauksia, ohjelmiston päivitysversioita ja joskus myös uusien ominaisuuksien kehittämistä. Ylläpitovaihe kestää ohjelmiston elinkaaren loppuun saakka, eli se päättyy, kun kukaan ei enää kehitä tai käytä järjestelmää. Käytännössä ylläpitovaihe päättyy usein toimittajan ilmoitukseen ylläpidon lopettamisesta ja uuden sovelluksen tarjoamiseen lopetetun tilalle. (Haikala & Märijärvi 2002, 37-41).

4.1.4 Kommentteja vesiputousmallista

Sovellusten ja toiminnanohjausjärjestelmien toteutusprojektien hallintaan tarkoitetut menetelmät ovat kehittyneet aikojen kuluessa huomattavasti ja nykypäivänä tuskin mikään yritys tai yhteisö kuvittelee kykenevänsä määrittelemään tarkasti kaikki järjestelmän toiminnallisuudet osa-alueet. Vanhanaikainen vesiputouksen lineaarinen toimintamalli on auttamattomasti vanhentunut. Sen tilalle on tullut vaihtoehdoksi useita erilaisia ketteriä menetelmiä, joiden selväksi eduksi voidaan laskea asioiden selvästi nopeampi toteutuminen. (Mäntylahti, 2009b.)

Vesiputousmallista löydetään edelleen hyviä puolia. Siitä pidetään vieläkin hyvänä vaihtoehtona sen takia, että se sopii projekteihin, joissa on selvät tavoitteet. Sen kaikki vaiheet on jaoteltu selkeästi. Sen selvien alku- ja loppuvaiheiden lisäksi siihen liittyvät dokumentit ovat aina ajan mukaisia. Se on menetelmänä perinteinen ja tuttu ja sen perusominaisuudet on helppo omaksua. Menetelmänä se on todella yksinkertainen käyttää. Malli sisältää käytännössä kaikki tarvittavat vaiheet. Menetelmä soveltuu sellaiseen projektiin, jossa tavoitteet on asetettu selkeästi ja yksiselitteisesti. Lopputuloksena saadaan vaivattomasti johdettava ja ennalta ennakoitava projekti. (Balci ym., 2001.)

Vesiputousmallin hyvien ominaisuuksien vastapainona eli sen huonoina puolina pidetään sitä, että sillä toteutettuja projekteja on vaikea tai jopa turha peruuttaa taaksepäin. Menetelmä omaksuu nykypäivän prosessille tavallisen iteratiivisuuden heikosti. Tuloksien saaminen asiakkaalle esitettäväksi varsin myöhäisessä projektin vaiheessa lasketaan yleisesti sen huonoksi ominaisuudeksi. Nykyaikaiset ohjelmistotototeutuksien projektit toimivat käytännössä erittäin harvoin lineaarisesti. Menetelmä edellyttää käsittelemään liian suuria osa-alueita kerrallaan. Osa-alueesta toiseen siir-

tyminen edellyttää erinäisiä tarkastuksia ja hyväksymiskäytäntöjä, mikä tekee käytettävästä menetelmästä mallina jäykän. Lisäksi tällä menetelmällä pystytään toteuttamaan toimiva järjestelmä asiakkaan käyttöön vasta projektin myöhäisessä vaiheessa. Tämä tarkoittaa sitä, että vasta projektin loppuvaiheessa asiakkaalla on mahdollisuus todeta, miten sen toiminnallisuudet toimivat loppukäyttäjän näkökulmasta tarkasteltuna ja millainen sovelluksen käyttöliittymä on ulkonäöltään. Yhtenä menetelmän haasteena nähdään myös se, että aivan projektin loppuvaiheessa ilmenneet muutostarpeet tulevat järjestäen erittäin kalliiksi asiakkaalle projektin kokoon verrattuna. (Balci ym., 2001.)

Vesiputousmenetelmän käyttäminen sovellushankkeiden projekteissa nähdään liikaa aikaa ja rahaa vievänä ainakin asianmukaisten määrittelyiden tekemisen suhteen. Määrittelydokumentti on itsessään vain kasa paperia, joka ei sellaisenaan tuota mitään, ennen kuin joku toteuttaa sen avulla konkreettisen sovellushankkeen ja siitä saatavan lopputuotteen. ”Onko järkevää käyttää kolmasosa projektin budjetista määrittelyn kirjoittamiseen ja huomata toteutustyön edetessä, että kaikkea ei sittenkään osattu ennakoida?” (Mäntylahti, 2009a.)

4.2 Ketterät menetelmät yleisesti

Ketterät ohjelmistokehitykseen tarkoitetut menetelmät tulivat kiinnostuksen kohteeksi 1990-luvulla. Perinteisen ohjelmistosuunnittelun ja -tuotannon mallit ja menetelmät eivät pystyneet kokonaan vastaamaan nopeasti kasvavan verkko- ja mobiilisovellusten kehitystyön tarpeisiin. Markkinoilla löytyi siis selvä tarve kevyemmille ja joustavammille menetelmille, joita hyödyntäen kyettiin tuottamaan nopeasti käyttökelpoisia ohjelmistoja ja näin pystyttiin vastaamaan nopeammin asiakkaiden vaatimuksiin. (Hypermedian opetus, 2008b.)

Ketterien menetelmien tyypillisiä ominaisuuksia ovat nopeus ja yksinkertaisuus. Ensisijaisena tavoitteena on toteuttaa sovellukseen vain tärkeimmät ja keskeisimmät toiminnot. Ensimmäinen sovelluksesta julkaistava versio julkistetaan heti tämän jälkeen ja julkaistusta sovelluksesta kerätään heti palaute asiakkaalta. Asiakkaalta saadun palautteen perusteella sovellukseen tehdään pieniä muutoksia. Ketterän menetelmän perustarkoituksena on siis kehittää sovellusta vähitellen eteenpäin aina kullakin kierroksella uusia ominaisuuksia lisäten. Sovelluksesta pyritään julkistamaan uusia versioita hyvin tiiviissä tahdissa. Tällä toimintamallilla saadaan asiakkaalta kerättyä tiivis ja ajanmukainen palaute. Inkrementaalisen kehitystyön päämääränä on siis pystyä reagoimaan nopeasti asiakkaiden tarpeisiin ja teknologian tuomiin nopeisiin muutoksiin. Keskeistä koko menetelmämuodossa on iteratiivisuus, jolloin eri prosessin vaiheet limittyvät pääsääntöisesti hyvinkin paljon keskenään. (Hypermedian opetus 2008b.)

Ketterä kehittäminen on yleisesti ottaen saanut koko ajan enemmän huomiota. Ketterät menetelmät eivät kuitenkaan ole niin yksiselitteisiä eikä menetelmien soveltaminen käytännön projekteissa todellakaan ole niin yksinkertaista. (Poimala ym. 2011a.)

Ketterien menetelmien perusperiaatteet pohjautuvat siihen, että ei ole olemassa yhtä ja ainoaa oikeaa tapaa toteuttaa tavoiteltu lopputuote. Tästä johtuen vain jotkut ketteristä menetelmistä ovat tarkasti säädeltyjä ohjeistoja siitä, miten missäkin vaiheessa tulee menetellä. Pakollisten käytäntöjen sijaan manifestin taustalla on kokoelma periaatteita, joita noudattamalla uskotaan päästävän hyvään lopputulokseen. (Agile Alliance 2001.)

Alla olevassa taulukossa 1 on esitelty Poimalan mukaan ketterissä menetelmissä yleisimmin esiintyvät periaatteet. Eri käytäntöjen hyödyntäminen projekteissa on todellisuudessa aina riippuvainen monesta eri muuttujasta, joten käytäntöjä voidaan hyödyntää erilailla aina projektin lähtökohdista riippuen.

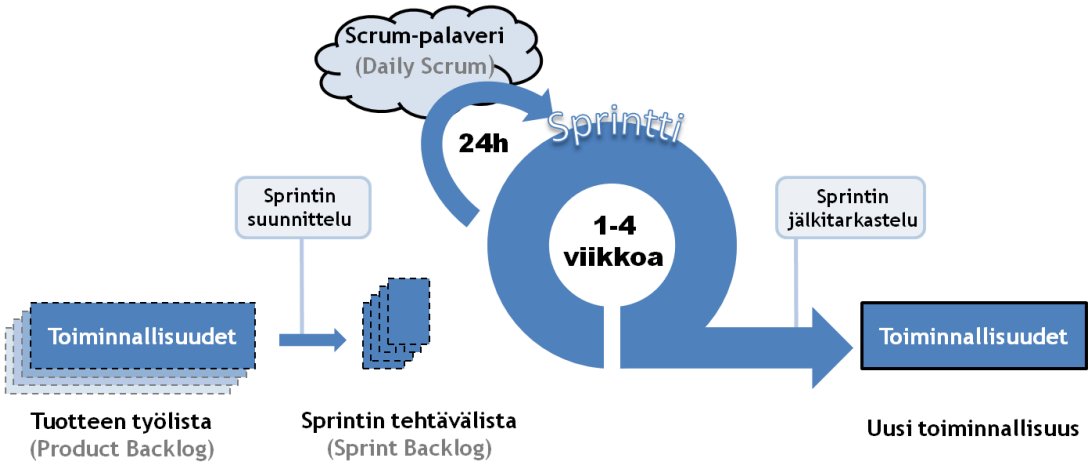
Taulukko 1. Ketterien mallien yhteiset periaatteet (Poimala ym. 2011a.)

Periaate	**Periaatteen kuvaus**
Asiakkaan tyydyttäminen	Tärkein tehtävä on tyydyttää asiakas toimittamalla nopeasti ja jatkuvasti arvokas sovellus.
Muutoksien hyväksyntä	Muuttuvat vaatimukset sallitaan, myös projektin loppupuolella. Hyväksytään muutokset, jotta asiakkaalle saadaan kilpailuetua.
Julkaisujen tekeminen aikaisin ja usein	Julkaistaan testattu sovellus usein, parin viikon - parin kuukauden välein. Tavoitteena aina mahdollisimman nopeat syklit.
Aina lähellä olevat asiakkaat	Toteuttajien ja liiketoiminnan tuntijoiden on toimittava yhdessä tai jopa samassa tilassa päivittäin koko projektin ajan.
Tekijöille annettu luottamuksenosoitus	Projektit perustetaan motivoituneiden henkilöiden ympärille. Toimijoille annetaan laadukas toimintaympäristö. Toimijat saavat tarvitsemansa tuen ja heidän työntekoonsa luotetaan.
Kasvotusten keskustelu	Välillisen viestinnän sijaan tehokkain kommunikointikeino on kasvokkain keskustelu.
Sovelluksen toimivuus	Ensisijainen edistymisen mittari
Työtahdin tasaisuus	Sponsoreiden, kehittäjien ja käyttäjien on pystyttävä ylläpitämään tasaista työtahtia jatkuvasti.
Teknisen laadun arvostaminen	Arvostetaan teknistä laatua ja hyvää suunnittelua.
Yksinkertaisuus	Tekemättä jätetyn työn määrän maksimointi on olennaista.
Tiimin sisäinen organisoituminen	Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat kehittyvät tiimeistä, jotka organisoivat itse toimintansa.
Itsetarkastelu tiimin sisällä	Tiimissä mietitään omaa tehokkuutta ja toimintatapoja säännöllisesti.

Ketteriä menetelmiä on useita erilaisia ja niistä on muokattu aikojen saatossa monia eri versioita soveltuen vain kyseessä olevaan projektiin tai

prosessiin. Yleisimmin käytössä olevia ketteriä menetelmiä ovat mm. AM, Crystal, Lean, ASD, Scrum ja XP. Näistä mainituista ketteristä menetelmistä tullaan tutkimuksessa keskittymään tarkemmin ainoastaan Scrum- ja XP –menetelmiin.

4.3 Scrum-menetelmän vaiheet



Kuva 3. Scrum-prosessi (Te.., A.-M. 2008b)

Scrum-prosessin on kehittänyt vuonna 1993 amerikkalainen Jeff Sutherland yhdessä Ken Schwaberin kanssa. Termi ”scrum” on lainattu Rugby-pelistä. Sutherland sai idean nimestä luettuaan tutkimusta, jossa verrattiin huipputehokkaita, ristiin toimivia tiimejä Rugbyn aloitusmuodostelmaan. Nykyisin Scrum on yksi yleisimmin käytetyistä ketterän ohjelmistokehityksen muodoista. (Hybridimenetelmä, 2009.)

Scrum on ketterän ohjelmistokehityksen malli, jota voidaan soveltaa myös verkkopalveluiden suunnitteluun. Se on iteratiivinen ja inkrementaalinen; tavoitteena oleva tuote rakentuu vähitellen täydellisemmäksi ja valmiimmaksi useiden toteutuskierrosten aikana. Tässä kappaleessa selitetään lyhyesti Scrum:n periaatteet virallisen oppaan (Schwaber & Sutherland, 2009) mukaan.

Scrum:ssa kehitys tapahtuu korkeintaan neljän viikon pituisissa pyrähdyksissä, eli sprinteissä. Jokaisen sprintin alussa pidetään suunnittelukokous, jossa suunnitellaan sprintin aikana toteutettavat toiminnallisuudet. Päivittäin pidetään 15 minuutin mittainen Scrum-palaveri, jossa tiimi tekee tilannekatsauksen jakamalla tietoa toisilleen tehdyistä töistä ja ilmenneistä ongelmista.

Sprinttien jälkitarkastelussa pidetään kaksi kokousta. Sprintin katselmoinnissa tarkastellaan sprintin onnistumista tuotteen kannalta, jolloin toteutetut toiminnallisuudet esitellään yhdessä tuoteomistajan ja muiden sidosryhmien kanssa. Sprintin retrospektiivissä tarkastellaan itse prosessia ja pyritään kehittämään työtapoja paremmiksi. (Schwaber & Sutherland, 2009.)

Scrum-menetelmän tarkoituksena on tarjota sovelluksien kehitykseen ohjenuoran, jonka perusteella projektia voidaan viedä eteenpäin. Scrum ei menetelmänä puutu ruohonjuuritason insinöörien ylläpitämiin käytäntöihin, vaan sitä käyttämällä painotetaan lähinnä prosessin vaiheistamista ja jatkuvasti toistuvaa kontrollointia prosessin edistymisestä. Poimalan mukaan nämä perusperiaatteet johdattavatkin asiakkaita hyvin ketterien menetelmien käyttämiseen eri ohjelmistoprojekteissa. (Poimala ym. 2011b)

4.3.1 Scrum-projektin roolit

Scrum-projekteissa käytetään ainoastaan kolmea eri roolia. Ne ovat tiimi, Scrum-mestari ja tuotteen omistaja. Tiimi on ristiin toimiva, itseohjautuva ryhmä, joka organisoi oman työnsä. Scrum-tiimiin ei kuulu perinteisiä ohjelmistokehitykseen liittyviä rooleja, kuten suunnittelija, arkkitehti, ohjelmoija, testaaja tai käyttöliittymäsuunnittelija. Jäsenet valitsevat listalta itse päivittäiset tehtävänsä. Tiimi sitoutuu toimittamaan tietyt toiminnallisuudet asiakkaan käyttöön ja sen jäsenillä on kaikki siihen tarvittava asiantuntemus. Jokaisen sprintin lopputuloksena saadaan toimituskelpoinen tuote tai jokin lisäosa. Projektin antamien suuntaviivojen puitteissa tiimi saa vapauden valita itse sopivimmaksi katsomansa työtavat sprintin tavoitteen saavuttamiseksi. Sprintin päättyessä työn lopputuloksesta esitetään demonstraatio tuotteen omistajalle. Tyypillinen koko Scrum-tiimille on 5-9 henkeä. (Hybridimenetelmä, 2009.)

Tiimiin jäseniksi kuuluvat kaikki projektissa toteuttamassa olevat toimijat. Tiimin sisällä ei erikseen nimitetä jäseniä titteleittäin vaan yksinkertaisena perusperiaatteena on, että tiimiin otetaan mukaan vain henkilöitä, joilla on projektin tarpeita vastaava osaaminen. Tiimi rakentaa yhteistyöllä asiakkaan tilaaman lopputuotteen. Scrum-menetelmän roolituksella pyritään korostamaan, että kukin tiimin jäsen on projektin kannalta saman arvoinen. Olennaista myöskin on, että tiimi vastaa kokonaisena yhteisönä tuotteen eri osa-alueista, eikä vastuuseen ikinä aseteta yksittäistä tiimin jäsentä. (Poimala ym. 2011b.)

Scrum-mestari jakaa yhdessä tiimin kanssa vastuun töiden edistymisestä, mutta mestari ei itse osallistu tiimin käytännön toteutuksiin. Mestarin tehtävänä on pitää huolto siitä, että tiimiläiset pysyvät jatkuvasti toimivina ja tuottavina sekä seuraavat toimissaan Scrum:n arvoja ja käytäntöjä. Tiimiläiset kertovat päiväpalavereissa ongelmista, jotka haittaavat työn etenemistä ja Scrum-mestarin tehtävänä on ratkoa ja poistaa nämä ongelmat. Mestari ei ole projektipäällikkö, vaan Scrum-tiimin ohjaaja, yhteyshenkilö ja Scrum-prosessin johtaja. Mestari vastaa päivittäisten Scrum-palaverien järjestämisestä. Käytännössä mestarina toimii usein jonkin tiimin esimies. (Hybridimenetelmä, 2009.)

Tuotteen omistaja edustaa osakkaita ja muita sidosryhmiä, valvoo heidän etujaan, päättää tuotteen ominaisuuksista ja niihin vaikuttavista seikoista sekä vastaa tuotteen kannattavuudesta. Tuotteen omistajan tehtävänä on listata vaaditut ominaisuudet ja priorisoida ne, eli laatia tuotteen työlista.

Tuotteen omistaja päivittää työlistaa haluttujen ominaisuuksien sekä priorisoinnin osalta niin, että jokaisen sprintin alussa on käytettävissä ajan tasalla oleva työlista, jolta tehtäviä sitten kerätään toteutettavaksi. Omistaja päättää eri versioiden julkaisupäivämääristä ja sisällöstä sekä vastaa sprinttien suunnittelukokouksien järjestämisestä. Kokouksissa Scrum-tiimi päättää yhdessä, mitkä kaikki työlistalla olevat työt se pystyy seuraavan sprintin kuluessa toteuttamaan ja laatii sen perusteella sprintin työlistan. Sprintin suunnittelun lisäksi tuotteen omistaja osallistuu myös sprintin jälkitarkasteluun. Omistajan tulee tämän lisäksi olla aina Scrum-tiimin tavoitettavissa mahdollisten lisäkysymysten varalta. (Hybridimenetelmä, 2009.)

Tuotekehitysprojekteissa tuotteen omistaja on normaaleissa projektiolosuhteissa yleisesti tuotepäällikkö, asiakasprojekteissa se voi olla asiakkaan oma edustaja tai toimittajan edustama tekninen projektipäällikkö. Scrum-menetelmä on varsinkin asiakkaan näkökulmasta erittäin vaativa, koska Scrum:n vaatimuksena on, että projektissa on asiakkaan puolelta vain yksi edustaja. Tämä merkitsee periaatteessa sitä, että asiakkaan puolelta edustamaan tuleva projektin omistaja ja projektipäällikkö pitää olla yksi ja sama henkilö. Jos näin ei voida asiakkaan puolesta tehdä, niin ainakin asiakkaan projektipäällikkönä toimivalle henkilölle tulisi voida antaa selvät valtuudet tehdä päätöksiä. (Poimala ym. 2011b.)

4.3.2 Scrum-projektin aktiviteetit

Ennen kuin projektia voidaan aloittaa, luodaan korkean tason visio projektiin kohdistuvista odotuksista ja lopputuloksista. Tiimi, Scrum-mestari ja tuotteen omistaja suunnittelevat projektin suuret linjat tekemällä julkaisusuunnitelman. Sprinttien lukumäärä ja kesto, arvio kunkin julkaisun tuomasta lisäarvosta, tarvittavien henkilöiden tai tiimien lukumäärä, julkaistavien versioiden lukumäärä ja julkaisupäivät suunnitellaan ja sovitaan. Tuotteen omistaja priorisoi työlistan, jonka perusteella kunkin sprintin suunnittelu aloitetaan. Scrum-projektiin kuuluvia aktiviteetteja ovat sprintin suunnittelukokous, päiväpalaveri, sprintti, sprintin katselmointi ja sprintin jälkitarkastelu. (Hybridimenetelmä, 2009.)

Korkean tason vision hahmottelun jälkeen voidaan muodostaa alustava lista tuotteeseen halutuista ominaisuuksista. Tätä kutsutaan siis työlistaksi. Tuotteen omistajan on kuitenkin pistettävä työlistalla oleva ominaisuuslista tärkeysjärjestykseen vielä ennen toteutuksen aloittamista. (Poimala ym. 2011b.)

Sprintin suunnittelukokousta varten varataan yleensä yksi työpäivä ja siihen osallistuvat tiimin kaikki jäsenet, Scrum-mestari sekä tuotteen omistaja. Kokouksen ensimmäisen puoliskon aikana tiimi analysoi tuotteen omistajan priorisoiman työlistan ja tuotteen omistaja kuvailee tärkeimmiksi asettamansa ominaisuudet. Jälkimmäinen puolisko käytetään sprintin suunnitteluun. Tiimi valitsee tuotteen työlistalta vaatimuksia, jotka katsoo pystyvänsä yhden sprintin aikana toteuttamaan ja ne siirretään sprintin työlistaan. Kokoukseen osallistujat määrittelevät yhdessä sprintin päämäärän ja laativat lyhyen kuvauksen siitä, mitä sprintin aikana pyritään saavut-

tamaan. Tämän perusteella sprintin katselmoinnissa arvioidaan jälkeenpäin sen onnistumista. (Hybridimenetelmä, 2009.)

Päiväpalaverit järjestää Scrum-mestari ja koko tiimin on oltava niissä läsnä. Muut projektiin tai sen sidosryhmiin kuuluvat henkilöt voivat osallistua palaveriin vain kuuntelijana. Palaveri alkaa aina sovittuna aikana ja kestää 15 minuuttia. Kukin tiimiläinen vastaa kolmeen kysymykseen: Mitä olet tehnyt eilen? Mitä aiot tehdä tänään? Onko esteitä, jotka vaikeuttavat tavoitteeseesi pääsyä? Muuta keskustelua ei käydä. Scrum-mestari kirjaa ylös saavutukset, tavoitteet ja ongelmat. Sprintin työlista ja arviot jäljellä olevasta työstä päivitetään ja pidetään kaikkien sidosryhmien nähtävillä. Kuka tahansa tiimistä voi tarvittaessa järjestää erillisen kokouksen päiväpalaverin jälkeen. (Hybridimenetelmä, 2009.)

Sprintin eli työvaiheen aikana tiimi toteuttaa valitsemansa toiminnallisuudet. Tiimi organisoi itsensä ja työnsä ja voi käyttää tarvittaessa myös ulkopuolista apua. Sprintin aikana sen työlistaa ei saa muuttaa. Jos aika käy niukaksi tai sitä tuntuu jäävän yli, voidaan tuotteen omistajan kanssa yrittää neuvotella joidenkin ominaisuuksien siirtämisestä seuraaviin sprintteihin tai joidenkin muiden ominaisuuksien toteuttamisesta meneillään olevan sprintin aikana. Silti jokaisen sprintin suunnittelu noudattaa aina tavanomaista menettelyä, kun päätetään mitä ominaisuuksia sprinttiin otetaan mukaan. (Hybridimenetelmä, 2009.)

Jokainen sprintti päättyy sprintin katselmointiin, jolloin tiimi esittelee tuotteen omistajalle, asiakkaalle ja muille sidosryhmille sprintin aikana valmistuneet toiminnallisuudet. Tavallisin tapa on demonstraation tekeminen. Katselmoinnin aluksi esitellään tuotteen työlista sekä sprintin tavoitteet. Suurimman osan katselmukseen käytetystä ajasta vie uusien toiminnallisuuksien esittely ja sidosryhmien kysymyksiin vastaaminen. Kukin katselmoija esittää oman mielipiteensä sprintin onnistumisesta. Tuotteen omistaja neuvottelee tiimiläisten ja sidosryhmien kanssa tuotteen työlistaan tehtävistä muutoksista. Kaikki kertovat omat näkemyksensä tuotteeseen tarvittavista muutoksista sekä niiden tärkeysjärjestyksestä. (Hybridimenetelmä, 2009.)

Katselmoinnin jälkeen tiimi kokoontuu sprintin jälkitarkastelua varten. Siihen osallistuvat kaikki tiimin jäsenet, Scrum-mestari ja halutessaan myös tuotteen omistaja. Jälkitarkastelun aikana kukin osallistuja kertoo, mikä hänen mielestään edellisessä sprintissä meni hyvin, mikä asia toimi ja mikä ei, sekä esittää parannusehdotuksia seuraavaa sprinttiä varten. Kukin tiimin jäsen voi esimerkiksi kertoa, mitä hän haluaisi alkaa tehdä, minkä tekemisen hän haluaisi lopettaa ja minkä tekemistä jatkaa. Scrum-mestari kirjaa ylös osallistujien havainnot. Selkeät muutosehdotukset priorisoidaan ja kirjataan seuraavan sprintin työlistaan. Jälkitarkastelun avulla pyritään prosessin jatkuvaan parantamiseen. (Hybridimenetelmä, 2009.)

4.3.3 Scrum-projektin työlistat

Scrum-projektissa käytetään tuotteen työlistaa, toteutusvaiheen työlistaa sekä joskus myös julkaisun työlistaa. Tuotteen työlista sisältää kaikki ne vaatimukset, jotka tuotteelle on suunniteltu toteutettavaksi. Kun tuotteen omistaja on priorisoinut tuotteen työlistan, siitä siirretään julkaisun työlistaan seuraavaa julkaisua varten valitut ominaisuudet. Julkaisun työlistalta valitaan toteutusvaiheen työlistalle seuraavassa sprintissä toteutettavat ominaisuudet. Mikäli julkaisun työlistaa ei käytetä, seuraavassa sprintissä toteutettavat ominaisuudet valitaan toteutusvaiheen työlistalle suoraan tuotteen työlistalta. Toteutusvaiheen työlistaa päivitetään joka päivä suoritettujen tehtävien osalta. Lisäksi tiimiläisten on pidettävä ajan tasalla arviota jäljellä olevista tehtävistä ja niihin käytettävästä ajasta. Arvioista tehdään yhteenveto ja graafinen esitys, jota pidetään päivitetyn työlistan ohella kaikkien sidosryhmien nähtävillä. (Hybridimenetelmä, 2009.)

4.3.4 Kommentteja Scrum-menetelmästä

Mäntylahden mukaan ketterissä menetelmissä ja Scrum-menetelmässä on omat ongelmansa. Scrum-menetelmä soveltuu Mäntylahden kokemusten mukaan heikosti tilanteisiin, joissa projektissa paljon on osa-aikaisia työntekijöitä, projekti on maantieteellisesti hajaantunut ja projektilla on ulkoisia riippuvuuksia.

Scrum-projektissa yksittäisen tehtävän aikataulun pettäminen on paljon suurempi ongelma kuin perinteisessä projektissa. Normaalissa kehitysprojektissa projektipäällikkö voi pitää ns. turvamarginaalipäiviä. Scrum:a käytettäessä ylimääräisten päivien sisällyttäminen projektiin ei ole mahdollista. Scrum-menetelmää käytettäessä varmuusmarginaalit ovat kaikille julkisia ja ne tulee useimmin myös hyödynnettyä jo työlistan suorittamista suunniteltaessa.

4.4 XP-menetelmän vaiheet

Extreme Programming tunnetaan myös lyhyemmin nimellä XP. Kent Beck esitteli menetelmän perusperiaatteet kirjassaan ”Extreme Programming Explained: Embracing Change” vuonna 1999 (Beck 1999). Vuonna 2004 Beck on vielä julkistanut 2. painoksen kyseisestä kirjasta.

XP-menetelmä on nimensä mukaisesti hyvin ohjelmointikeskeinen menetelmä. XP-menetelmä eroaa selvästi muista ketteristä menetelmistä sillä, että se tarjoaa tarkkaan valitun joukon eri käytäntöjä, joita XP-menetelmää noudatettaessa tulee toteuttaa yhdessä. XP-menetelmän mahdollisimman tehokas hyödyntäminen edellyttää, että kaikkia käytäntöjä todellakin noudatetaan yhdessä, sillä käytäntöjen kokoonpano on rakennettu tasapainottamaan toisiaan. (Poimala ym. 2009b.)

Aivan pelkkä käytäntökokoelma ei XP-menetelmäkään ole, sillä mukana on koko joukko periaatteita ja myös menetelmän omat arvot. Vaikka pää-

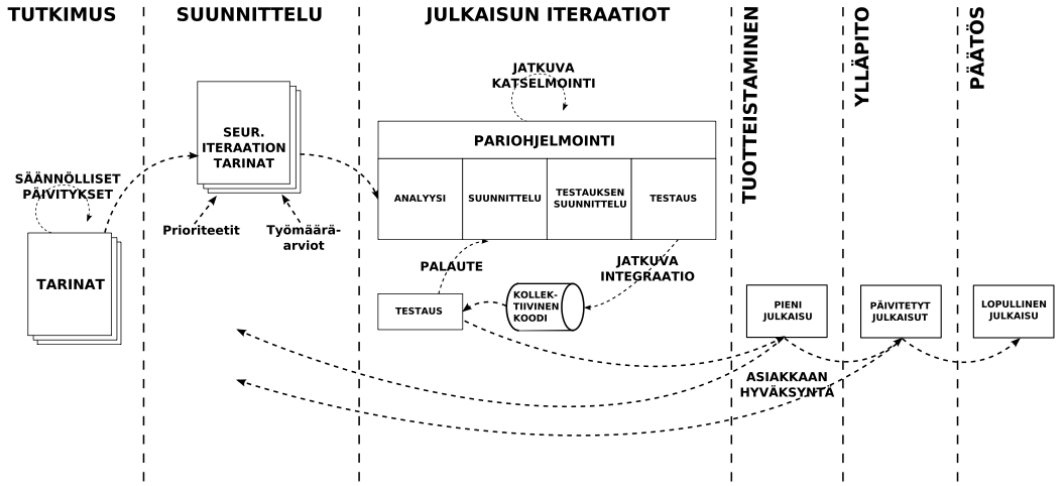
paino onkin toteutuskäytännöissä, XP:ssä on jonkin verran myös projektinhallintaan liittyviä käytäntöjä, kuten suunnittelupeli. Itse asiassa Kent Beckin suurin huoli onkin, että XP:n toteutuksissa keskitytään liikaa insinöörikäytäntöihin. Menetelmän todellinen luonne tulee kuitenkin esiin vain, jos sen arvoja ja periaatteita noudatetaan kokonaisuutena. XP:n monet käytännöt ohjaavat toteutusta teknisesti parempaan suuntaan, mutta Beckin mukaan olisi tarpeen keskittyä vastuulliseen kehitykseen. (Poimala ym. 2009b.)

XP-menetelmä on kehitetty ratkaisemaan perinteisen suunnitelmaohjautuvan ohjelmistotuotannon prosessimallien, kuten vesiputousmallin, mukanaan tuomia ongelmia ja haasteita (Beck, 1999). Tällaisia haasteita ovat esimerkiksi vaatimuksiin kohdistuvien muutoksien suuri vaikutus ohjelmistoprojektien työmääriin ja prosessin jäykkyys.

Abrahamssonin & ym. (2002) mukaan Beck (1999) esittää XP:n perusajatukseksi sen, ettei ole olemassa prosessia, joka sopii sellaisenaan kaikkiin projekteihin. Sen sijaan on olemassa yksittäisiä käytäntöjä, jotka voidaan räätälöidä jokaiseen projektiin. XP on joukko tällaisia parhaita käytäntöjä yhdessä sovellettuna ja se on tarkoitettu pienille ja keskisuurille projektitiimeille.

4.4.1 XP:n prosessin vaiheiden kuvaus

XP:n elinkaarimalli muodostuu kuudesta eri vaiheesta: tutkimus, suunnittelu, julkaisun iteraatiot, tuotteistaminen, ylläpito sekä päätös.



Kuva 4. XP –prosessin elinkaarimalli (Abrahamsson ym. 2002.)

Kuvassa 4 on esitetty XP:n ohjelmistoprosessin elinkaarimalli. Vaiheet suoritetaan vasemmalta oikealle ja tietyissä kohdissa palataan takaisin aiempaan vaiheeseen.

Abrahamssonin ym. mukaan projekti aloitetaan tutkimusvaiheesta, jossa asiakkaat kirjoittavat järjestelmän ensimmäiselle versiolle asettamansa

ominaisuus- ja muut vaatimukset yksittäisiksi ja mahdollisimman yksinkertaisiksi käyttötarinoiksi. Kukin käyttötarina kuvaa tietyn yksittäisen järjestelmään toteutettavan ominaisuuden. Samalla projektiryhmä tutustuu projektissa käytettäviin välineisiin, teknologioihin ja käytäntöihin. Käytännössä tutkimusvaihe korvaa perinteisen vaatimusmäärittelyvaiheen. Tutkimusvaihe voi kestää muutamasta viikosta muutamiin kuukausiin.

Suunnitteluvaiheessa priorisoidaan käyttötarinoiden toteutusjärjestys ja sovitaan ensimmäiseen pienempään julkaisuun sisältyvät ominaisuudet. Kun ominaisuudet ensimmäiseen julkaisuun on valittu, ohjelmoijat antavat asiakkaalle työmääräarvion kunkin ominaisuuden toteuttamisesta. Asiakas päättää ensimmäiseen versioon toteutettavista ominaisuuksista ja sen jälkeen aikataulu lukitaan. Työmääräarvioita helpottaakseen ohjelmoijat voivat toteuttaa nopean, muutamassa tunnissa tehdyn prototyypin, jolla jonkin tietyn toteutustavan tai -teknologian käyttöä voidaan kokeilla ennen toteutuksen aloittamista. Ensimmäinen julkaisu pyritään tuottamaan yleensä alle kahdessa kuukaudessa. Suunnitteluvaihe kestää useimmiten muutaman päivän ja vastaa käytännössä perinteistä projektisuunnittelua. (Abrahamsson ym., 2002.)

Yhdessä tutkimus- ja suunnitteluvaihe muodostavat yhden XP:n ydinkäytännöistä eli suunnittelupelin, johon osallistuvat asiakkaat ja ohjelmoijat. Suunnittelupelin tavoitteena on saada sekä asiakas että ohjelmoijat ymmärtämään, mitä ollaan tekemässä, ja valita ensimmäiseen julkaisuun suurimman vaikutuksen tuottavat ominaisuudet (Beck, 1999.)

Abrahamsson:n ym. mukaan julkaisun iteraatiot sisältävät useita iteraatioita järjestelmästä ennen ensimmäistä julkaisua. Suunnitteluvaiheen aikataulu pilkotaan eri iteraatioille ja näiden toteutus kestää yleensä yhdestä viikosta neljään viikkoon. Ensimmäinen iteraatio luo kokonaisen järjestelmän arkkitehtuureineen. Tämä mahdollistuu kun valitaan “tarinat”, jotka pakottavat rakentamaan toimivan järjestelmän. Jokaisen iteraation jälkeen tehdään toiminnallisia testejä. Viimeisen iteraation jälkeen järjestelmä on valmis siirtymään tuotteistusvaiheeseen.

Tuotteistamisvaiheessa ohjelmistolle tehdään vielä lisää toiminnallisuuteen ja suorituskykyyn liittyviä testejä. Tässäkin vaiheessa voi tulla vielä uusia muutostarpeita, mutta niiden mukaan ottamista valmistuvaan julkaisuun on harkittava, ja niistä on tehtävä päätös. Tässä vaiheessa suoritetaan tarvittaessa uusia iteraatio-kierroksia, kunnes julkaisu on valmis julkaistavaksi. Toteuttamatta jätetyt muutostarpeet listataan ylläpitovaiheessa mahdollisesti toteutettaviksi muutoksiksi

Kun järjestelmän kehitysiteraation tuloksena julkaistaan uusi versio, aloitetaan valmistuneen version osalta ylläpitovaihe. Tässä vaiheessa ylläpidetään julkaistua versiota erilaisten ylläpitotoimintojen, kuten käyttäjätuen, virheiden korjaamisen ja teknisen tuen kautta. Julkaistuun järjestelmään voidaan kehittää lisää ominaisuuksia uusissa iteraatioissa. Tällöin niistä tehdään uusia käyttäjätarinoita, jotka lisätään aiempien käyttötarinoiden joukkoon odottamaan toteuttamistaan.

Kun kaikki asiakkaan projektin aikana antamat käyttötarinat on toteutettu järjestelmään ominaisuuksiksi, eikä uusia käyttötarinoita enää ole tulossa, järjestelmän kehittämisprosessi siirtyy päätösvaiheeseen. Tällöin järjestelmä täyttää kaikki sille asetetut vaatimukset, ja siitä on tehty lopullinen julkaisu. Tämän jälkeen järjestelmän ohjelmakoodiin, dokumentaatioon tai tekniseen arkkitehtuuriin ei enää tehdä muutoksia, ja järjestelmäversio jäädytetään. Päätösvaiheeseen voidaan päätyä myös siinä tapauksessa, jos järjestelmä ei täytä sille asetettuja vaatimuksia, tai sen kehittäminen on käynyt tarpeettomaksi tai liian kalliiksi. (Abrahamsson ym., 2002.)

4.4.2 XP:n käytännöt

Beck:n mukaan XP:n käytännöt jaetaan ensisijaisiin ja seuraajakäytäntöihin. Käytännöt tukevat ja toteuttavat periaatteiden välityksellä XP:lle määritetyt arvot.

Työntekijän hyvinvointia ja humaaniusarvoa tukevia ensisijaisia käytäntöjä ovat erityisesti kanssakäyminen yhdessä koko tiimin kanssa, informatiivinen työympäristö, innostava työ ja löyhän ajan järjestäminen. Järjestämällä tiimiläisille yhteiset informaationvälitystä tukevat tilat ja mahdollisuus tehokkaaseen kanssakäymiseen saadaan aikaan tiimihenkeä tukeva ilmapiiri, jossa tiimiläiset tuntevat kuuluvansa yhteen ja pyrkivät auttamaan toisiaan niin työssä ja kasvamisessa kuin oppimisessakin. Työtaakan kohtuullisena pitämistä voidaan auttaa löyhällä ajalla, jota voidaan saada esimerkiksi määrittelemällä joka iteraatiolle muutama vähempiarvoinen tehtävä, joka voidaan tarpeen tullen pudottaa pois.

Muita ensisijaisia käytäntöjä ovat pariohjelmointi ja tarinat, viikoittaiset syklit, neljännesvuosittaiset syklit, kymmenen minuutin järjestelmätestaus, jatkuva integraatio, testivetoinen kehitys ja inkrementaalinen suunnittelu.

Seuraajakäytännöt ovat käytäntöjä, joihin siirtyminen on mielekästä vasta kun ensisijaiset käytännöt on pääpiirteittäin sisäistetty. Seuraajakäytäntöihin kuuluu olennaisesti inkrementaalinen järjestäytyminen uuden ajattelumallin mukaisesti eli siirrytään askeleittain vanhoista perinnejärjestelmistä pois. Olennaista on myös, että erikseen järjestetään tiimit, jotka sitten aina pidetään yhdessä. Tiimissä analysoidaan yhdessä kaikki ydinongelmat.

Seuraajakäytäntöihin kuuluvat myös kaikille yhteinen ohjelmistokoodi, jota yhdessä testauksen kanssa pidetään lähtökohtana kaikille muille dokumenteille. Koodaus perustuu yksittäisen koodijuuren hallintaan eli ei pidetä useita päällekkäisiä koodipolkuja samassa projektissa. Tähän perustuen tehdään ohjelmistosta päivittäin uusi ohjelmistoversion julkaisu, koska ohjelmoijan koneella olevan koodin ja julkaisun välinen ero on aina riski.

Lisäksi seuraajakäytännöt perustuvat lyhyihin sopimuskausiin eli riskien minimoimiseksi sovitaan mieluummin monta pientä kiinteää sopimusta

kuin yksi pitkä. Oletuksena käytössä on myös käytönaikainen laskutus eli kerätään maksu järjestelmän käytön määrästä. (Beck, 1999.)

4.4.3 XP:n neljä arvoa

Kehitysmallilla on neljä määriteltyä arvoa, jotka palvelevat sekä ihmisten että yritysten tarpeita: kommunikaatio, yksinkertaisuus, palaute ja rohkeus. Beckin mielestä toimimaton kommunikaatio ei ole sattumaa, vaan ihmiset aiheuttavat sen itse omalla toiminnallaan. Kommunikaatio pyritäänkin ylläpitämään sujuvana hyödyntämällä erilaisia käytäntöjä, jotka vaativat toimivaa kommunikaatiota. Tällaisia ovat esimerkiksi pariohjelmointi ja tehtävien arviointi. (Beck, 1999.)

Beck:n mukaan on tärkeämpää tehdä asia ensin yksinkertaisesti ja vasta myöhemmin tarvittaessa muuttaa sitä, vaikka muutos maksaisikin enemmän. Tällöin vältytään tuottamasta jotain monimutkaista, jota ei ehkä koskaan tulla tarvitsemaan. Yksinkertaisuus liittyy kommunikaatioon, koska kommunikaatiolla asioista saadaan yksinkertaisempia. Mitä yksinkertaisempi järjestelmä on, sitä vähemmän tarvitaan kommunikaatiota ihmisten välillä.

Järjestelmän nykytilasta saatu palaute on arvokasta tietoa. Ohjelmistokehityksessä optimismi on vaarallinen asenne, joka voi estää löytämästä virheitä ohjelmakoodista ja saamasta tietoa asiakkaalta. Konkreettinen palaute muun muassa testauksen muodossa estää olettamuksien aiheuttamia ongelmia. Mitä enemmän saadaan palautetta, sitä parempaa on kommunikaatio. Tällöin myös järjestelmästä tulee yksinkertaisempi.

Rohkeus auttaa ohjelmoijaa kokeilemaan uskomattomalta vaikuttavia ratkaisuja, jotka toisinaan tuottavat erinomaisia tuloksia. Ilman kommunikaatiota, yksinkertaisuutta ja palautetta, rohkeus on kuitenkin vaarallista ja siitä voi koitua ylimääräistä haittaa. (Beck, 1999.)

4.4.4 Kommentteja XP-menetelmästä

Lemmetin mukaan XP-menetelmän lähestymistavan vahvuus perustuu vahvasti sen ominaisuuteen sopeutua nopeasti erilaisiin muutoksiin. Näin sovellus säilyy helposti muutettavana koko kehitysprojektin ajan, eikä kehitystiimin tarvitse käytettävän menetelmän takia käyttää ylimääräistä aikaa dokumentaation ajan tasalla pitämiseen.

Lemmetin tulkinnan perusteella XP-menetelmää käyttämällä tuotetut sovellukset ovat helpommin ylläpidettäviä ja muokattavia ja niissä on myös parempi kokonaisrakenne verrattuna pitkälti etukäteen suunniteltuihin sovelluksiin. XP-menetelmä tavoittelee lähtökohtaisesti juuri näitä ominaisuuksia sovelluksen kehitysprojekteissa. XP-menetelmän heikkona kohtana Lemmetti toteaa, että laatuvaatimuksia ei XP-menetelmää käyttämällä oteta välttämättä tarpeeksi tarkasti huomioon.

Lemmetin mukaan XP-menetelmä soveltuu parhaiten sellaisenaan pieniin ja keskisuuriin projekteihin, joiden vaatimukset ovat itsessään jollain tasolla epävakaita ja joilla ei ole tarkkoja laatuvaatimuksia. Lisäksi Lemmetti mainitsee, että XP-menetelmä soveltuu paremmin tapauksiin, joissa sovelluksen toteuttajat eivät tiedä hyvin jo ennalta käsin käsiteltävää ongelma-aluetta, koska silloin etukäteissuunnittelu on hankalaa ja toiselta kantilta katsottuna jatkuva lähestymistapa lisää tietämystä projektin kokonaiskuvasta jatkuvan palautteen kautta. Lisäksi monimutkaisen suunnitelman sisäistäminen ja toteuttaminen taitaa olla sen toteuttajille selvästi hankalampaa kuin mahdollisimman yksinkertaisesta kokonaisratkaisusta aloittaminen ja vaiheittainen kehittäminen.

Lemmetin ajatuksien mukaan eniten epäilyksiä XP-menetelmässä herättää sen lähestymistavan sopeutuminen suuriin kehitysprojekteihin, etenkin jos sovellukselle asetetaan liian tarkkoja laatuvaatimuksia. Lisäksi Lemmetti toteaa kommenteissaan, että dokumenttien tekemättömyys on toiselta kantilta katsottuna etu ja jopa edellytys projektin sujuvuudelle. XP-menetelmä ei hänen mukaansa näin toteutettuna tule todennäköisesti toimimaan, jos projektin koko suurenee liikaa. Jos kehitettävä sovellusratkaisu on tarpeeksi laajamittainen, tulee sen hahmottamisesta mahdotonta ilman yhtään tallennettua dokumenttia. (Lemmetti 2007.)

4.5 SAP:n ASAP-menetelmän vaiheet

ASAP:n (Accelerated SAP) tarkoituksena on suunnitella SAP:n toiminnanohjausjärjestelmien käyttöönottoja mahdollisimman tehokkaalla tavalla. Tavoitteena on tehokas resurssien käyttö; ajan, käytössä olevien ihmisten ja laadun suhteen. ASAP perustuu 6-vaiheiseen käyttöönottoohjeistukseen. (Genieholdings 1999.)

Kuva 5. SAP:n ASAP-menetelmän vaiheet (SAP AG 2010).

Ensimmäisessä Project Preparation- eli projektin valmisteluvaiheessa projektitiimi määrittelee projektin tavoitteet, projektin karkean tason sisällön ja projektisuunnitelman. Lisäksi toimeksiantajalta varmistetaan tarvittava rahoitus. Samalla sovitaan projektissa käytössä olevista standardeista ja perustetaan projektiryhmä. Ensimmäisessä vaiheessa määritellään toimintasuunnitelma, joka hyväksytään erikseen toimeksiantajan toimesta. Kaikille projektiryhmän jäsenille sovitaan roolit ja vastuualueet, jotka myös

dokumentoidaan. Projektin kohteet tarkistetaan ja kaikki projektin aktiviteetit dokumentoidaan projektitaulukkoon (SAP AG 2010).

ASAP:n toinen vaihe on ns. Business Blueprint-vaihe. Tämän vaiheen aikana dokumentoidaan kaikki tekniset suunnitelmat business blueprint-dokumenttiin. SAP-toiminnanohjausjärjestelmän toteuttamiseksi järjestetään erillisiä kokouksia, joissa SAP:n kokoama asiantuntijoiden konsulttiryhmä kokoaa yhteen kaikki tarvittavat prosessit ja sovellukseen tarvittavat ratkaisut. Samalla käydään läpi kaikkien SAP:n standardiratkaisut ja toimeksiantajan antamat kuvaukset yhdessä SAP-asiantuntijoiden kanssa, jotta päästäisiin mahdollisimman hyvään kokonaisratkaisuun. Kaikki toiminnalliset ja tekniset vaatimukset sekä havaitut puutteet ja eroavaisuudet dokumentoidaan SAP:n ratkaisunhallintajärjestelmään (SAP AG 2010).

Kolmannessa Realization- eli toteutusvaiheessa SAP:n järjestelmää muokataan ja testataan monissa eri sykleissä. Aluksi konfiguroidaan vain SAP:n perustoiminnot, jotka vastaavat ainoastaan liiketoiminnan ydinprosesseihin. Se testataan ja lopuksi vahvistetaan liiketoiminnan toimesta. Sen jälkeen järjestelmää kehitetään kierros kierrokselta niin kauan kunnes kokonaisratkaisu on valmis. Kokonaisratkaisun valmistumisen jälkeen tehdään vielä useita testikierroksia. Kaikki tehdyt konfiguraatiot dokumentoidaan SAP:n ratkaisunhallintajärjestelmään. Vanhasta järjestelmästä tehtävät konversio-ohjelmat testataan. Tämän jälkeen toteutetaan tuotantojärjestelmä tehdyn konfiguraation mukaan (SAP AG 2010).

Neljännessä Final Preparation- eli viimeistelyvaiheessa kaikki toimintaan liittyvien järjestelmien integraatioiden toiminta SAP-järjestelmän kanssa testataan. Toimeksiantajalta vaaditaan aina hyväksyntä kaikkien integraatioiden toimivuudesta. Tässä vaiheessa kaikkien integraatioiden pitäisi teknisesti jo toimia. Tarkemmat toiminnanohjausjärjestelmän muutos- ja käyttöönottosuunnitelmat tehdään valmiiksi. Samalla myös muodostetaan asiakkaan tarpeita palveleva tukiorganisaatio. Tässä vaiheessa järjestelmään ladataan kaikki viimeisimmät liiketoiminnan operatiivinen data asiakastietoineen. Tämän vaiheen lopussa tuotantojärjestelmä otetaan liiketoiminnan operatiiviseen käyttöön uudessa ympäristössä (SAP AG 2010).

Viidennen Go-Live Support–vaiheen tarkoitus on siirtyä lopullisesti vanhan järjestelmän käytöstä SAP-toiminnanohjausjärjestelmän käyttöön. Tässä vaiheessa tulee olla jo loppukäyttäjiä tukeva organisaatio kaikkien loppukäyttäjien saatavilla (SAP AG 2010).

Kuudennen Run-vaiheen tarkoituksena on taata operatiivisen toiminnallisuuden toimiminen. IT-ratkaisujen tulee olla sillä tasolla, että yrityksen toiminta on turvattu ja järjestelmä on jatkuvasti yrityksen käytettävissä. On suositeltavaa aloittaa tämä vaihe heti Go-Live-vaiheen yhteydessä. Tässä vaiheessa on hyvä tutkia vielä mahdolliset poikkeavuudet SAP:n standardiratkaisusta ja niiden perusteella tulee järjestelmään tehdä parannuksia asiakkaan tarpeiden mukaan. Kaikki järjestelmään liittyvät lopulli-

set dokumentit tulee vielä tallentaa SAP:n ratkaisunhallintajärjestelmään (SAP AG 2010).

4.6 Yhteenveto tutkimuksessa kuvatuista malleista

Yhteenvedossa käsitellään vesiputous, Scrum- ja XP-menetelmän eroja ja yhteneväisyyksiä keskenään eikä SAP:n lanseeraamaa ASAP-menetelmää ole otettu mukaan vertailuun. YIS:n omiin käyttötarkoituksiin muokattua ASAP-menetelmää verrataan tarkemmin edellä mainittuihin menetelmiin tutkimuksen luvussa 8.

Tutkimuksen kolmesta eri prosessimallista on helposti havaittavissa suurilinjaiset eroavaisuudet. Lineaarinen vesiputousmalli erottuu selvästi muista tutkimuksessa esiintyvistä ketteristä malleista selvästi kankeampana ja huonommin yllättäville muutoksille soveltuvana prosessimallina. Se profiloituu selvästi vanhan ajan malliksi, jota kuitenkin käytetään edelleen menestyksellisesti varsinkin suuremman luokan käyttöönottoprojekteissa. Se vaatii kuitenkin tarkan ja hyvinkin pitkäkestoisen suunnitteluajan onnistuakseen. Mallista on aikojen saatossa tietysti tehty erilaisia muunnelmia, joita sitten pystytään soveltamaan erilaisissa käyttöönottoprojekteissa. Nykyaikaisetkin ketterät menetelmät pohjautuvat ainakin osittain vesiputousmallissa käytettyyn vaiheistukseen.

Scrum-malli niin kuin monet muutkin ketterät menetelmät pohjautuvat vaiheistukseltaan hyvinkin vahvasti vesiputousmalliin, mutta taasen eroavat selvästi vesiputousmallista sen perusperiaatteiltaan. Scrum- ja XP – mallin käyttäminen käyttöönottoprosesseissa ei edellytä, että varsinainen projekti olisi alkutekijöissään kovinkaan pitkälle valmiiksi suunniteltu kokonaisuus, kun taas vesiputousmallin käyttö edellyttää projektin kauaskantoisia suunnitelmia. Scrum-mallissa pystytään helpommin tekemään projektin aikaisia muutoksia käyttöönottoprosessin sisällä ja sen takia Scrum-mallin käyttäminen pienemmissä projekteissa onkin luontevampaa.

XP-menetelmää verrattaessa Scrum-menetelmään voidaan heti havaita tiettyjä yhteneväisyyksiä. Kummatkin ketterinä malleina antavat mahdollisuuden lyhytjänteiseen työskentelyyn ottamalla huomioon projektin aikana mahdollisesti tulevat muutokset. Muutoksiin pystytään reagoimaan prosessin sisällä nopeasti. Vaiheistus näiden kahden metodin välillä on lähes tulkoon samanlainen. Mallien eroavaisuutena on, että XP-menetelmässä pyritään valmistamaan lopputuote ehkä vieläkin pienemmissä osissa kuin Scrum-menetelmällä. Samalla korostuu XP-menetelmän ohjelmointikeskeisyys, jossa ohjelmoimalla pieniä osia kerrallaan saadaan vähitellen rakennettua kokonaista toimivaa sovellusta. Tässä korostuukin Scrum:n ja XP:n suurin eroavaisuus. Scrum-menetelmää käytettäessä tehdään iteroivassa osuudessa suurempia kokonaisuuksia kerrallaan. Näiden tekeminen kestää yleensä 1-4 viikkoa, kun taas XP-menetelmä perustuu jatkuvaan ohjelmointiin ja testaukseen, jossa katselmointia ja sovelluksen testausta tehdään kutakuinkin reaaliajassa päivittäin kuitenkin niin, että suuremmat kokonaisuudet tehdään myös 1-4 viikon jaksoissa.

5 TUTKIMUKSEN TOIMINTAYMPÄRISTÖ

Tässä luvussa kerrotaan tarkemmin tutkimuksessa käytössä olleesta toimintaympäristöstä. Samalla tarkennetaan käsitteitä case-yritys (YIS) ja case-sovellus (MDM SC).

5.1 YIT

YIT tarjoaa palveluja rakentamisen ja kiinteistötekniikan eri osa-alueilla. YIT rakentaa asuntoja, toimitiloja, kokonaisia alueita ja tarvittavia infrastruktuureja sekä ylläpitää kiinteistöjä erilaisilla kiinteistöteknisillä ratkaisuilla sekä niiden huollolla ja kunnossapidolla.

YIT on suurin kiinteistöteknisten palvelujen tarjoaja Pohjoismaissa ja yksi johtavista palvelutarjoajista myös Keski-Euroopassa. YIT on Suomessa suurin omaperustaisten asuntojen rakentaja ja suurin yksityinen teiden kunnossapitäjä. Venäjällä YIT on yksi merkittävimmistä ulkomaisista asuntorakentajista.

YIT on perustettu vuonna 1912, jolloin Yleinen insinööritoimisto aloitti toimintansa. Vuonna 2011 YIT:n toimialojen liikevaihto oli 4,5 miljardia euroa. YIT.llä on noin 26 000 työntekijää Suomessa, Ruotsissa, Norjassa, Tanskassa, Venäjällä, Virossa, Latviassa, Liettuassa, Saksassa, Itävallassa, Puolassa, Romaniassa, Tshekissä ja Slovakiassa. Konsernin neljä toimialaa ovat Pohjois-Euroopan kiinteistötekniset palvelut, Keski-Euroopan kiinteistötekniset palvelut, Suomen rakentamispalvelut ja Kansainväliset rakentamispalvelut. YIT:n osake noteerataan NASDAQ OMX Helsinki Oy:ssä (YIT Corporation 2011).

5.2 YIS

Tutkimuksen tilaajana toimii YIT Information Services Oy (YIS), jonka toimintaperiaatteena on palvella ja ohjata YIT:n kaikkia yhtiöitä ympäri Eurooppaa tietoteknillisissä haasteissa.

YIT Information Servises Oy.n palveluihin kuuluu mm. Helpdesk-palvelu, infra-palvelut sekä SAP-toiminnanohjausjärjestelmän tuki-, kehitys- sekä projektitoiminta. Tutkimuksen tekijä kuuluu ryhmään Group IT, joka palvelee asiakkaita SAP:iin liittyvissä projekteissa sekä kehitys- ja tukiasioissa. Tutkimuksen tekijän toimenkuva koostuu lähinnä SAP:n tuki- ja kehitystehtävistä, mutta tekijä on myös osallistunut SAP:n käyttöönottoprojektien toimintaan muutamia kertoja.

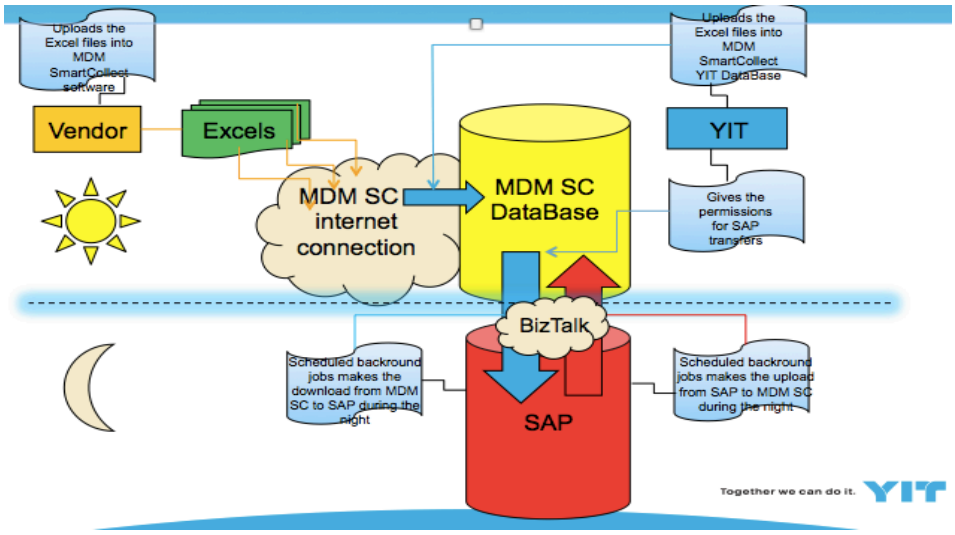
5.3 MDM SC –sovellus

MDM SC on alun perin YIS:n sisäiseen käyttöön tarkoitettu verkossa toimiva työkalu, jolla pystytään päivittämään SAP-toiminnanohjausjärjestelmän materiaalien ostohintoja.

MDM SC:n toimii käytännössä niin, että esim. hinnastopäivitysmateriaali ladataan verkon yli Excel-tiedostossa MDM SC:n tietokantaan ja sieltä siirtoluvan antamalla tarvittavat tiedot siirtyvät SAP-toiminnanohjausjärjestelmään. Siirrot kuvataan tarkemmin seuraavissa kappaleissa.

MDM SC on perustoiminnallisuudeltaan hyvin yksinkertainen. Toiminnallisuus perustuu siihen, että MDM SC:ssä on aina SAP:n ajanmukaiset tiedot. Tämä toteutetaan käyttämällä SAP:iin ohjelmoituja ajastettuja delta-ajo-ohjelmia, jotka poimivat tausta-ajoilla kaikki tarpeelliset SAP:ssa muuttuneet tiedot siirtotiedostoihin, jotka siirretään ajastetusti BizTalk –tiedostonsiirto-työkalulla MDM SC:n palvelimelle ja sitä kautta MDM SC:n tietokantatauluihin. SAP:iin rakennetut ohjelmat poimivat aina kaikki uudet ja muuttuneet tiedot kaksi päivää taaksepäin. Tällä varmistetaan, että varmasti kaikki muuttuneet tiedot siirtyvät MDM SC:n käyttöön.

MDM SC:sta SAP:iin päin tiedot siirtyvät myös BizTalk:n kautta. MDM SC:ssa annetaan kullekin SAP:iin siirrettävälle tietueelle aina erikseen siirtolupa, jonka perusteella BizTalk poimii SAP:iin siirrettävät tiedot siirtotiedostolle ja siirtää ne ajastetusti SAP:n levynkulmalle. Siirtoehdoissa on paljon muitakin ehtoja, jotka tulee täyttyä aina ennen kuin tiedot poimitaan siirrettävään siirtotiedostoon. SAP:n levynkulmalta tiedot poimitaan SAP:iin rakennettujen ohjelmien avulla ajastetusti. Poiminnan jälkeen tiedot luetaan järjestelmällisesti SAP:n tietokannan eri tauluihin.



Kuva 6. MDM SC:n toiminnallisuuden prosessikuvaus

Kyseinen MDM SC –sovellus tulee asiakkaidemme käyttöön ympäri Eurooppaa, joten sovelluksen käyttöönottoprojektissa tulee ottaa huomioon usean eri maan tarpeet ja toimintatavat.

6 TUTKIMUSMENETELMÄT

Tutkimustyön teoriaosuudessa käytettiin yleisesti saatavilla olevaa kirjallisuutta ja luotettavat internet-lähteet toimivat hyvänä pohjana tarvittavien tietojen kasaamiseksi.

Käytännönosuuden pohjana toimivat pääasiassa tutkimuksen tekijän osallistuvan havainnoinnin seurauksena tulleet kokemukset käyttöönotettavan sovelluksen kehitys- ja käyttöönottoprojektista. Lisäksi tutkimuksen empiirisessä osuudessa tehdään erillisiä haastatteluita tutkimuksen toimeksiantajan ohjaajan kanssa valittujen yrityksen työntekijöiden kanssa. Valitut henkilöt ovat olleet useammassa sovelluksen käyttöönotossa mukana ja tietävät YIS:llä yleisesti käytössä olevat menetelmät. Lisäksi haastatellaan mm. tutkimuksen tekijän esimiestä sekä käyttöönotettavan MDM SC –sovelluksen toimittajan edustajaa.

Saadun aineiston käsittelyssä tehdään vertailevaa analyysiä teoria- ja käytännönosuuden välillä. Tavoitteena on löytää teoreettisesti paras mahdollinen sovelluksen käyttöönottoprosessin malli YIS:lle. Saatujen tulosten perusteella tehdään vertailevaa tutkimusta käyttöönotettavan sovelluksen käyttöönottoprosessin ja muiden yrityksessä tehtyjen käyttöönottoprosessien välillä. Näiden saatujen tutkimustulosten perusteella tehdään johtopäätökset ja annetaan parannus- ja kehitysideat yrityksen käyttöön.

Tutkimustyyppinä on case-tutkimus. Tutkimuksen tarkoitus ei ole löytää yleismaailmallista mallia, joka toimisi kaikissa yrityksissä ja tilanteissa vaan tavoitteena on peilata teoriamalleja lähinnä yleisesti YIS:n sekä erityisesti käyttöönotettavan sovelluksen kaltaisten prosessien tarpeisiin.

6.1 Osallistuva havainnointi

Osallistuva havainnointi tutkimuksen tapauksessa tarkoittaa sitä, että osallistuvaa havainnointia pyritään käyttämään haastatteluilla kerättyjen tuloksien täydentävänä havainnointimuotona. Havainnoinnin osuus korostuu erityisesti käyttöönotettavan sovelluksen käyttöönoton analysoimisessa. Sovelluksesta tehdyt havainnot perustuvat tutkimuksen tekijän kokemuksiin kyseisen sovelluksen käyttöönottoprosessista. Tutkimuksen tekijä toimi sovelluksen käyttöönotossa yrityksen yhteyshenkilönä ja projektipäällikkönä.

Tutkimuksen tekijän osallistuvan havainnoinnin ja analysoinnin perusteella pystytään dokumentoimaan sovelluksen käyttöönottoprosessissa mukana olleet vaiheet ja menetelmät, ja näiden perusteella tehdään vertailua teoriaosuudessa käytettyihin metodeihin.

6.2 Haastattelut

Haastatteluilla pyritään saamaan mahdollisimman selkeä kokonaiskuva YIS:llä käytettävistä sovellusten käyttöönottomalleista ja kuinka niitä on toteutettu käyttöönottoprosesseissa.

Haastateltavat valittiin painottamalla MDM SC:n käyttöönoton osuutta. Haastateltavat ovat siis työskennelleet YIS:llä pääosin samankaltaisten käyttöönottoprosessien kanssa. Lisäksi haastateltavien joukossa henkilö, joka vastaa YIS:llä suurempien käyttöönottoprojektien etenemisestä. Lisäksi haastateltavaksi valittiin yksi MDM SC –sovelluksen toimittajan edustajista.

Kaikilla haastatelluilla on kokemusta usean eri sovelluksen käyttöönotosta. Haastateltavien käyttöönottamat sovellukset ovat vaihdelleet kooltaan laidasta laitaan. Projekteissa on ollut jäseniä mukana aina kahdesta henkilöstä jopa 50 henkilöön. Käyttöönotettavilla sovelluksilla on ollut loppukäyttäjiä yhdestä käyttäjästä 2000 käyttäjään. Käyttöönotettavien sovellusten projektit ovat kestäneet muutamasta päivästä 1,5 vuoteen.

Haastateltavien nimiä ei tutkimuksessa julkaista. Jotta saataisiin haastateltavista henkilöistä parempi yleiskuva, on haastateltavat henkilöt esitelty titteleittäin liitteessä *Opinnäytetyön haastattelun kysymykset ja vastaukset* (LIITE 1). Samaisesta liitteestä tulee myös tarkemmin ilmi, millaisissa käyttöönotoissa haastateltavat ovat olleet mukana. Liitteeseen kirjatut vastaukset ovat lyhennelmiä haastateltavien vastauksista. Vastauksia jouduttiin muutenkin hieman muokkaamaan yrityssalaisuuksien suojelemiseksi.

7 YIS:N KÄYTTÖÖNOTTOPROSESSIT

Tämä luku perustuu haastatteluista saatuihin tietoihin ja tutkimuksen tekijän osallistuvaan havainnointiin. MDM SC –sovelluksen käyttöönottoprosessista kertova kappale perustuu hyvin pitkälti tutkimuksen tekijän havaintoihin ja osittain haastatteluihin. YIS:llä yleisesti käytössä olevat menetelmät –kappale perustuu haastatteluista saatuihin tietoihin.

Seuraavissa kappaleissa kerrotaan tarkemmin YIS:llä käytössä olevista sovelluksen käyttöönottojen prosessimenetelmistä. Yksityiskohtaisemmin perehdytään YIS:llä haastatteluiden perusteella yleisimmin käytössä olevaan ASAP-menetelmään. Lisäksi kerrotaan tarkemmin MDM SC –sovelluksen käyttöönottoprosessista. Näiden käyttöönottomenetelmien lisäksi YIS:llä on ollut yleisesti käytössä lineaarinen vesiputousmalli sellaisenaan, iteroiva vesiputousmalli ja Scrum-käyttöönottomalli. Määrittelyvaiheessa on näiden lisäksi joissakin projekteissa ollut käytössä protoilumalli.

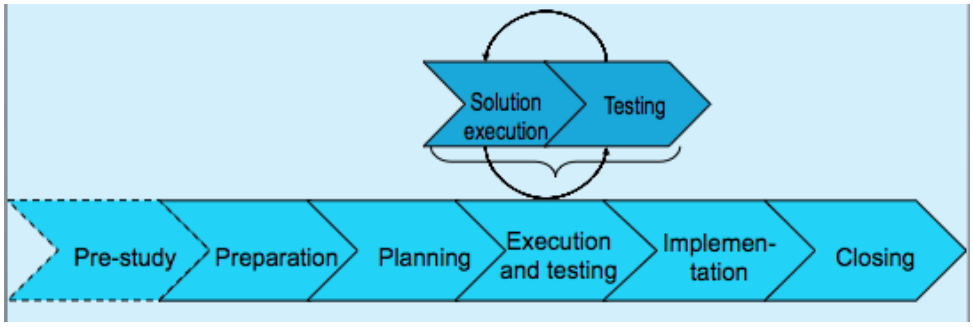
7.1 Sovelluksen käyttöönoton menetelmät YIS:llä yleisesti

Haastatteluiden perusteella voidaan sanoa, että YIS:llä on ollut käytössä monenlaisia sovelluksen käyttöönottoprosessimalleja. Prosessi- ja projektimallit ovat vaihdelleet vuosien aikana ja käytössä on ollut useita erilaisia menetelmiä tietysti riippuen käyttöönotettavan sovelluksen koosta ja siihen liittyvistä tavoitteista. Nykyisistä käytössä olevista menetelmistä nousi selkeästi käyttökerroiltaan kuitenkin yksi ylitse muiden, koska useimmat haastatelluista totesivat pääasiallisesti käyttävän vesiputousmalliin pohjautuvaa ASAP-menetelmää sovellusten käyttöönottojen prosessimallina. Vaikka YIS:n käyttämä prosessimallinnusmenetelmä ei olekaan täysin vastaava SAP:n lanseeraaman ASAP –menetelmän kanssa, käytetään menetelmästä tutkimuksessa ASAP-nimeä, koska sillä nimellä menetelmä YIS:n käytössä myös tunnetaan.

7.2 YIS:lle muokatun ASAP-menetelmän vaiheet

SAP:n lanseeraamaa ASAP-menetelmää käytetään YIS:llä yrityksen omiin käyttötarkoituksiin ja periaatteisiin muunneltuna versiona lähinnä suurempien SAP-toiminnanohjausjärjestelmän käyttöönottoprojektien mallina. Lisäksi ASAP-menetelmää käytetään soveltaen yleisesti myös muiden pienempien sovellusten käyttöönottoprojekteissa.

Malli pohjautuu hyvin vahvasti perinteiseen vesiputousmalliin ja noudattaa näin hyvin pitkälti samanlaista vaiheistusta kuin vesiputousmalli. Ominaisuuksiltaan se muistuttaa jonkun verran iteratiivista vesiputousmallia, jossa on ainakin osittain mahdollista palata takaisin vaiheen alkuun, ja suorittaa se näin alusta uudelleen. Käytännössä sitä ei kuitenkaan tehdä kuin ASAP-mallin toteutus- ja testausvaiheessa. Jokaisen vaiheen lopussa saavutetaan tietty tulos, ja ennen kuin tulos portti (Gate) on saavutettu, ei prosessin vaiheistuksessa voida mennä eteenpäin.



Kuva 7. ASAP-menetelmän vaiheet

Käyttöönottoprosessi alkaa Pre-study- eli esitutkintavaiheesta, jossa tehdään taustaselvitystä ja pyritään selvittämään korkealla tasolla tarpeita ja tavoitteita. Samalla tehdään tarkempi selvitys mm. asiakkaan käytössä olevista prosesseista. Tämän vaiheen lopussa on projektin aloituksen päätöstä koskeva portti - Gate 0 (G0 = Project Start Approval), jonka seurauksena siis tehdään päätös aloitetaanko projekti vai ei.

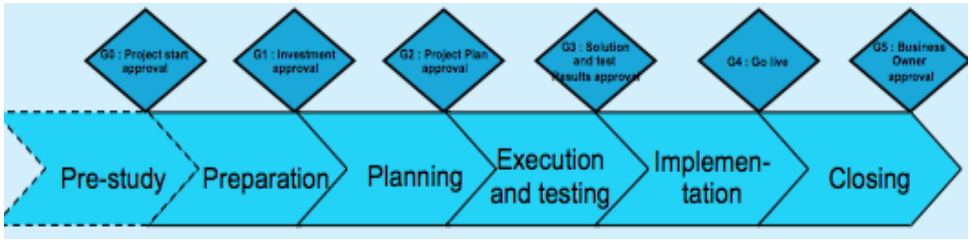
Preparation- eli valmisteluvaiheessa keskitytään yksityiskohtaisemmin sovelluksen vaatimusmäärittelyyn, ja samalla määritellään tarkemmin olemassa olevat prosessit, ja kuinka prosessit tulevat uuden sovelluksen myötä muuttumaan. Valmisteluvaiheessa määritellään tarkemmin uudesta sovelluksesta saatavat hyödyt prosesseissa. Tässä vaiheessa määritellään myös lopullisesti käytettävä sovellusmalli ja sen toimittaja. Lisäksi tuotoksena on tarkat taloudelliset suunnitelmat, jotka ovat perusteena seuraavan portin (G1 = Invensment Approval) hyväksymisvaiheessa.

Planning- eli suunnitteluvaiheessa tehdään perusteellinen projektisuunnitelma, joka pitää sisällään kaiken oleellisen uuden sovelluksen käyttöönottoon liittyen. Suunnitteluvaiheessa järjestetään myös virallinen kick-off, joka laukaisee projektin aloituksen käytännön tasolla liikenteeseen. Suunnitteluvaiheessa tehdään myös alustava testaussuunnitelma, jota sitten myöhemmin voidaan hyödyntää pohjana tarkemmalle testaussuunnittelulle. Vaihe läpäistään silloin, kun projektisuunnitelma hyväksytään asiakkaan toimesta (G2 = Project Plan Approval).

Execution and testing- eli suoritus- ja testausvaiheessa tehdään varsinainen työ eli sovellus tuotteistetaan vaatimuksien mukaiseksi, ja samalla sovelluksen eri osa-alueet testataan huolellisesti. Tämä vaihe pitää yleensä sisällään ns. iteratiivisen suorittamisen. Se tarkoittaa sitä, että tässä vaiheessa on mahdollista, ja jopa suositeltavaa, valmistaa sovellus käyttökelpoiseksi pienemmissä kokonaisuuksissa. Jokainen kokonaisuus tehdään valmiiksi testauskierroksilla niin, että kutakin kokonaisuutta korjataan ja testataan niin kauan kunnes se todetaan hyväksyttävällä tavalla toimivaksi. Vaiheen viimeisenä testivaiheena on hyväksyntätestaus, jossa asiakas testaa, ja sen jälkeen hyväksyy tai hylkää tuotoksen. Seuraavaan vaiheeseen eteneminen vaatii siis hyväksyntätestien hyväksynnän asiakkaalta (G3 = Solution and Test Results Approval).

Introduction- eli varsinainen käyttöönottovaihe pitää sisällään mahdollisesti varsinaisen sovelluksen pilotoinnin. Tässä vaiheessa viimeistellään vielä kaikki dokumentaatiot koulutusmateriaaleineen. Samalla myös tiedotetaan sekä koulutetaan varsinaiset loppukäyttäjät niin, että heillä on täysi valmius työskennellä kyseisen sovelluksen kanssa itsenäisesti. Tämän vaiheen hyväksytty suorittaminen edellyttää sovelluksen täyttä käyttövalmiutta. Ennen lopullista vaiheen hyväksyntää kaiken dokumentaation koulutusmateriaaleineen tulee olla hyväksyttynä asiakkaan toimesta (G4 = Go-Live).

Käyttöönottovaiheen jälkeen on vielä yksi vaihe (Closing), jolla asiakas antaa virallisesti hyväksynnän toimituksen sisällöstä kokonaisuutena. Tällä siis vahvistetaan asiakkaan puolesta se, että asiakas on tyytyväinen käyttöönottoon. Samalla varmistetaan, että projekti on ollut asiakkaan kannalta onnistunut ja näin se voidaan lopullisesti sulkea. Tässä vaiheessa on yleensä kerätty jo asiakkaan puolelta ja loppukäyttäjiltä mahdollisia tulevaisuuden kehitysideoita sovelluksen suhteen. Näistä tehtyä dokumentaatiota voidaan sitten käyttää hyväksi jatkokehitystä tehtäessä. Samalla tehdään erillinen kirjallinen sopimus sovellukseen liittyvistä tukitoimista. Viimeisen vaiheen lopullinen hyväksyntä (G5 = Business Owner Approval) annetaan asiakkaan toimesta ja tällä projekti suljetaan päättyneeksi.



Kuva 8. ASAP-mallin Gate:t (vaiheiden portit ja niistä saatavat tulokset)

7.3 MDM SC:n käyttöönottoprosessin vaiheet

MDM SC:n käyttöönottoprosessissa ei sinällään tietoisesti käytetty mitään tiettyä nimettävissä olevaa käyttöönottoprosessimallia, mutta siitä löytyy selviä yhtymäkohtia mm. iteroivan vesiputousmallin kanssa ja ASAP-käyttöönottomallin vaiheistuksien kanssa. Samalla kuitenkin itse prosessi eteni hyvin pitkälti Scrum-menetelmän perusperiaatteiden mukaan pienemmissä osissa niin, että itse sovelluksen valmistaminen tehtiin täysin eri kokonaisuutena sovelluksen toimittajan toimesta, ja kaikki muu sovelluksen kokonaisprosessin kannalta oleellinen toiminnallisuus tehtiin omina kokonaisuuksinaan valmiiksi pieninä osakokonaisuuksina. Sovelluksen käyttöönottoprosessissa käytettiin teoriaosuudessa esitellyistä prosessimalleista kokonaisuuden kannalta lähimpänä XP-mallin kaltaista sovelluksen käyttöönottoprosessimenetelmän vaiheistusta, josta kerrotaan tarkemmin tulevissa kappaleissa.

Sovelluksen vaatimusmäärittely tehtiin YIS:n toimesta, koska alkuperäinen tarkoitus sovellukselle oli lähinnä olla YIS:n toimintaa tukeva työkalu.

Vaatimusmäärittely lähetettiin useammalle toimittajalle, jotka sitten kävivät esittelemässä omia ratkaisujaan määrittelyjen perusteella. Kahden parhaan vaihtoehdon välillä voiton vei sovellus, joka soveltui paremmin juuri YIS:n määrittelemiin käyttötarkoituksiin. Tämän jälkeen valitun sovelluksen toimittajan kanssa tehtiin erikseen sopimus toimituksesta.

Valitulle sovellukselle (MDM SC) tehtiin vielä tarkemmat vaatimusmääritykset, jotka lähtökohtaisesti perustuivat hyvin pitkälti SAP:n Template:n asettamiin vaatimuksiin. Sovelluksen tarkempia ja yksityiskohtaisempia määrityksiä oli tekemässä useampi YIS:n työntekijä ja lisäksi sovelluksen kokonaistoiminnallisuuden takia mukaan piti saada myös muiden toimittajien edustajia. Jo tässä vaiheessa päätettiin YIS:n toimesta, että yritys haluaa itse hallinnoida tiedostonsiirron valitun sovelluksen ja SAP:n välillä. Tämä tarkoitti mm. sitä, että BizTalk-tiedostosiirto sovittiin hoidettavan YIS:n toimesta. Lisäksi SAP:iin tarvittiin uusia ohjelmia, jotta tietojenlukua SAP:sta MDM SC:iin ja MDM SC:sta SAP:iin pystyttäisiin tekemään hallitusti. SAP:n ohjelmointi päätettiin hoitaa käyttämällä ulkoista konsultointia.

Esitutkinnan ja määrittelyvaiheen jälkeen tehtiin vielä tarkempi projektin kuvaus ja siihen liittyvä projektisuunnitelma projektin etenemisestä ja toteutuksesta. Samalla myös perustettiin projektille työryhmä, johon nimettiin vastuulliset henkilöt kullekin osa-alueelle. Projektiryhmään kuului YIS:n vastuuhenkilöiden lisäksi itse sovelluksen toimittajan edustaja, SAP–ohjelmoinnista vastaava konsultti sekä ulkoisia BizTalk –konsultteja.

Varsinainen projektityö eli sovelluksen pystyttäminen ja testaaminen aloitettiin hyvin pian hyväksytyn projektisuunnitelman jälkeen. Sovelluksen toiminnallisuutta lähdettiin rakentamaan pienissä osissa niin, että perustoiminnallisuus pyrittiin rakentamaan ensimmäisessä vaiheessa valmiiksi ennen kuin siirryttiin tarkempien määrityksien tekemiseen. Perustoiminnallisuuden pystyttämisen jälkeen keskityttiin tekemään vaatimusmäärittelyissä todettujen SAP:n Template:n vaatimat erityisvaatimukset sovelluksen suhteen. Sen jälkeen, kun sovelluksen tarkemmat määrittelyt saatiin tehtyä, alettiin pystyttämään sovelluksen kokonaisprosessin kannalta tärkeitä integraatioyhteyksiä ja SAP:iin rakennettavia ohjelmia.

Jokainen erillinen osa-alue tehtiin erillisinä suorituksina valmiiksi niin, että jokainen osa-alue sisälsi omat testi- ja korjauskierroksensa. Testejä ja korjauksia tehtiin niin kauan kunnes pystyttiin toteamaan, että kyseinen osa-alue toimii yksittäisenä kokonaisuutena oikein. Testikierroksia tuli yhteensä useita kymmeniä lukuun ottaen kaikkien eri osa-alueiden eri testikierrokset. Tämän jälkeen tehtiin vielä yksikkö- ja integraatiotestit koko sovelluksen käyttöprosessista alusta loppuun niin, että voitiin olla varmoja kokonaisprosessin toimivuudesta. Viimeisenä testinä ennen sovelluksen pilotointia järjestettiin hyväksyntätestaukset asiakkaan kanssa.

Asiakkaalta saadun hyväksynnän jälkeen sovellus otettiin pilotointi-käyttöön pienissä osissa niin, että ensin otettiin käyttöön vain pieni osa itse sovelluksen toiminnallisuudesta. Ensimmäiset tuotantokäytön datasiirrot tehtiin aluksi pienillä määrillä ja määriä kasvatettiin vähitellen kuukausittain.

Sovellus ei ole vielä tätä kirjoittaessa käytössä täydellä volyymilla, mutta portaittaisen käyttöönoton johdosta siihenkin vaiheeseen päästään piakkoin ja asiakkaalta tullaan todennäköisesti saamaan lopullinen sovelluksen hyväksyntä, jolloin voidaan todeta sovelluksen käyttöönottoprosessi suljetuksi. Sovellukseen tulevasta kehitystyöstä on jo alustavasti tehty ehdollinen sopimus niin, että mikäli asiakas pilotointi-käytön jälkeen toteaa sovelluksen toimivan perustoiminnoiltaan vaadittavalla tasolla, voidaan heti sen jälkeen siirtyä luontevasti sovelluksen kehitystyöhön.

8 JOHTOPÄÄTÖKSET JA KEHITYSEHDOTUKSET

Tässä luvussa kerrotaan tarkemmin tutkimuksen teoriaosuudessa käytettyjen prosessimallien ja empiirisen tutkimuksen tuloksena saatujen prosessimallien välisestä vertailusta. Vertailujen ja haastatteluiden tulosten perusteella annetaan lisäksi johtopäätökset ja muutamia kehitysehdotuksia yrityksen sovelluksen käyttöönottomenetelmien käytöstä.

8.1 Vertailu eri käyttöönottomallien kesken

Käyttöönottomallien vertailussa ovat mukana kaikki teoriaosuudessa esitellyt käyttöönottoprosessin mallit; vesiputous- , Scrum- ja XP-malli, joita verrataan YIS:llä yleisemmin käytettyihin prosessimalleihin. Tutkimuksen perusteella YIS:llä selvästi yleisimmin käytössä ollut käyttöönottomalli oli sovellettu ASAP-malli, joten vertailua tehdään pääosin siihen.

8.1.1 Vesiputousmallin piirteet YIS:llä

Niin kuin jo teoriaosuudessa on mainittu, vesiputousmalli on kautta historian käytetyin sovelluksen käyttöönottoprosessien malli, ja sen vuoksi se toimiikin yleisesti, ainakin jollakin tasolla, aina kaikkien sen jälkeen kehitettyjen menetelmien pohjana. Vertailupohjana tutkimuksessa käytetään iteroivan vesiputousmallin vaiheistusta.

Haastatteluiden perusteella myös YIS:llä on käytetty vesiputous–mallia käyttöönottoprosessien menetelmänä. Käytössä on aikojen kuluessa ollut niin perinteinen lineaarinen vesiputousmalli kuin iteratiivinen vesiputousmalli. YIS:llä on ollut käytössä myös vesiputousmallin omaan käyttötarkoitukseen sovelletut versiot. Näistä syistä johtuen vesiputousmallin piirteitä onkin helppo löytää myös YIS:llä käytetyimmän ASAP-mallin vaiheistuksesta.

YIS:n ASAP-menetelmän vaiheistus perustuu tutkimuksen mukaan selvästi suoraan vesiputousmalliin. Selvästi tunnistettavissa olevia samoja vaiheita ovat ainakin esitutkimus-, määrittely- ja suunnitteluvaihe, jotka sellaisenaan ovat täsmälleen samanlaisia kuin vesiputousmallissa. Oikeastaan ainoa eroavaisuus vaiheistuksessa on se, että toteutus- ja testivaihe on ASAP-mallissa yhdistetty samaan vaiheeseen, kun vesiputousmallissa ne ovat eriytettynä omiksi vaiheikseen. Näiden vaiheiden jälkeen kummassakin menetelmässä seuraava vaihe on käyttöönottovaihe. ASAP-menetelmässä tulee vielä käyttöönottovaiheen jälkeen erillinen projektin sulkemisvaihe, jota ei vertailussa käyttämässäni iteroivassa vesiputousmallista erikseen ole käytetty. Huomioitavaa kuitenkin on, että esim. teoriaosuudesta löytyvän lineaarisen vesiputousmallin versiossa mainitaan erikseen projektin päättämisvaihe.

Selvänä eroavaisuutena iteroivan vesiputousmallin ja YIS:llä käytetyn ASAP-menetelmän välillä on havaittavissa juurikin niiden iteratiivisuus. Iteratiivisessa vesiputousmallissa on käytännössä mahdollista aina palata

vaiheistuksen vaiheissa taaksepäin, kun taas ASAP-menetelmässä iteratiivisuus esiintyy vain toteutus- ja testivaiheen sisällä, jossa on mahdollista tehdä useita toteutus- ja testivaiheita iteratiivisen mallin mukaisesti. Huomioitavaa kuitenkin on, että haastatteluiden perusteella iteratiivisuutta on käytännön projekteissa mahdollisuus toteuttaa myös ASAP-menetelmää käyttäen. Oleellista tässä huomiossa on, että mahdollisuus iteratiiviseen menettelyyn on sitä helpompaa, mitä pienemmästä projektista on kysymys ja mitä aikaisemmin prosessissa ilmaantuu muutostarpeita. Toisin sanoen isoimmissa SAP-käyttöönottoprojekteissa palaaminen vaiheistuksessa taaksepäin ei ole niinkään mahdollista, kun taas pienemmissä projekteissa se on mahdollista, mikäli prosessin muutostarpeet ilmaantuvat tarpeeksi ajoissa eikä vasta esimerkiksi käyttöönottovaiheessa.

8.1.2 Scrum-mallin piirteet YIS:llä

Scrum-menetelmästä löytyy myös selkeitä yhtymäkohtia ASAP-menetelmän kanssa, mutta lähtökohtaisesti sen vaiheistusmalli on tarkoitettu hieman pienemmille projekteille, joissa on mahdollista toteuttaa käyttöönottoprosessia valmistamalla sovellusta pienemmissä osissa. Scrum-mallin suurin ero ASAP-menetelmään verrattaessa on sen prosessin eteneminen valmistaen tulevaa sovellusta pienemmissä osakokonaisuuksina. Myös ASAP-mallissa tehdään kokonaisuuksia pienemmissäkin osissa, mutta niitä ei prosessin vaiheistuksessa määritellä aina erikseen omiksi kokonaisuuksiksi, vaan ne suunnitellaan ja valmistetaan aina suurempien kokonaisuuksien kanssa samanaikaisesti.

Vaikka Scrum-mallin perustoiminnallisuus poikkeaa selvästi ASAP-menetelmästä, sen sprintin vaiheistuksesta löytyy samankaltaisia vaiheistuksia. Selkeimmin niistä esiin tulevat yhtäläiset vaiheet ovat tuotteen työlistan kerääminen, sprintin suunnittelu ja sprintti.

Tuotteen työlistan keräämistä voidaan suoraan verrata ASAP-menetelmän esitutkintavaiheeseen, jossa kartoitetaan sovellukseen tarvittavia ominaisuuksista ja toiminnallisuuksista. Sprintin suunnitteluvaihe puolestaan vastaa täysin ASAP-menetelmän suunnitteluvaihetta. Scrum-menetelmän sprinttivaihe vastaa ASAP-menetelmän toteutusvaihetta, jossa itse käytännön työ sovelluksen valmistamiseksi tehdään.

Haastatteluiden perusteella myös Scrum-mallia on käytetty YIS:lla sovelluksien käyttöönottoprosessien menetelmänä. Menetelmää on käytetty juurikin pienempien sovellusten valmistuksen projekteissa, joissa voidaan selvästi perehtyä aina pienempiin sovelluksen osakokonaisuuksien valmistamiseen kerrallaan. Myös MDM SC –sovelluksen käyttöönotossa prosessi eteni niin, että sovelluksen käyttöprosessin kannalta oleelliset osa-alueet valmistettiin erillisinä kokonaisuuksina. Sovelluksen käyttöönottoprosessi vastasi kokonaisuutena kuitenkin enemmän YIS:lla yleisesti käytössä olevaa ASAP-mallia.

8.1.3 XP-mallin piirteet YIS:llä

XP-mallin vaiheistus vastaa hyvinkin pitkälti ASAP-mallin vaiheistusta. XP-mallin ensimmäisenä vaiheena oleva tutkimusvaihe vastaa täysin ASAP-menetelmän Pre-Study-vaihetta, jossa tutkitaan myös sovelluksen kannalta oleellisia toiminnallisuuksia ja mahdollisuuksia niiden toteuttamiseen. Seuraavana oleva suunnitteluvaihe vastaa myös täysin ASAP-mallin suunnitteluvaihetta. Samoin julkaisun iteraatiot-vaihe yhdessä tuotteistamis- ja ylläpitovaiheen kanssa vastaa käytännössä suoraan ASAP-mallin toteutusvaihetta. Samoin XP-mallin päätösvaihe vastaa täysin ASAP-mallin sulkemisvaihetta.

XP-mallia ei tietoisesti ainakaan haastatteluiden perusteella ole YIS:lla yleisesti käytetty, mutta sen vaiheistus ja perusperiaatteet vastaavat hyvinkin pitkälti piirteiltään case-sovelluksen käyttöönottoprosessia iteratiivisuutensa myötä. Lukuun ottamatta XP-mallin lähtökohtaista ohjelmointilähtökohtaisuutta sen perusperiaatteet vastaavat hyvin lähellä MDM SC –sovelluksen käyttöönottoprosessissa toteutettua toimintamallia. Sovelluksen käyttöönottoprosessissa tehtiin kokonaistoiminnallisuutta pieninä osina niin, että jokainen osa-alue tehtiin palasina iteroiden valmiiksi ja näin myös XP-mallissa oleellinen iteratiivisuus toteutui jokaisen sovelluksen osa-alueen toteutuksessa.

Suurimpina eroina sovelluksen käyttöönottoprosessiin on havaittavissa, että sovelluksen toteutusvaiheessa ei tehty jatkuvaa jokapäiväistä katselmointia, eikä jokaista erillistä osa-aluetta hyväksytetty asiakkaalla erikseen, vaan asiakkaalle vietiin hyväksyttäväksi periaatteessa täysin valmis kokonaisratkaisu. Lisäksi XP-menetelmälle tyypillistä pariohjelmointia ei case-sovelluksen toteutusvaiheessa ollut missään muotoa havaittavissa.

8.2 YIS:lle ehdotettu sovelluksen käyttöönottomalli perusteluineen

YIS:llä on tälläkin hetkellä käytössä yli 200 eri sovellusta. Niiden määrää pyritään vähentämään, mutta tarvetta uusillekin sovelluksille tulee jatkossa aina olemaan, koska YIS:llä käytettävien ja käyttöönotettavien sovelluksien kirjo on niin laaja. Tutkimuksessa ehdotetun käyttöönottoprosessimallin valinta on tarkoitettu lähinnä case-sovelluksen kaltaisten sovellusten tuleville käyttöönottoprosesseille käytettäväksi.

Käytettävä malli YIS:llä riippuu aina itse käyttöönotettavasta sovelluksesta, eli kuinka laajaa ja monimutkaista sovellusta ollaan ottamassa käyttöön. Mitä suurempaa sovellusta ollaan ottamassa käyttöön, sitä vähemmän sovelluksen käyttöönotossa saa esiintyä iteratiivisuutta. Suuressa roolissa on myös, kuinka paljon asiakas tulee osallistumaan itse käyttöönottoprosessiin. Mitä pienempi sovellus, sen helpompi on ottaa asiakas mukaan prosessiin ja sovelluksen käyttöönotossa tapahtuviin määrityksiin ja kehittämiseen. Asiakaslähtöisyys sovelluksen kokoluokan ja monimutkaisuuden kanssa määrittelee hyvin pitkälle käytettävän käyttöönottoprosessimenetelmän muodon ja samalla sen millaista menetelmää tulisi kulloinkin käyttää.

Tutkimuksen perusteella havaittiin, että YIS:llä ei pääasiassa ollut ainakaan virallisesti käytössä kuin yksi käyttöönottoprosessimalli – ASAP. Sen lisäksi joissakin projekteissa oli käytetty muitakin menetelmiä, mutta virallisesti käytössä ei ollut kuin YIS:n soveltama ASAP-menetelmä. Menetelmä sinällään on hyvin toimiva suuremmissa projekteissa kuten SAP:n toiminnanohjausjärjestelmän käyttöönottoprojekteissa, mutta sellaisenaan se ei ole käyttökelpoinen pienemmille case-sovelluksen kaltaisille käyttöönotoille, koska menetelmä ei lineaarisuutensa vuoksi toimi pienempien käyttöönottoprojektien nopeasti muuttavassa toimintaympäristössä.

Erittäin hyvänä esimerkkinä tästä on MDM SC –sovelluksen käyttöönottoprosessi, jossa ei tosin missään vaiheessa käytetty tietoisesti mitään tiettyä menetelmää, mutta vaiheistukseltaan se vastasi alussa täysin ASAP-menetelmän vaiheistusta. Alkuperäisen projektisuunnitelman mukaankin oli tarkoitus edetä vaihe kerrallaan eteenpäin niin, että iteratiivista menettelyä ei juurikaan olisi käytetty. Projekti kuitenkin eli prosessin aikana niin paljon, että käytännössä oli mahdotonta edetä vaihe kerrallaan vain eteenpäin ja siitä johtuen jouduttiinkin palaamaan monta kertaa projektin aikana vaiheistuksessa taaksepäin. Prosessissa ilmenneet nopeat muutostarpeet käytännössä pakottivat omaksumaan erilaisten ketterien menetelmien toimintatapoja. Prosessin etenemisestä on havaittavissa selvästi, että jo projektin aikaisessa vaiheessa päädyttiin ratkaisuun, jossa sovelluksen toteutusta alettiin valmistamaan pienempinä osa-alue kerrallaan, koska sovelluksen prosessin kokonaisvaltainen toiminta sitä vaati. Tämä perustui siihen, että sovelluksen kokonaistoiminnan kannalta siihen liittyi niin monta erillistä osa-aluetta, jotka kaikki olivat riippuvaisia toisistaan, mutta niiden toteuttaminen yksi kerrallaan oli kuitenkin välttämätöntä toimivan kokonaisratkaisun aikaansaamiseksi. Tässä vaiheessa projektia prosessi eteni Scrum-menetelmän kaltaisilla menetelmillä.

Prosessin edetessä havaittiin projektin onnistumisen kannalta suuria riskitekijöitä, ja projektissa jouduttiin jälleen ottamaan taka-askeleita. Käytännössä tämä tarkoitti sitä, että prosessissa jouduttiin palaamaan takaisin sovelluksen kokonaisprosessin kannalta perustoiminnallisuuksien toteutuksiin, ja siksi jouduttiin tekemään jälleen uusia kierroksia sovelluksen toteutus- ja testausvaiheessa. Useiden pienempien vaikeuksien ja uusien kierroksien lisäksi prosessissa jouduttiin myös palaamaan kokonaan prosessin alkuvaiheisiin, jossa itse sovelluksen määrittelyjä jouduttiin tekemään uudelleen ja näin projektissa palattiin jälleen alkutekijöihin.

Näihin faktoihin perustuen voidaan todeta, että case-sovelluksen käyttöönottoprosessimenetelmänä oli lopulta lähimpänä XP-menetelmän prosessia. Kaikki toimintatavat eivät olleet täsmälleen samanlaisia kuin XP-menetelmässä, mutta prosessin eteneminen vastasi vaiheistukseltaan ja iteratiivisuudeltaan hyvin pitkälti juuri XP-menetelmää.

Näin ollen onkin perusteltua todeta, että case-sovelluksen kaltaisten sovelluksien käyttöönottoprosessimenetelmäksi voidaan YIS:lle suositella XP-menetelmän kaltaista metodia. Lähtökohtaisesti XP-mallin käyttäminen

YIS:n käyttötarkoituksiin sellaisenaan ei voisi toimia, koska sen perusperiaatteet perustuvat ohjelmistosuunnitteluun ja sitä kautta ohjelmointi näyttelee suurta osaa koko menetelmästä. Kuitenkin hiukan sovelletulla versiolla, mm. jättämällä ohjelmointilähtökohtaisuus pois, siitä saataisiin YIS:lle erittäin toimiva sovellusten käyttöönottoprosessimalli. Se on erittäin käyttökelpoinen prosessimalli YIS:n pienille ja keskisuurille sovellusten käyttöönotoille. XP-menetelmän käyttöä pitää kuitenkin vielä soveltaa sen mukaan, kuinka suurta sovellusprojektia ollaan toteuttamassa.

XP-menetelmään oleellisesti kuuluvat jatkuvat katselmoinnit vahvistavat yhteistyön toimivuutta asiakkaiden ja YIS:n välillä. Jatkuvien katselmointien pitäminen helpottaa huomattavasti sovelluksen hyväksymisprosessia, koska kumpikin osapuoli on koko käyttöönottoprosessin ajan paremmin tietoisia haluttujen toimintojen vaatimuksista ja missä niiden kanssa mennään. XP-menetelmän yhtenä merkittävänä ominaisuutena onkin, että siinä otetaan paremmin asiakkaan mielipiteet ja tarpeet huomioon, jolloin asiakkaan ja YIS:n välille saadaan rakennettua parempaa yhteistyötä. Lisäksi asiakas oppii koko projektin ajan lisää sovelluksen eri toiminnallisuuksista ja niihin liittyvistä teknisistä asioista. XP-menetelmälle tyypillinen jatkuva kanssakäyminen asiakkaan kanssa edesauttaa YIS:n tietoisuutta asiakkaan tarpeista ja käytössä olevista prosesseista. Samaan aikaan asiakas oppii projektin alusta lähtien enemmän ja enemmän itse sovelluksen toimintaa. Näiden seikkojen edellytyksenä lopputuloksena saadaan aina tehokkaammin toteutettu, ja asiakkaan tarpeisiin paremmin yksilöity ja kokonaisuuden kannalta toimivampi sovellus.

8.3 Kehitysehdotuksia sovellusten käyttöönottotapoihin YIS:llä

Ensimmäisenä yleisenä kehitysideana olisi, että YIS:llä tulisi harkita tarkemmin, millaisia sovelluksen käyttöönottoprosessin menetelmiä käytetään erilaisissa sovelluksen käyttöönotto- ja kehitysprojekteissa. Tällä hetkellä periaatteessa yritys käyttää virallisena ohjenuoranaan vain yritykselle sovellettua ASAP-menetelmää, jota käytännössä yritetään muokata projektikohtaisesti erilaisiin lähtökohtiin sopivaksi.

Toisena kehitysideana olisi, että erilaiset sovelluksen käyttöönottoprosessit luokiteltaisiin niiden koon ja asiakaslähtöisyyden perusteella luokkiin, joiden perusteella käytettävä menetelmä valittaisiin ennen projektien virallista aloitusta. Luokat voitaisiin jakaa esimerkiksi alla taulukon 2 mukaisesti. Taulukossa ei puututa usein kokoluokkien perusteena olevaan taloudelliseen kokoluokkaan.

On myös huomioitavaa, että olkoon kyseessä minkä tahansa kokoluokan projekti, itse menetelmää tulee aina muokata vaatimusten mukaiseksi. Yksikään menetelmä ei ole sellaisenaan täysin valmis menetelmä käytettäväksi. Jokaisen sovelluksen käyttöönottoprosessien vaatimuksen poikkeavat toisistaan, jolloin sen edellyttämät toimintatavat ja käytännöt muuttuvat sen mukana.

Taulukko 2. Kehitysehdotus sovellusten käyttöönottojen luokituksista ja ehdotukset käytettäviksi prosessimalleiksi YIS:llä

#	Projektien luokitukset	Ehdotus käytettäväksi prosessimalliksi
1	Sovelluksen pienkehitys	XP
2	Sovelluksen kehitys	XP / Scrum
3	Pienen sovelluksen käyttöönotto (asiakas vahvasti mukana)	XP
4	Pienen sovelluksen käyttöönotto	XP (mukautettu)
5	Keskisuuren sovelluksen käyttöönotto (asiakas vahvasti mukana)	XP / Scrum / ASAP
6	Keskisuuren sovelluksen käyttöönotto	Scrum / ASAP
7	Suuren sovelluksen käyttöönotto	ASAP

Tutkimuksessa tehtyjen haastatteluiden perusteella tuli selkeästi ilmi, että asiakas halutaan osallistuvan enemmän varsinkin pienempiin projekteihin. Mitä pienemmästä projektista on kysymys, sitä useammin prosessiin voidaan ottaa mukaan jatkuvaa katselmointia, jossa asiakkaalta on edustajat mukana hyväksymässä sovelluksen eri toimintoja. Asiakkaan tarpeet ja vaatimukset tulee ottaa paremmin huomioon ja näin asiakasrajapintaan saadaan jatkuvampaa yhteistyötä aikaiseksi. Tällä varmistetaan, että asiakas on koko prosessin ajan tietoinen missä mennään ja näin asiakas myös oppii koko projektin alusta asti sovelluksen eri toiminnallisuuksia ja niihin liittyviä teknisistä asioista. Samalla prosessimallin interaktiivisuus asiakkaaseen edesauttaa sovelluksen toimittajan tietoisuutta asiakkaan tarpeista ja käytössä olevista prosesseista. Nämä kaikki yhdessä johtaa siihen, että lopputuloksena saadaan nopeammin ja tehokkaammin toteutettu ja asiakkaan tarpeisiin paremmin toimiva lopputuote.

8.4 Johtopäätökset

Sovelluksien käyttöönotoille on kehitetty erilaisia prosessimalleja jo 1970 –luvulta asti, jolloin ensimmäinen versio vesiputousmallista tuli julkiseen käyttöön. Siitä lähtien vesiputousmallia on käytetty prosessimallina aina näihin päiviin asti. Sen jälkeen tulleet menetelmät ovat joko olleet suoraan vesiputousmallista muunneltuja versioita, tai vesiputousmallia on käytetty ainakin uuden prosessimallin kehityksen pohjana. Vaiheistukset eivät sinällään ole muuttuneet, eikä vaiheistuksien muuttaminen toisenlaiseksi olisikaan kovin järkevää. Sovelluksien käyttöönottojen prosessimallien vaiheistukset ovat saaneet erilaisia nimityksiä, mutta käytännössä niillä tarkoitetaan aina samoja asioita.

Erilaisia prosessimalleja on nykyään saatavilla useita kymmeniä erilaisia ja niiden menetelmät ja tarkoituksenmukaiset käyttökohteet vaihtelevat hyvinkin paljon. Kaikilla on kuitenkin yhteistä se, että ne tavalla tai toisella pohjautuvat vesiputousmalliin ja niiden käyttöä sellaisenaan ei voida suositella minkä tahansa sovelluksen käyttöönottoprojektin käyttöön. Pro-

sessimallit voivatkin olla vain suuntaa-antavia ohjenuoria, joiden avulla voidaan määritellä ja toteuttaa käyttöönottoprojekteja aina kunkin projektin tarpeiden mukaan. Ne toimivat lähinnä peruspilarina sille, miten prosesseja tullaan käytännössä toteuttamaan. Kullekin projektille voidaan aina valita jokin tietty parhaiten projektia palvelema prosessimalli, mutta lähestulkoon poikkeuksetta prosessimalleja joudutaan aina muokkaamaan itse projektin tarpeita tyydyttäväksi. Vaikka prosessimalli olisi kuinka hyvin valittuna ja prosessia pyrittäisiinkin toteuttamaan kirjaimellisesti, hyvin harvoin itse käytännön toteutuksessa päästään absoluuttisesti toteuttamaan prosessimallin ohjeistamia käytäntöjä.

Prosessimalleja on käytännössä kahdenlaisia – lineaarisia ja iteratiivisia malleja. Niiden käyttötarkoitukset eroavat toisistaan niin, että lineaarisia malleja suositellaan käytettäväksi suuremman kokoluokan projekteissa, joissa on varattu hyvin aikaa projektin kokonaisvaltaiselle suunnittelulle. Niille on myös tyypillistä, että projektin aikana ei tehdä kovinkaan herkästi paluutta prosessin aikaisempiin vaiheisiin, vaan prosessia pyritään viemään läpi vaihe kerrallaan eteenpäin edeten niin, että jokaisen vaiheen lopussa on joku tietty tuotos valmiina.

Iteratiivisissa prosessimalleissa taasen varaudutaan siihen, että vaiheistuksessa voidaan ja kuuluukin palata vaiheen alkuun tai jopa prosessimallin edellisiin vaiheisiin. Tämä käytäntö perustuu usein siihen, että käyttöönotettavaa sovellusta pyritään valmistamaan pieninä osa-alueina pala kerrallaan ja mahdollisiin yllättäviin muutostarpeisiin pystytään reagoimaan nopeammin eivätkä yllättävät muutokset pysäytä koko projektia, vaan projektia pystytään jatkamaan ketterästi eteenpäin. Iteratiivisissa malleissa ei myöskään niiden ominaisuuksien takia tarvita aina niin pitkäkestoista suunnittelutyötä, vaan prosesseissa päästään nopeasti alkuun ja itse projektia suunnitellaankin usein pienempien kokonaisuuksien yhdistelmänä.

Yrityksellä on ollut käytössä molempia edellä mainittuja sovelluksen käyttöönottoprosessimalleja. Nykyisistä käytössä olevista menetelmistä nousi kuitenkin ASAP-menetelmä selvästi ylitse muiden, koska useimmat haastatelluista totesivat pääasiallisesti käyttävän vesiputousmalliin pohjautuvaa ASAP-menetelmää prosessimallina. Malli pohjautuu hyvin vahvasti perinteiseen vesiputousmalliin ja noudattaakin näin hyvin pitkälle samanlaista vaiheistusta kuin vesiputousmalli. Menetelmä on siis hyvin pitkälle lineaarinen prosessimalli, vaikka sen toteutus- ja testivaiheessa on kuitenkin mahdollista tehdä useita kierroksia kunkin toiminnallisuuden aikaansaamiseksi. Perusperiaatteiltaan sitä käytettäessä ei voida palata prosessissa kuitenkaan vaiheistuksessa taaksepäin. Periaatteessa ASAP-menetelmää käytettäessäkin on mahdollista palata taaksepäin, mutta se tulisi usein niin kalliiksi, että taaksepäin paluuta ei voida toteuttaa. ASAP-menetelmälle onkin tyypillistä, että mitä pidemmälle projektia on viety eteenpäin, sitä hankalampaa sen vaiheistuksissa on palata taaksepäin.

Case-sovelluksen käyttöönottoprosessissa ei sinällään tietoisesti käytetty mitään tiettyä nimettävissä olevaa käyttöönottoprosessimallia, mutta sen eri vaiheista löytyy selviä yhtymäkohtia yrityksessä yleisessä käytössä

olevan ASAP-käyttöönottomallin vaiheistuksien kanssa. Samalla kuitenkin itse prosessi eteni hyvin pitkälle XP-menetelmän perusperiaatteiden mukaan pienemmissä osissa niin, että itse sovelluksen valmistaminen tehtiin täysin eri kokonaisuutena sovelluksen toimittajan toimesta ja kaikki muu sovelluksen kokonaisprosessin kannalta oleellinen toiminnallisuus tehtiin omina kokonaisuuksinaan valmiiksi pieninä osakokonaisuuksina. XP-mallin käyttämiseenkään ei päädytty mitenkään tietoisesti vaan projektissa ilmenneet yllättävät muutokset veivät prosessia enemmän iteroivaan muottiin. Projektin sisällä tapahtui niin monta yllättävää käännettä, että siinä oli pakko palata useita kertoja joko meneillään olevan vaiheen alkuun tai jopa aikaisempiin vaiheisiin. Tämän tyyppiset yllättävät muutokset ovat nykypäivän sovellusten käyttöönottoprosesseissa ennemminkin normaaleja tapahtumia kuin poikkeuksia.

MDM SC –sovellusta käyttöönotettaessa oli jo lähtökohtaisesti tiedossa, että sen kokonaisratkaisu vaatii useita eri sovellusratkaisuja ja niiden toteuttamiseksi yhdeksi kokonaisuudeksi jokainen osa-alue piti tehdä erillisinä kokonaisuuksina valmiiksi. Jo pelkästään tämä puoltaa sitä, että käytettäväksi menetelmäksi olisi alusta pitäen pitänyt valita joku nykyajan ketteristä menetelmistä niiden iteratiivisuuden takia. Tutkimuksen perusteella kuitenkin XP-menetelmä sopii vähän muunneltuna versiona kaikista parhaiten juuri case-sovelluksen tyyppisiin sovelluksen käyttöönottoprosesseihin.

Näistä edellä mainituista syistä johtuen onkin suositeltavaa käyttää aina iteroivaa ketterää prosessimenetelmää, kun käyttöönotettavan sovelluksen koko ja asiakasrajapinta antavat sille mahdollisuuden.

9 YHTEENVETO

Yhteenvetona tutkimuksen annista voidaan todeta, että tutkimuskysymyksiin on pystytty vastaamaan hyvin. Teoriaosuudessa on pystytty kertomaan monipuolisesti, millaisia vaiheita sovellusten käyttöönottoprosessit yleisesti sisältävät. Samalla tutkimuksesta tulee myös ilmi, mitkä käyttöönottomallit MDM SC –sovelluksen käyttöönottoprosessissa ovat toteutuneet. Myös yrityksessä käytössä olevat käyttöönottomallit ja erityisesti eniten käytössä ollut menetelmä saatiin esiteltyä tutkimuksessa. Kehitysehdotuksissa on myös saatu yritykselle muutamia ideoita, kuinka yrityksessä käytettäviä käyttöönottoprosesseja voitaisiin parantaa.

Itse tutkimuksen tekijänä sain paljon oppia erilaisista käyttöönottoprosesseihin saatavilla olevista menetelmistä ja varsinkin MDM SC –sovelluksen käyttöönoton vaiheiden tutkiminen tuotti tekijälle itselleen paljon lisätietoa ja uusia näkökulmia, kuinka käyttöönottoprosessin tulisi edetä ja miten käyttöönottoprosesseja voidaan kehittää.

Käyttöönotetun sovelluksen koko käyttöönottoprosessi oli erittäin haasteellinen siinä tulleiden useiden eri yllätysten takia. Sovelluksen käyttöönotossa ei alun perin tietoisesti käytetty mitään tiettyä prosessimallia, joten siitä johtuen olikin mielenkiintoista ja opettavaista tulkita prosessin vaiheistuksia näin jälkikäteen ja ehkä jopa juuri sen takia sainkin enemmän irti tutkimuksesta oppimismielessä. Sovelluksen käyttöönottoprojektin eri vaiheissa oli havaittavissa eri menetelmille ominaisia piirteitä. Toisin sanoen käytetty menetelmä vaihtui prosessin edetessä lennossa. Aivan projektin alussa käytettiin lähinnä ASAP-menetelmän kaltaista ajatusta, mutta jo projektin alkutekijöissä menetelmä vaihtuikin vähitellen Scrum-menetelmään ja sen jälkeen siirryttiin vielä hieman iteratiivisempaan XP-menetelmään. Nämä muutokset johtuivat siis lähinnä siitä, että projektin aikana tuli niin paljon erilaisia muutoksia, että prosessia oli mahdotonta viedä ainoastaan eteenpäin. Projektin vaiheistuksessa jouduttiinkin palaamaan useampaan kertaan taaksepäin ja mm. toteutusta ja testauksia tehtiin useammalla eri kierroksella. Näistä toimista oli helppo tunnistaa iteratiivisten ketterien menetelmien tunnuspiirteet.

Tutkimusta tehdessä tuli siis perehdyttyä tarkemmin nykyaikaisempiin ketteriin menetelmiin ja niistä aukesi täysin uusi näkökulma koko käyttöönottoprosessien ajatusmaailmaan. Huomattavaa oppia sain myös tutkimuksen yhteydessä tehdyistä haastatteluista. Haastatellut projektityön ammattilaiset antoivat osviittaa, kuinka käyttöönottoprosessit yleensä etenevät ja miten ne käytännössä toteutuvat.

Jatkossa tulenkin käyttämään saamaani oppia tulevissa käyttöönottoprojekteissa ja erityisesti pyrin hyödyntämään saamaani oppia jo lähitulevaisuuden MDM SC –sovelluksen kehitysprojekteissa. Lisäksi toivon, että yrityksessämme otetaan kehitysideat ainakin jollain tasolla huomioon ja käytäntöön.

10 LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. (2002). Agile software development methods: Review and analysis. VTT Publications 478, VTT, Espoo.

Agile Alliance 2001. Agile Manifesto. Agile Alliance. Viitattu 28.12.2011. http://agilemanifesto.org/

Balci, O. ym. Animations to Assist Learning Some Key Computer Science Topics. Blacksburg (VA.): Virginia Polytechnic Institute and State University. Department of Computer Science, 2001. Software Engineering. The Waterfall Model. Viitattu 28.12.2011.
http://courses.cs.vt.edu/~csonline/SE/Lessons/Waterfall/index.html

Beck K. 1999. Extreme Programming Explained: Embrace Change. Canada: Addison Wesley Professional, 224, pdf-tiedosto. Viitattu 29.12.2011.
http://www.mip.sdu.dk/~brianj/Extreme%20Programming%20Explained%20-%20Kent%20Beck%3B%20Addison-Wesley,%201999.pdf

Genieholdings.com Inc. 1999. Viitattu 3.4.2012
http://www.erpgenie.com/asap-solution-manager/asap-methodology

Haikala, I. & Märijärvi, J. 2002. Ohjelmistotuotanto, 8. Uudistettu painos. Helsinki: Talentum Media Oy.

Hybridimenetelmä:Vesiputousmalli ja Scrum. 2009. Viitattu 28.12.2011.
http://hybridimenetelma.suntuubi.com/?cat=11

Hypermedian opetus, 2008a. Te.., A.-M. VPSIST-oppimateriaali,4 Menetelmiä ja malleja, 4.3 Suunnittelumallit, 4.3.1 Ohjelmistotuotannon mallit. Tampere: Tampereen teknillinen yliopisto. Matematiikan laitos ja Hybermedialaboratorio. Viitattu 28.12.2011.
http://hlab.ee.tut.fi/hmopetus/vpsist-oppimateriaali/4-menetelmia-ja-malleja/4-3-suunnittelumalleja/4-3-1-ohjelmistotuotannon-malli

Hypermedian opetus, 2008b. Te.., A.-M. VPSIST-oppimateriaali,4 Menetelmiä ja malleja, 4.4 Ketterät menetelmät. Tampere: Tampereen teknillinen yliopisto. Matematiikan laitos ja Hybermedialaboratorio. Viitattu 28.12.2011.
http://hlab.ee.tut.fi/hmopetus/vpsist-oppimateriaali/4-menetelmia-ja-malleja/4-4-ketterat-menetelmat

Lemmetti, J. 2007. Ohjelmistoarkkitehtuurisuunnittelu Extreme vahvuudet ja
Programmingissa heikkoudet - periaatteet, vahvuudet ja heikkoudet. Teknillinen korkeakoulu. Tietotekniikan osasto. Kanditaatin työ. Pdf-tiedosto. Viitattu 5.1.2012.
http://www.soberit.hut.fi/preago/english/publications/LEMM07Thesis.pdf

Mäntylähti, O. 2009a. Scrum ei ole kaiken ratkaiseva hopealuoti. Tietokone -verkkolehti. Ossi Blogi – Tietotekniikkaa konsultin silmin. Viitattu 27.12.2011.
http://blogit.tietokone.fi/ossi/2009/09/scrum-ei-ole-kaikenratkaiseva-hopealuoti/

Mäntylähti, O. 2009b. Kommentti: It-projektien vesiputousmalli joutaa unholaan. Tietokone -verkkolehti. Viitattu 27.12.2011.
http://www.tietokone.fi/uutiset/2009/kommentti_it_projektien_vesiputousmalli_joutaa_unholaan

Poimala, S. & Heikniemi, J. & Blåfield, H. 2009a. Ketterät käytännöt. Vaatimuksia vai lisäarvoa. Viitattu 28.12.2011.
http://www.ketteratkaytannot.fi/fi-FI/Ketteryys/VaatimuksiaVaiLisaarvoa

Poimala, S. & Heikniemi, J. & Blåfield, H. 2009b. Ketterät käytännöt. Menetelmät. XP. Extreme Programming. Viitattu 29.12.2011.
http://www.ketteratkaytannot.fi/fi-FI/Menetelmat/XP/

Poimala, S. & Tolvanen, P. 2011a. Ketteryys haltuun: Yleisimmät ketterät menetelmät. Sininen meteoriitti. Ketteryys haltuun 2 (3). Viitattu 27.12.2011.
http://www.meteoriitti.com/fi-FI/tiedotteet/ajankohtaista/ketteryys-haltuun-yleisimmat-ketterat-kaytannot

Poimala, S. & Tolvanen, P. 2011b. Ketteryys haltuun: Scrum pähkinänkuoressa. Sininen meteoriitti. Ketteryys haltuun 3 (3). Viitattu 28.12.2011.
http://www.meteoriitti.com/fi-FI/tiedotteet/ajankohtaista/ketteryys-haltuun-scrum-pahkinankuoressa

SAP AG 2010. Viitattu 3.4.2012
http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/com.sap.km.cm.docs/lw/asap%20methodology/asap%20methodology%20for%20introduction/Index.htm

Schwaber, K. & Sutherland, J. 2009. Scrum Guide. Viitattu 28.12.2011.
http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20FI.pdf#view=fit

Winston W. Royce 1970. Managing the Development of Large Software Systems,_Technical Papers of Western Electronic Show and Convention (WesCon), Los Angeles, USA, 25,28(1970).

YIT Corporation 2011. Viitattu 31.3.2012
http://www.yit.fi/palvelut/yritysinformaatio/perustietoa

26.1.2012 Haastattelu: Projektipäällikkö, YIS
26.1.2012 Haastattelu: Kehityspäällikkö, YIS
8.2.2012 Haastattelu: MDM SC –sovelluksen edustaja
9.2.2012 Haastattelu: MD Team Leader, YIS
9.2.2012 Haastattelu: IT Manager, YIS

LIITE1

Opinnäytetyön haastattelun kysymykset ja vastaukset

- Millaisia sovelluksia olette ottaneet käyttöön?

Projektipäällikkö (YIS):
SAP ECP –kustannuslaskentatyökalu Easy Cost Planning
Estimator –kustannuslaskentasovellus, työasemasovellus, joka käyttää palvelimella olevaa tietokantaa
PLOT Map –Web-pohjainen Cognos-sovellus maanhallinnan karttailmentymä, strateginen johdon työkalu, jolla suunniteltiin maankäyttöä vuosia ja vuosikymmeniä eteenpäin.

Kehityspäällikkö (YIS):
Paljon pieniä sovelluksia (ulkopuolisia toimittajia)
Notes –sovellukset (paljon pieniä sovelluksia, YIS:llä itse tehtyjä)
Jaska (YIS:n ja YIT Kaluston käyttämä toiminnanohjausjärjestelmä)

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Data/ tietosisältö työkalut:
Tiedonkeruujärjestelmä
Massamuokkaustyökalu

IT Manager (YIS):
SAP –käyttöönottoja YIS:llä
Navision –template määrittely käynnissä
Web –kaikkien ulkoisten sivujen uusiminen, uusi platform –teknologia
Konsoliratkaisu

MD Team Leader (YIS):
SAP –käyttöönottoja, suoria 10 (itse mukana) ja epäsuoria 5 (taustalla valvomassa)
D&B (Dunns & Bradstreet)
MDM SC:n kartoitus ja määrittely / ei itse käyttöönotto

- Kuinka monta henkilöä käyttöönottoprosesseissa on ollut mukana? Kuinka paljon loppukäyttäjiä?

Projektipäällikkö (YIS):
ECP: (liiketoiminta 6-10), konsultteja 1, SAP Developer 3, muut toimittajat 5, reilu 100 käyttäjää
Estimator: käyttäjiä reilu 500, YIS 2 henkilöä, toimittajalta 2, liiketoiminnasta 12.
PLOT Map: reilu 30 käyttäjää, liiketoiminnasta 8, YIS 2, toimittajilta 3

Kehityspäällikkö (YIS):
Käyttäjiä mukana 1-200 sovelluksesta riippuen.
Esim. Projekteissa mukana 2-3, 10-20

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
1-2 nimettyä henkilöä per projekti (tarvittaessa mukana enemmänkin)
3-30 käyttäjää per sovellus

IT Manager (YIS):
10-50 henkilöä projekteissa mukana, loppukäyttäjiä 10-2000

MD Team Leader (YIS):
SAP –käyttöönotoissa (projektiryhmä 40-50), MasterData 5-8 henk. (1000 käyttäjää)
D&B –käyttöönotossa 10 template määrittelyissä, projektiryhmässä 4 henk. (käyttäjiä 20-30)

- Kuinka kauan IT-projektit suunniteltiin kestäväksi ja kuinka kauan ne ovat keskimäärin kestäneet?

Projektipäällikkö (YIS):
ECP: Suunniteltu 6 kk – toteutu 15 kk
Estimator: Suunniteltu 6 kk – toteutu 12 kk
PLOT Map: Suunniteltu 4 kk – toteutu 5 kk

Kehityspäällikkö (YIS):
Pienissä Notes –sovelluksissa 2-3 päivää
Jaska Arvioitu 3 kk ja toteuma 3 kk
Muissa sovelluksissa esim. Arvioitu 3 kk ja toteuma 6 kk

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Prosessit 2 vko – 3 kk, yleisesti ottaen aikatauluissa on pysytty

IT Manager (YIS):
SAP –käyttöönotoissa suunniteltu aika 9 kk, toteuma ollut 15 kk
Hyberian 9 kk ja toteuma 9 kk
Navision 12 kk

MD Team Leader (YIS):
SAP –FICO –käyttöönotto 4-6 kk ja myös toteutui (valmis template, joten sovellus oli 90% valmis)
2. vaiheen operatiivisien prosessien käyttöönotto 1 vuosi. Muutaman kuukauden myöhästymisiä.
D&B suunniteltu 1 vuosi ja toteutui 1,5 vuotta

- Onko edellä mainituissa käyttöönotoissa ollut selkeästi joku prosessimalli käytössä?

Projektipäällikkö (YIS):
Kaikissa ollut käytössä.

Kehtiyspäällikkö (YIS):
On aina ollut tietty tapa toimia. Tekovaiheessa ei ole päätetty, että käytetään jotain tiettyä mallia.

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
On ollut käytössä.

IT Manager (YIS):
On

MD Team Leader (YIS):
On käytetty ASAP –mallia

- Millaisia sovelluksen käyttöönottoprosessien malleja on ollut käytössä?

Projektipäällikkö (YIS):
Kulloinkin IT-projektien mukainen projektimalli, jotka ovat muuttuneet ajan saatossa. (Prosessin kuvaus löytyy In Touch:sta)

Kehityspäällikkö (YIS):
Esim. Scrum, Vesiputousmalli jne.

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Steps –projektimalli (Suomen projekti-instituutin projektimalli)

IT Manager (YIS):
Vesiputous-malli (iteroiva)
Scrum
ASAP –vesiputousmallista kehitetty SAP:n oma käyttöönottomalli

MD Team Leader (YIS):
ASAP, 2. Vaiheen Template määrittelyvaihe tehtiin Protomalli

- Mitä vaiheita käyttöönottoprojektit sisältävät?

Projektipäällikkö (YIS):
(In Touch, G1, G2..jne. G6?)

Kehityspäällikkö (YIS):
Vesiputousmalli pohjana.

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Taru toimittaa vaiheistusmallin (powerpoint)

IT Manager (YIS):
Mallien lisäksi projektiprosessi kuvattuna. Noudattelee vesiputousmallin elementtejä vaiheineen. Vaiheet kuvattuna.

1. Project Preparation

   <= Pre-Study
   Preparation
   Planning
   Execution and Testing
   Implemantaion
   Closing
2. Business Blueprint (GAP- analyse)
3. Realisation (Configuration)
4. Final Preparation & Go-Live

MD Team Leader (YIS):

1. Määrittely (Template –vaihe)
2. Templaten pilotointi (päätös pilotoinnissa) => RoadMap
3. Käyttäjäkoulutukset (Projektiryhmä)
4. GAP –analyysi (Template vs. lokaalit prosessit)
5. Konfigurointi (lokalisaatiovaihe)
6. Testausvaihe
   a. Yksikkö
   b. Integraatio
   c. Hyväksyntä => Käyttöönottopäätös
7. CutOver
8. Go-Live

- Onko jokaisen vaiheen lopussa jokin tietty tuotos ja/tai katselmointi?

Projektipäällikkö (YIS):
Edellytys seuraavaan vaiheeseen siirtymiselle on aina ollut katselmuksessa vaiheen läpikäynti ja että se on todettu suoritetuksi.

Kehityspäällikkö (YIS):
Osassa selkeä tuotos ja sen katselmointi. Joissain määrittelyt ovat muuttuneet toteutuksen aikana. Nykyään pyritään toteuttamaan YIS:n käytettäviä projektimalleja.

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Projektin koosta riippuva ..joissakin tapauksissa ei pysähdytä ollenkaan ja joissakin johtoryhmä kokoontuu aina tekemään päätöksiä. Projektisuunnitelma tulee aina olla kunnossa ennen kuin seuraavaa jatketaan.

IT Manager (YIS):
Jokaisen vaiheen jälkeen dokumentoidaan tehdyt asiat (tarvitaan aina hyväksyntä) ja suunnitellaan tulevaa vaihetta

MD Team Leader (YIS):
Jokaisen vaiheen jälkeen pitää olla valmiina tuotokset/hyväksyntä eikä seuraavaan vaiheeseen siirrytä ennen kuin edellinen on valmis. Voidaan kuitenkin mennä limittäin.

- Kuinka kirjaimellisesti niiden eri vaiheita on noudatettu ja toteutettu?

Projektipäällikkö (YIS):
Vaiheistusta seurataan dokumenttien perusteella ohjausryhmässä.

Kehityspäällikkö (YIS):
Vaiheita toteutetaan joustavasti riippuen projektin koosta. (Projektin suunnitellulla kestolla ja budjetille merkitystä)

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Vaiheistusta noudatetaan yleisesti ottaen vain isommissa projekteissa ja riippuu paljon asiakkaan vaatimuksista vaiheiden noudattamisen suhteen. Projektisuunnitelma valmiina ennen kuin edetään.

IT Manager (YIS):
Prosessimallit antavat suuntaviivat ja niistä poiketaan aika paljon. Aina sovelletaan sen mukaan, mikä on asiakkaan tahtotila ja kyky toimia. Käytetään mm. Kunkin vaiheen ehdollista hyväksyntää ja asioihin palataan myöhemmin. Steering mukana päätöksenteossa.

MD Team Leader (YIS):
Välillä on tullut eri vaiheiden välillä liukumaa eri ja näille on aina saatu kaikkien osallisten hyväksyntä.

- Voidaanko käyttöönottoprosessien vaiheissa palata taaksepäin suorittamaan kertaalleen jo lopetettua vaihetta uudestaan tai siirtyä jopa aikaisempaan vaiheeseen uudestaan?

Projektipäällikkö (YIS):
Muutoksia voi tulla matkan varrella ja sen mukaan reagoidaan sen mukaisesti.

Kehityspäällikkö (YIS):
Mitä pienemmästä toteutuksesta on kysymys, sen helpompi on palata projektissa taaksepäin.

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Voidaan palata aina edellisiin vaiheisiin tarpeen mukaan. Esim. Smartin kanssa palattiin määrittelyvaiheeseen uusien kenttien suhteen.

IT Manager (YIS):
Ei oikeastaan, koska vaiheessa kesken jääneet asiat siirtyvät seuraavaan vaiheeseen tai sovitaan niiden tekemisestä tulevaisuudessa.

MD Team Leader (YIS):
Ei voida pääsääntöisesti palata takaisinpäin. Ainoastaan kriittiset jutut korjataan ja loput sitten jätetään tulevaisuudessa kehitettäväksi.

- Kuinka helposti asiakkailta saadut vaatimukset on saatu toteutettua?

Projektipäällikkö (YIS): Asiakkaiden ilmaisevat vaatimukset on aina haasteellisia pukea tavoitteiksi. Vaatii usein pitkiä jatkokeskusteluja ja asioiden konkretisoimista käytännön esimerkein.

Kehityspäällikkö (YIS):
Joskus on tullut sellaisia vaatimuksia, joita ei ole pystytty toteuttamaan sen hetkisillä toteutusmahdollisuuksilla.

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Kaikkia ei millään pysty toteuttamaan. Mukana tietysti sellaisiakin, jotka kyllä pystyttäisiin toteuttamaan, mutta tulee jo lähtökohtaisesti suhteettoman kalliiksi. Osa pystytään toimittamaan vaatimusten mukaisesti ja jotkut vaatimukset ovat jopa helppoja toteuttaa.

IT Manager (YIS):
Kaikissa projekteissa on ollut huomattavia vaikeuksia. Todellinen business ja IT –osaaminen puuttuu. Eli projekteissa ei ole sellaisia ihmisiä, jotka ymmärtäisivät kokonaisuuden kummastakin puolesta.

MD Team Leader (YIS):
Riippuen aina kuinka hyvin määrittelyvaihe on tehty. Jos määrittelyä ei ole tehty tarpeeksi yksityiskohtaisesti, tulee aina ongelmia käyttöönottoissa. Voi tulla joko järjestelmän puolesta tai Templaten puolesta mahdottomia vaatimuksia joita ei pystytä toteuttamaan.

- Kuinka paljon projektien aikana on tullut muutoksia määrittelyihin?

Projektipäällikkö (YIS):
Aina tulee jotain muutoksia. Liian tiukka aikataulu aiheuttaa enemmän muutostarpeita kuin hyvin suunniteltu ja esiselvitelty projekti. Ensin kokonaiskuva ja sitten tarkemmat määrittelyt.

Kehityspäällikkö (YIS):
Vaihtelevasti joskus muutospyyntöjä on tullut kesken projektien ja joskus ei sitten ollenkaan.

Palvelupäälliikkö (case-sovelluksen toimittajan edustaja):
Ei yleensä tule isoja muutoksia, mutta testausvaiheessa ilmene joskus muutostarpeita. Muutostarpeet tulevat yleensä vasta varsinaisen projektin jälkeen ja ovat tällöin lisä- /kehitysprojekteja.

IT Manager (YIS):
Taloudellisesti on tullut 10-30 %:n verran muutoksia. Ei paljoa toiminnallisia muutoksia. Raportointi on hyvä esimerkki sellaisesta, mihin tulee aina matkan varrella paljon muutospyyntöjä.

MD Team Leader (YIS):
Osaaminen karttuu aina projektin myötä ja sen takia muutosvaatimuksia yleensä tulee enemmän projektin loppupuolella. Eli enemmän ja enemmän tulee, mitä pitemmälle mennään.

- Kuinka ketterästi muutoksiin on pystytty vastaamaan?

Projektipäällikkö (YIS):
Sitä vaikeampaa kuin mitä myöhempään muutostarve tulee. Poikii kerrannaisvaikutuksia sitä enemmän, mitä myöhempään ne tulee ilmi. Vasta testauksessa havaitut isot muutostarpeet ovat isoja haasteita projekteille.

Kehityspäällikkö (YIS):
Sitä helpompaa, mitä aikaisemmin muutostarve on tullut esiin. Oleellista on projektin koko ja missä vaiheessa ollaan menossa.

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Osaan pystytään vastaamaan ketterästi, mutta totta kai riippuu paljon itse vaatimuksesta ja työkuormista / resursseista. Kuinka paljon työvaiheita vaatimukset sisältää ja millä tasolla dokumentoinnin pitää olla.

IT Manager (YIS):
Suuremmissa projekteissa (Template –tason projektit) ei ole mahdollista toteuttaa nopeita muutoksia ja kaikki muutokset täytyy käyttää aina tietyn hyväksyntäkaavan kautta.

MD Team Leader (YIS):
Lokalisointivaiheessa muutoksiin pystytään vastaamaan paremmin kuin vasta testausvaiheessa tulleet muutokset.

- Mitä kehitettävää te näette käytössä olevissa käyttöönottoprosessimalleissa?

Projektipäällikkö (YIS):
Asiantuntijoilla niin IT:ssä kuin liiketoiminnan puolella tulee olla kattavampi tietotaito kunkin projektin kokonaisuudesta. Lisäksi tarvitaan enemmän erikois-asiantuntijoita (liiketoiminta) ja konsultteja (markkinoilla oleva ajantasainen tietämys)

Kehityspäällikkö (YIS):
Määrittelyvaiheessa pitäisi olla mahdollisimman hyvä kartoitus asiakkaan puolella niin kuin YIS:nkin puolella ja syvempää yhteistyötä. Vuoropuhelua asiantuntijoiden ja asiakkaiden välissä tulee kehittää. Lähtötilanteessa tulee olla selvänä projektin sisältö (Scope) ja tavoitteen ja kaikilla projektin osapuolilla on yhtenevä näkemys siitä. Projektin sisältöön kannattaa lisätä myös mahdolliset rajaukset.

Palvelupäällikkö (case-sovelluksen toimittajan edustaja):
Projektimalliin ei itsessään tarvitse tehdä muutoksia. Määrittelyitä pitäisi arvostaa paljon enemmän – määrittelyt tarkemmalle ja yksityiskohtaisemmalle tasolle. Aikataulutukseen tulisi myös kiinnittää enemmän huomiota. Pitää huomioida tarkemmin toimijoiden määrä.

IT Manager (YIS):
Itse prosesseissa ei ole paljoa kehitettävää vaan lähinnä sen jalkauttamisessa sen toteuttajille (liiketoiminta & YIS). Prosessista puuttuu laadullinen seuranta. Laadullisesti tuotos olisi hyvä ja tasalaatuinen. Samalla voitaisiin helpommin tunnistaa projektien riskit ja siihen liittyvät asiat.

MD Team Leader (YIS):
ASAP –malli sopii parhaiten tapauksiin, joissa käyttöönotettava prosessi ei muuta paljon olemassa olevaa prosessimallia. ASAP –mallin käyttöönotto vaatii siis vahvan Template –mallin, jonka päälle sovelluksen rakentaminen on helppoa. Sinällään se on aika jäykkä malli – ainakin tapauksissa, joissa itse sovelluksen takia joudutaan muokkaamaan käytettäviä prosesseja. Haasteellista siinä on, että osaaminen uudesta sovelluksesta tulee vasta myöhäisemmässä vaiheessa ja sen takia palaaminen käyttöönoton edellisiin vaihesiin vaikeutuu. Käytännön järjestelmätasolle mennään ehkä liian myöhään ja ensimmäisissä vaiheissa käsitellään asioita pintapuolisesti tyyliin PowerPointilla esittelemällä. Eli sovelluksen varsinainen toiminnallisuus tulee käyttäjille liian myöhään selväksi ja sen takia muutosvaatimukset voivat tulla liian myöhään käyttöönoton kannalta.