Opinnäytetyö (AMK)

Liiketalous

Bioalat ja liiketalous

2015

Tuomas Sinervo

MIKSI KETTERÄÄ KEHITYSTÄ OHJELMISTOPROJEKTEISSA?

–suomalainen vakuutusyhtiö

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Yrittäjyys ja liiketoimintaosaaminen | Liiketalous, ICT ja kemiantekniikka

2015 | 81 + 5

Markku Heikkilä

Tuomas Sinervo

MIKSI KETTERÄÄ KEHITYSTÄ OHJELMISTOPROJEKTEISSA? -SUOMALAINEN VAKUUTUSYHTIÖ

Ohjelmistoprojektit muodostavat yhä tärkeämmän kokonaisuuden yritysten kilpailukyvyn kehittämisessä ja ylläpitämisessä. Nopea markkinoille pääsy vaatii tehokkaita ja joustavia menetelmiä sovellusten kehittämiseen, sekä nopeaa reagointia asiakkaan toiveisiin. Ketterät sovelluskehitysmenetelmät pyrkivät tarjoamaan tähän mahdollisuuden.

Tämän tutkimuksen tavoitteena oli selvittää, miten suomalainen vakuutusyhtiö on onnistunut siirtymään perinteisistä sovelluskehitysmenetelmistä ketteriin sovelluskehitysmenetelmiin ja miten yrityksen työntekijät kokevat muutoksen verrattuna vanhaan kehitystapaan. Tutkimuksen teoriaosa käsittelee perinteisten- ja ketterien sovelluskehitysmenetelmien periaatteita sekä niiden prosesseja. Varsinainen tutkimus toteutettiin puolistrukturoituina haastatteluina vuosina 2014 - 2015 ja haastateltaviksi valittiin 13 kappaletta yrityksen sovellusprojekteissa työskenteleviä henkilöitä toteuttajista johtohenkilöihin.

Tutkimustulokset raportoitiin ryhmittämällä haastatteluiden tulokset tietojärjestelmäprojekteille tyypillisiin teemoihin tai sisältökategorioihin, kuten aikataulutus, resursointi, sisällön- ja versioiden hallinta, dokumentointi, testaus ja jatkoseuranta. Tutkimuksessa havaittiin, että yritys on onnistunut kohtalaisesti ketterien kehitysmenetelmien käyttöönotossa, ottaen käyttöön ketteristä kehitysmenetelmistä sellaiset prosessit, jotka soveltuvat parhaiten yrityksen järjestelmien kokonaiskehitysprosessiin. Prosessin ylläpidossa on kuitenkin haasteita, johtuen esimerkiksi yrityksen järjestelmien kompleksisuudesta, kerrostuneisuudesta ja rajapintojen määrästä.

ASIASANAT:

Ketterät menetelmät, vakuutus, sovellusprosessit, sovelluskehittimet

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business | Business ICT ja Life Sciences

2015 | 81 + 5

Markku Heikkilä

Tuomas Sinervo

WHY TO USE AGILE DEVELOPMENT IN SOFTWARE PROJECTS? - A FINNISH INSURANCE COMPANY

Sofware projects hold an even more important role in the development and maintenance of competitiveness in companies. Fast time to market requires efficient and flexible processes for software development as well as quick reaction to customer needs. Agile methods are hailed as an answer to these. The aim of this research is to find out how a Finnish insurance company has managed to change its process from traditional software development to agile processes and how the employees of the company view the change compared to the old way of developing software.

The theory part of the research addresses the priciples and processes of both tradtional and agile development methods. The actual study was conducted through interviews between 2014 and 2015. 13 people were selected form various posts within the organisation working with software projects.

The results were reported by grouping the interview subjects into common software project category themes like timetable, resources, content and version management, documentation, testing and follow up. In the study it was found out that the company had reasonably successfully transitioned form traditional methods to agile methods by deploying parts of the agile methodologies most suitable for the overall processes of the company though some challeneges were found in the upholding of the process caused for example by the layered sturcture of the system, lack of time or interface complexity.

KEYWORDS:

Agile, insurance, software development, iterative, scrum, software processes

SISÄLTÖ

KÄYTETYT LYHENTEET (TAI SANASTO) 6

1 JOHDANTO 8

1.1 Tutkimusongelman asettelu 8

1.2 Projektityöskentely 9

2 OHJELMISTOPROJEKTIT KÄYTÄNNÖSSÄ 11

2.1 Vesiputousmalli 15

2.2 Vesiputousmalli käytännössä 20

2.3 Kohti ketteriä kehitysmenetelmiä 23

3 KETTERÄT KEHITYSMENETELMÄT 25

3.1 Pois noudatetusta suunnitelmasta ja rautakolmiosta 25

3.2 Ketterät menetelmät ja kommunikointi 27

3.3 Nopeammin toteutuksesta tuotantoon 29

3.4 Scrum 32

3.5 Scrum -prosessi 35

3.6 Haasteita ja väärinymmärryksiä Scrum -työskentelytavan käyttämisessä 37

4 HAASTATTELUJEN JA TUTKIMUKSEN TOTEUTUS 44

4.1 Kysymyskategoriat 46

4.2 Haastateltavien valinta 46

4.3 Aineiston keruu 47

4.4 Tutkijan oma asema ja sen vaikutuksen tiedostaminen 48

4.5 Aineistoanalyysi 48

5 TUTKIMUKSEN TULOKSET 51

5.1 Johdanto 51

5.2 Vanhan kehitystavan käyttökokemukset 51

5.2.1 Vanha kehitysmalli ja aikataulutus 51

5.2.2 Järjestelmäversioiden sisällönhallinta 53

5.2.3 Vaatimusten määrittely ja läpikäynti 55

5.2.4 Vesiputousmallin projektisuunnittelu ja työn etenemisen seuranta 56

5.2.5 Järjestelmien testaus 58

5.2.6 Toteutuksen resursointi ja henkilöriskit 59

5.2.7 Projektien dokumentointi ja jälkiseuranta 62
5.3 Uuden kehitystavan käyttökokemukset 63
5.3.1 Uuden kehitystavan käyttöönotto 63
5.3.2 Uusi kehitysmenetelmä ja aikataulutus 65
5.3.3 Järjestelmäversioiden sisällönhallinta 66
5.3.4 Toteutuksen resursointi ja henkilöriskit 67
5.3.5 Ketterän mallin projektisuunnittelu ja etenemisen seuranta 69
5.3.6 Projektien dokumentointi, järjestelemävaatimukset ja jälkiseuranta 71

6 JOHTOPÄÄTÖKSET JA SUOSITUKSET 75

LÄHTEET 80

LIITTEET

Liite 1. Kysymykset
Liite 2. Haastattelut

KUVAT

Kuva 1. Kustannushyötyanalyysin konteksti (Remenyi ja SherwoodSmith 1999, 16) 11
Kuva 2. Vesiputousmalli (Leffingwell 2011, 5) 15
Kuva 3. Rautakolmio (Leffingwell 2011, 7) 19
Kuva 4. Iteratiivinen vesiputousmalli (Mall 2013, 41) 23
Kuva 5. Vesiputous ja ketterä menetelmä (Leffingwell 2011, 16) 26
Kuva 6. Toimenpide suunnitteluprosessi (Cooke 2010, 155) 31
Kuva 7. Scrumilla toteutettu projekti 33
Kuva 8. Scrumilla toteutettu projekti, yhden pyrähdyksen sisältö 35
Kuva 9. Seinä 36
Kuva 10. Laadullisen tutkimuksen prosessikaavio (Kananen 2012, 93) 49

TAULUKOT

Taulukko 1. Johtopäätökset tiivistettynä 78

KÄYTETYT LYHENTEET (TAI SANASTO)

Kustannushyötyanalyysi	Kustannushyötyanalyysien tarkoituksena on tarjota tietoa siitä, kuinka lupaavia yrityksesi investoinnit tai projektit ovat. Analyysissä yhden vaihtoehdon kustannuksia verrataan sen oletettuun hyödyllisyyteen. (http://st.merig.eu/?id=286&L=2)
Tuotteen kehityslista	Lista priorisoiduista ominaisuuksista, joka sisältää lyhyet kuvaukset toiminnallisuuksista, jotka halutaan (https://www.mountaingoatsoftware.com/agile/scrum/product-backlog)
Työmääräarvio	Määrä työtä, jonka tietty itsenäinen kokonaisuus saa tehdyksi tiettynä ajanjaksona, tai keskimääräinen määrä työtä, jonka tietty itsenäinen kokonaisuus pystyy tekemään tietyssä ajassa. (https://www.techopedia.com/definition/13544/workload)
Ohjemointirajapinta	Määritelmä, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään. (http://apisuomi.fi/sanasto/ohjelmointirajapinta/)
Vesiputousmalli	suoraviivainen projektinhallinnan malli, jossa edetään vaiheittain projektin määrittelyn kautta sen suunnitteluun, toteutukseen, testaukseen ja toimitukseen. (http://4dsoftware.fi/projektinhallinnan-vesiputousmenetelma-vs-ketterat-menetelmat/)
Ketterät kehitysmallit	projektinhallinnan malli, jossa projekti jaetaan pienempiin osiin (moduuleihin) ja aina moduulin valmistuttua se testataan ja projektisuunnitelmaa uudelleen arvioidaan. (http://4dsoftware.fi/projektinhallinnan-vesiputousmenetelma-vs-ketterat-menetelmat/))
Iteratiivinen	prosessi, jossa tiettyä ohjekokonaisuutta toistetaan tietty määrä kertoja tietyn lopputuloksen saavuttamiseksi. (http://www.thefreedictionary.com/iterative)

Pyrähdys	Tuotekehityksessä pyrähdys on ennalta määritelty ajanjakso, jonka aikana tietty työ pitää valmistua tarkastelua varten. (http://searchsoftwarequality.techtarget.com /definition/Scrum-sprint)
Scrum Master	Scrum Master on Scrumia käyttävän tuotekehitystiimin työn mahdollistaja. (http://whatis.techtarget.com/definition /scrum-master)
Tuoteomistaja	Tuoteomistaja on Scrum -kehitysrooli henkilölle, joka edustaa liiketoiminta tai käyttäjäkuntaa, ja on vastuussa käyttäjäkunnan kanssa siitä, mitä ominaisuuksia tuotteeseen tulee sen käyttöönotossa. (http://searchsoftwarequality.techtarget.com/ definition/product-owner)
Pyrähdyksen tarkastelu	Pyrähdyksen lopussa toteutustiimi esittelee valmistuneen työn tuoteomistajalle ja tuote omistaja käyttää kriteerejä, jotka on päätetty pyrähdyksen suunnittelukokouksessa työn hyväksymiseen tai hylkäämiseen. (http://searchsoftwarequality.techtarget.com /definition/Scrum-sprint)
Sovellusvaatimus	Toteamus asiakkaan toiveesta, päämäärästä, ehdosta tai tuotteen kyvystä, joka tuotteen pitää täyttää tyydyttääkseen tarpeen tai tavoitteen. (http://rmblog.accompa.com/2012/04/ software-requirements-definition/)
Tarina Scrum -kehityksessä	Pyrähdyksen aikana toteutettavat toiminnot (http://www.ti-es.com/Sovelluskehitys.aspx)

1 JOHDANTO

1.1 Tutkimusongelman asettelu

Tämän tutkimuksen tarkoituksena on lisätä ymmärrystä siitä, miten ketterään kehitysmalliin siirtyminen on vaikuttanut tutkimani yrityksen projektityöskentelyyn sen koko elinkaaren aikana ja miten uuden kehitystavan käyttöönotto on onnistunut. Tämän lisäksi pyrin hahmottamaan miten yrityksessä koetaan aiemmin käytössä ollut vesiputousmallinen kehittäminen suhteessa uuteen ketterään kehitysmalliin.

Tämä saavutetaan vastaamalla tutkimusongelmaan: Miksi käyttää ketterää kehitystä projektityössä?

Opinnäytetyössäni tutkin suomalaisen vakuutusyhtiön sovellusprojektien siirtymää perinteisistä kehitysmenetelmistä ketteriin menetelmiin. Tutkimani yritys on siirtynyt vanhoista kehitysmenetelmistä uusiin ketteriin menetelmiin vuoden 2013 loppupuolella ja nyt sen sisällä oli kiinnostusta tutkia sitä, miten menetelmien käyttöönotto oli onnistunut ja millä tavoin ketterät menetelmät sopivat tutkimani yrityksen projektinhallinnan kokonaisprosesseihin.

Työskentelen yrityksen liiketoiminnan puolella sovellusasiantuntijana, missä toimenkuvaani kuuluu järjestelmämäärittelyiden tekeminen ja sovellusprojekteissa työskentely. Tärkeimmät rajapinnat työni kannalta ovat yrityksen IT -osasto, tuoteyksikkö ja jakelu.

Yrityksellä on oma IT -osasto, joka on sen historian aikana ollut sekä toimittajan roolissa, että osana yritystä vaihtelevasti. Nykyään IT -osaston rooli on vakiintunut ja se on osa yritystä. Tutkimuksen tuloksia, johtopäätöksiä ja suosituksia tulkitessa onkin hyvä huomioida, että ne eivät ole suoraan verrattavissa tilanteeseen, jossa yrityksen IT -osasto on ulkoistettu ulkopuoliselle toimittajalle vaan tutkimus kuvaa yhtenäisen yrityksen henkilöstön näkemyksiä tapahtuneesta prosessin muutoksesta.

Tutkimusongelmaan pyrin vastaamaan haastattelemalla yrityksen henkilöstöä eri työtehtävissä saadakseni kokonaiskäsityksen siitä, miten ketterä kehitysmalli toimii tällä hetkellä yrityksessä, miten sen käyttö on mahdollistettu, millaisia haasteita on kohdattu, ja millaisia muutoksia uuden mallin käyttöönotto on yrityksessä saanut aikaan projektityöskentelyssä.

1.2 Projektityöskentely

Projekti on käsitteenä jokaisen suomalaisen yrityksen työntekijälle tuttu käsite. Projektit ja projektiluontoinen työskentely on nopeatempoisessa ja yritysten tiukassa resurssien hallinnassa tehokas ja usein ainoa tapa toteuttaa asioita tiukassa aikataulussa. Yrityksillä on usein oma projektinhallintaprosessi, jossa on määritelty eri vastuualueet sekä projektin tilaajalle, toimittajalle että projektipäällikölle ja projektiryhmälle. Johdon tehtävänä on allokoida projekteihin resursseja ja mahdollistaa projektin sujuminen mahdollisimman sulavasti sekä seurata ja ohjata projektin etenemistä erilaisten ohjausryhmien kautta, joihin osallistuvat sellaiset henkilöt, joilla on jokin panos projektin onnistumiseen. Projektityöskentelyn merkitys kasvaa koko ajan, koska yrityksillä on koko ajan kovempi paine saada tehtyä useita kehityshankkeita samanaikaisesti ja vähenevillä resursseilla.

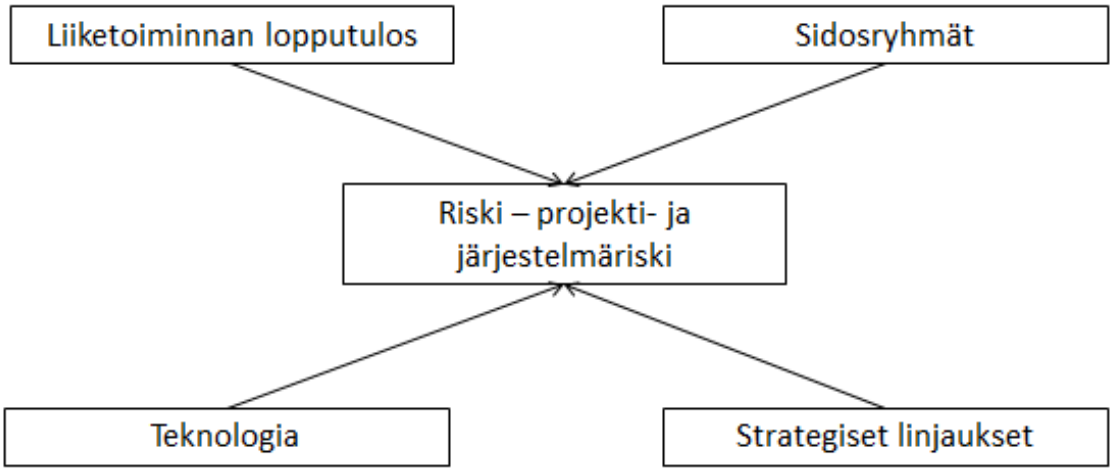
Projektimallien tarkoitus on antaa yrityksille ja organisaatioille suuntaa-antavia ohjeita ja toimintamalleja projektien läpivientiin ja määritellä erilaisia onnistumisen kriteereitä, joiden kautta pystytään mittamaan projektin onnistumista. Erilaisia projektimalleja on useita, mutta niille ominaista on tavoitteen tarkka määrittely sekä tavoitteen saavuttamiseen käytettävän ajan rajallisuus. Projekti saattaa yksinkertaisimmillaan olla vain aikajana, jonka alussa määritellään alkutilanne ja janan päätepisteessä sijaitsee määritelty lopputulos. Tätä kutsutaan projektin elinkaareksi. Usein elinkaaren määrittely ei kuitenkaan ole niin yksinkertaista, vaan projektijanalla on useita osamaaleja, joiden saavuttaminen on lopputuloksen ja projektin juoksutuksen kannalta oleellista.

Projektipäällikkö määritellään usein projektin toimitusjohtajaksi ja hänen roolinsa on kriittinen projektin onnistumisen kannalta. Projektipäällikkö vastaanottaa projektin tilaajalta tehtävänannon ja tämän jälkeen pyrkii kokoamaan ympärilleen tiimin, jonka avulla tehtävänanto pystytään toteuttamaan ja projektin tavoite saavutetaan. Projektipäällikön päivittäisiin tehtäviin kuuluu mm. projektin etenemisen seuraaminen ja kokonaisarvion luominen siitä, onko projekti aikataulussa ja oikeassa kurssissa.

2 OHJELMISTOPROJEKTIT KÄYTÄNNÖSSÄ

Ohjelmistoprojektit eivät eroa tavanomaisista projekteista erityisesti. Ohjelmistoprojekteissa pyritään myös määrittelemään lähtöpiste ja haluttu lopputulos, jonka saavuttamiseksi järjestelmiin pyritään rakentamaan erilaisia toimintoja, joiden kautta lopputuloksena on toiminnallisuus tai tuote, joka vastaa määriteltyä lopputulosta tai toimintoa, jolla lopputulos saavutetaan.

Ohjelmistoprojektiprosessin, kuten muidenkin projektien, alkupisteenä toimii yleensä kustannushyötyanalyysin laatiminen. Remenyi ja SherwoodSmith määrittelevät kirjassaan IT Investment Making a Business Case kustannushyötyanalyysin seuraavanlaisesti.



Kuva 1. Kustannushyötyanalyysin konteksti (Remenyi ja SherwoodSmith 1999, 16)

Heidän mukaansa onnistuneen kustannushyötyanalyysin tekeminen edellyttää liiketoiminnan lopputuloksen, sidosryhmien, teknologian sekä strategisten linjausten huomioon ottamista. Näiden kaikkien osa-alueiden yhteen laskemisen tuloksena saadaan lasketuksi projektiin ja teknologiaan kohdistuva riski. Yksinkertaistetusti laskelman tarkoituksena on arvioida projektista saatavaa hyötyä suhteessa kustannuksiin ja riskiin. Laskelma voi yksinkertaisimmillaan olla vain

yhden Power Point -dian kokoinen arvio siitä, millaisia liiketoiminnallisia hyötyjä projektista on yritykselle tai organisaatiolle saatavissa projektin valmistuttua. Kuten monissa muissa yrityksissä myös tutkimassani yrityksessä on kustannushyötyanalyysin tekeminen ensimmäinen edellytys projektin käynnistämiselle, koska se antaa ensimmäisen käsityksen siitä, onko projektilla edellytyksiä edetä päätäntävaiheeseen. Laskelmaa ei tulisikaan käsitellä vain pakollisena projektiprosessin osana, vaan se näyttelee merkittävää roolia projektin määrittelyssä ja tavoitteissa. Hyvin määritelty kustannushyötyanalyysi onkin onnistunut kuvaus projektista saavista hyödyistä. Ohjelmistoprojekteissa kustannushyötyanalyysillä pyritään antamaan projektien aikataulutuksesta vastaaville tahoille tarvittava informaatio projektin aloituspäätöstä varten.

Ohjelmistoprojektien tekeminen vaatii isoissa yrityksissä aikataulutusta ja resursointia. Aikataulutusta hallitaan yleensä tuotteen kehityslistalla, johon kerätään kaikki eri sovelluksia koskevat työt, jotka voivat olla sekä uusia liiketoiminnallisia vaatimuksia, lisäyksiä että muutoksia olemassa oleviin toimintoihin tai tuotekokonaisuuksiin. Stacia Viscardi kuvaa kirjassaan Professional ScrumMaster’s Handbook tuotteen kehityslistaa hyvin tuoteomistajan toivelistaksi. Se sisältää Viscardin mukaan kaiken ja mitä tahansa, mitä tilaaja saattaa haluta sovellukseensa (Viscardi 2013, 23). Usein työlistalla on oma osio myös selvitystä vaativille järjestelmävirheille, jotka ovat ilmenneet kehitysvaiheessa ja jotka tulisi korjata mahdollisimman pian. Tuotteen kehityslistan tärkein tehtävä on hallita sekä aikataulutusta että resursseja, joita käytetään töiden tekemiseen. Peter Saddington määrittelee tuotteen kehityslistan kaikeksi siksi työksi, jonka parissa toteutustiimi työskentelee tietyn projektin parissa ja se on koko ajan kasvava ja muuttuva jokaisen projektin osalta. (Saddington 2012, 37)

Tuotteen kehityslistan käyttäminen kehitystyövälineenä olisi hyödytöntä, mikäli sen avulla ei pystyttäisi arvioimaan eri töiden kokoa tai työn vaatimaa henkilöstöpanosta. Tämän vuoksi jokaiselle työlle pyritään löytämään työmääräarvio, joka kuvaa työhön vaadittavaa työpanosta. Yksinkertaisimmillaan lajittelu voi olla esimerkiksi pieni, keskikokoinen tai suuri työ. Työmääräarvioista on aina

vastuussa IT -toteuttaja, joka tekee arvion saamansa vaatimuksen tai virheen vakavuuden perusteella.

Tutkimassani yrityksessä kaikki eri sovellusalueiden työt on jaoteltu tuotteen kehityslistalle ja niille tulee hakea työmääräarvio IT-osastolta. Työn tilaajan tehtävänä on määritellä uudelle kehitystyölleen prioriteetti sen mukaan, miten välttämättömäksi työn tekeminen sovellukseen koetaan.

Töiden varsinainen aikataulutus eli päätös siitä, missä vaiheessa kyseinen työ viedään järjestelmiin, tapahtuu sovelluskehityksen ohjausryhmässä. Mikäli ohjausryhmässä ei päästä yksimielisyyteen siitä, mitkä kehitystyöt tulee tehdä ensimmäisenä, voidaan päätös vielä viedä ylempään päätöselimeen, jossa on edustettuna sekä tuoteosastojen, että myyntiosastojen ylempi johto.

Ohjelmistoprojektin varsinainen työ alkaa, kun vaatimusten määrittelyt on saatu tehdyksi ja IT -toteuttaja aloittaa muutoksen toteutuksen järjestelmään. Aiemmin ohjelmistoprojektin tekeminen on ollut staattinen ja suoraviivainen prosessi, joka alkoi määritysten tekemisellä ja päättyi testauksen kautta tuotantoon siirtoon. Prosessi oli selkeä ja helposti hallittavissa vanhoissa ja yksioikoisissa järjestelmissä, joissa toimintojen, järjestelmäkerrosten ja rajapintojen määrä oli pieni. Vanhemmista kehitysmenetelmistä suosituin ja useimmiten jossain muodossa käytetty oli niin sanottu vesiputousmalli. Vesiputousmalli aloitettiin aina ohjelman määrittelyllä, jota seurasi suunnitteluvaihe, jonka jälkeen ohjelma kehitetään ja testataan ennen kuin se otetaan käyttöön, eli annetaan asiakkaalle tai laitetaan myyntiin (Kasurinen 2013, 22). Vesiputousmalli toimii hyvin, mikäli toteutettava työ on suoraviivainen ja kaikilla siihen osallistuvilla tekijöillä on selkeä näkemys työn tekemisestä. Mikäli aikataulu alkaa venyä, tehtävät ominaisuudet eivät valmistu ajoissa ja virheet alkavat kasautua, törmätään ongelmiin. Vesiputousmalli ei anna toteuttajille mahdollisuutta reagoida kovin tehokkaasti muutoksiin aikataulussa. Kasurinen mainitsee kirjassaan myös vesiputousmallin ominaisuudesta ohjata työskentelyä. Hänen mukaansa lähestymistapa, jossa pitkällä projektilla on ainoastaan muutama isompi takaraja työvaiheen loppuun saattamiselle, johtaa helposti siihen, että tehdään turhaa työtä tai että työtahti

vaihtelee kesken projektin ollen keskellä työvaihetta löysä ja alkaen kiihtyä aina loppua kohden.

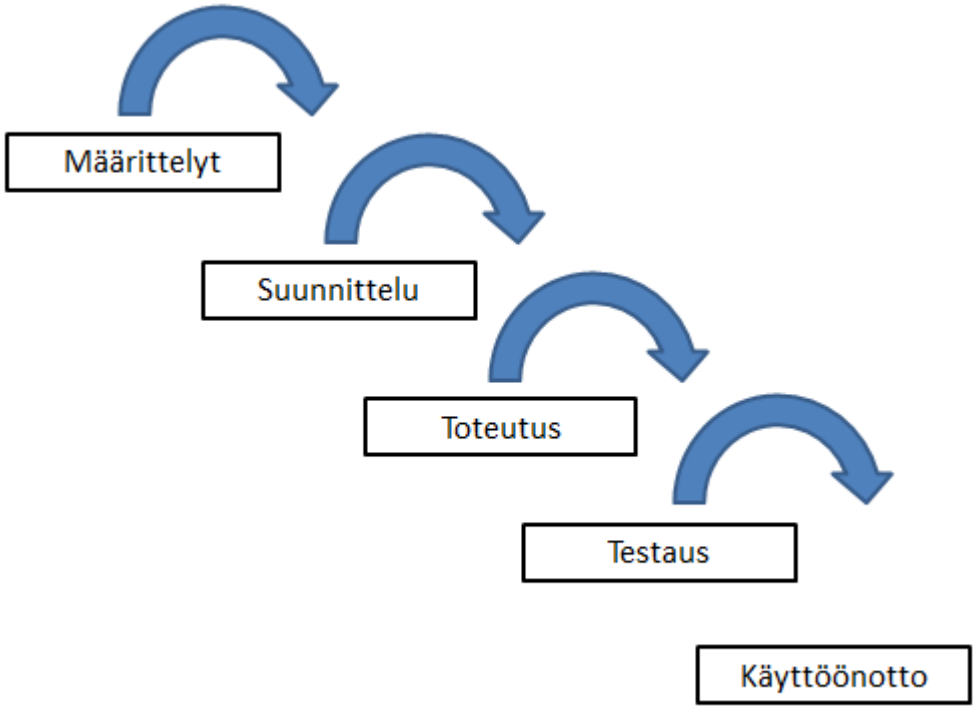
Vesiputousmalli on ollut käytössä laajasti myös tutkimassani yrityksessä ja tulenkin käsittelemään työskentelyä sen kanssa myöhemmissä kappaleissa, joissa pyrin vertailemaan haastattelujen kautta miten vesiputousmallin käyttö eroaa ketterien kehitysmallien käytöstä projektin läpiviennissä.

Toteutus ei nykyaikaisissa ohjelmistoprojekteissa ole staattinen prosessi, jonka aikana IT-yksikkö tai IT-toimittaja noudattaa suoraviivaisesti saamiaan järjestelmämäärityksiä. Tällöin riskinä on se, että joitain asioita on ymmärretty väärin tai ei ole tarkennettu tarpeeksi. Silti myös ketterällä kehitysmenetelmällä tehty projekti alkaa aina työmääräarvion tekemisestä. Annettua työmäärää käsitellään kuitenkin arviona sen sijaan, että lähdettäisiin orjallisesti toteuttamaan projektia työmäärän antamissa puitteissa. Ketterissä menetelmissä käydääm toteutuksen aikana vuoropuhelua IT -toteuttajan projektitiimin sisällä sekä IT-yksikön, että ohjelmistoprojektin tilaajan välillä. Tällä tavalla saadaan aikaan vuorovaikutteinen joustava prosessi, joka mahdollistaa puuttumisen mahdollisiin virheymmärryksiin. Vuorovaikutteisuus ja dialogin käyminen ovatkin ketterien kehitysmenetelmien oleellisimmat piirteet. Kasurinen kirjoittaa kirjassaan, että ketterien järjestelmäkehitysmenetelmien lähtökohtana käytetään vuonna 2001 julkaistua "Ketterän ohjelmistokehityksen julistusta" (Manifesto for agile Software Development), vaikka ketterien kehitysmenetelmien juuret ovatkin paljon vanhempia. IBM kehitti ohjelmistojaan iteratiivislla menetelmillä jo vuonna 1957. Yrityksille ketterien ohjelmistokehitysmenetelmien laajempi käyttöönotto on tarkoittanut joustavampaa, selkeämmin aikataulutettavaa ja kustannustehokasta tapaa kehittää ohjelmistoja. Ohjelmistotuottajille taas ketterät menetelmät tarjoavat mahdollisuuden toteuttaa ohjelmistot ajallaan ja annetun budjetin ja aikataulun puitteissa.

2.1 Vesiputousmalli

Ohjelmistoprojekteja on vuosien varrella tehty erilaisilla menetelmillä. Niistä yleisin oli jo aiemmin mainittu vesiputousmalli. Dean Leffingwellin esittelee teoksessaan Agile Software Requirements; Lean Requirements Practices for Teams, Programs, and the Enterprise vesiputousmallin. Se syntyi Leffingwellin mukaan tarpeesta ennustaa sekä kontrolloida paremmin ohjelmistoprojekteja 1950 ja 1960-luvuilla. (Leffingwell 2011, 5).

Vesiputousmalli pyrkii hahmottamaan ohjelmistokehityksen työvaiheet ja mahdollistamaan niiden toteutuksen strukturoidusti ja selkeästi. Leffingwell on kuvannut sitä kirjassaan yksinkertaistetusti alla olevan kuvan näköisesti.



Kuva 2. Vesiputousmalli (Leffingwell 2011, 5)

Vesiputousmallin määrittelyvaiheessa pyritään keräämään mahdollisimman tarkasti koko toteutettavan muutoksen järjestelmävaatimukset tilaajalta ja arvioimaan siihen käytettävä aika. Näiden vaatimusten pohjalta toimittaja lähtee ku-

vaamaan järjestelmään vaadittavaa mallia. Mallin pohjalta rakennetaan järjestelmätoiminnallisuus toteutusvaiheessa. Tämän jälkeen toteutettu toiminnallisuus testataan ja mikäli se todetaan toimivaksi, siirrytään tuotantoon siirtovaiheeseen.

Vesiputousmalli ei kuitenkaan yleensä saavuttanut sille määriteltyjä vaatimuksia ohjelmistoprojektien kulujen ja aikataulujen ennustettavuudesta. Projektien lopputuloksien arvioinneissa havaittiin näet, että suuressa osassa projekteja halutut toiminnallisuudet ja lopputulokset saavutettiin vain osittain tai ei ollenkaan projektien jäädessä kokonaan kesken. Robert Wysocki määrittelee vesiputousmallin toimintatavaksi, joka ei katso taaksepäin. Kun yksi vaihe on saatu valmiiksi, niin prosessi siirtyy seuraavaan vaiheeseen (Wysocki 2013, 368). Voidaan siis hyvin olettaa, että vesiputousmalli on herkkä aikataulumuutoksille ja viivästyksille. Kehitysmalli, jossa siirrytään orjallisesti vaiheesta toiseen, vaatii viivästystilanteissa, että aikaa otetaan seuraavasta vaiheesta aiemmin vaiheen aikavajeen korvaamiseen.

Leffingwell lainaa teoksessaan Standish Groupin Chaos report surveyta vuodelta 1994, jossa esitellään tilastolukuja projekteista, joissa on käytetty vesiputousmallia.

- 31 % projekteista peruttiin ennen kuin ne saatiin päätökseen.
- 53 % tuli maksamaan enemmän kuin 189 % annetusta arviosta
- Vain 16 % valmistui aikataulun ja budjetin puitteissa.
- Suurimpien yritysten valmistuneet projektit tuottivat vain 42 % alkuperäisistä ominaisuuksista ja toiminnallisuuksista.

(Leffingwell 2011, 6).

Vesiputousmallin käyttäminen sovelluskehityksessä aiheutti siis sekä ylittyviä kustannuksia, projektien perumisia että vaillinaisten toiminnallisuuskokonaisuuksien toimittamista loppuasiakkaalle. Voidaanko siis yllä olevien prosenttien perusteella päätellä, että vesiputousmalli on täysin toimimaton sovelluskehitysmenetelmä? Näin ei tietenkään ole, mutta sen käyttö on haastavaa erityisesti projekteissa, joissa toimittajan ja tilaajan ymmärrys projektin tavoitteista, vaati-

muksista, kuluista ja aikatauluista eivät ole yhtäläiset. Kuitenkin vesiputousmallia käytetään edelleen monissa eri yhtiöissä onnistuneesti. Charles Cobb kirjoittaakin kirjassaan, että polarisaatio ketterien menetelmien ja vesiputousmallin välillä on ollut suuri. Tämä johtuu hänen mukaansa siitä, että ketterät menetelmät edustavat poistumista byrokraattisista toimintamalleista sovelluskehityksessä ja ketterien kehitysmallien käyttäjät halusivatkin ottaa mahdollisimman paljon etäisyyttä vesiputousmalliin. (Cobb 2011, 7).

Vesiputousmallin suurimpiin heikkouksiin kuuluu sen joustamattomuus. Jokainen viivästys prosessin eri vaiheissa aiheuttaa aikatauluongelmia myöhäisemmissä vaiheissa. Tämä johtuu siitä, että vesiputousmallissa toteutettavat järjestelmämäärittelyt on lyöty lukkoon jo ennen toteutuksen alkua niin tarkasti, etteivät ne mahdollista joustoa, jos aikataulu tai kulut alkavat venyä. Jos esimerkiksi toteutusvaiheessa ilmenee ongelmia, täytyy aikataulua säätää. Vesiputousmallissa otetaan usein aikaa toteutusvaihetta seuraavasta vaiheesta eli testauksesta, mikä johtaa siihen, että järjestelmän virheettömyys kärsii, koska testihenkilöstö ei ehdi tarkastamaan järjestelmää tarpeeksi hyvin virheiden varalta. Tämä taas johtuu siitä, että vesiputousmallissa ei ole aikataulullista joustoa ja uudelleen allokoitua aikaa ei enää saa takaisin.

Testaukseen käytettävän ajan problematiikka on ollut melko yleinen ongelma tutkimassani yrityksessä ennen ketterien menetelmien käyttöönottoa. Prosessin aikatauluhaasteiden kertautuminen kokonaisprosessin joka vaiheessa aiheutti melko usein tilanteen, jossa testaukseen käytettävä aika saattoi pahimmassa tapauksessa puolittua ajasta, joka oli alun perin allokoitu testaukseen suunnitteluvaiheessa. Tällöin ei testaaja tai toteuttaja voinut sanoa varmasti, että tuotettu kokonaisuus oli siirtokelpoinen tuotantojärjestelmiin.

Testauksen ongelmallisuuden vesiputousmallin osalta toteavat myös Ilkka Haikala ja Tommi Mikkonen kirjassaan Ohjelmistotuotannon käytännöt. He lainaavat kirjassaan Roycen toteamusta. Royce toteaa, että vesiputousmallissa on erittäin tärkeää iteroida myös taaksepäin, ja mikäli järjestelmä jota ollaan tekemässä on täysin uusi, tulisi se toteuttaa kahdesti. Roycen mukaan vesiputousmalli soveltuu siis erittäin huonosti uusien järjestelmäkokonaisuuksien tekemi-

seen. Totesin jo aiemmin, että vesiputousmalli ajautuu useasti ongelmiin, jos tilaajalla ja toimittajalla ei ole yhteistä näkemystä projektin tavoitteista, vaatimuksista, kuluista ja aikatauluista. Tällaisia ongelmia tulee useasti vastaan juuri uusien sovellusten tai toiminnallisuuksien tekemisessä.

Haikalan ja Mikkosen kirjassa Royce toteaa viimeisenä, että mikäli järjestelmän ongelmat paljastuvat vasta testausvaiheessa voi kustannusarvio helposti kaksinkertaistua (Haikala - Mikkonen 2011, 37).

Leffingwell listaa teoksessaan seuraavia Standish Groupin havaintoja siitä miksi vesiputousmallilla tehdyt projektit ajautuvat usein ongelmiin.

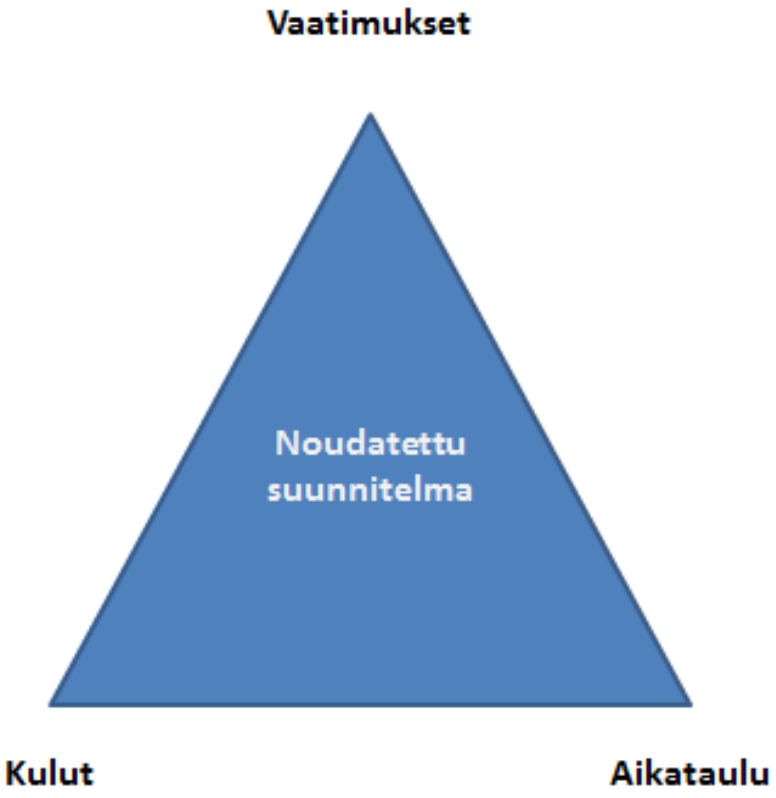
- Käyttäjien palaute puuttuu: 13 % kaikista projekteista
- Keskeneräiset määritykset ja vaatimukset: 12 % kaikista projekteista
- Muuttuvat määritykset ja vaatimukset: 12 % kaikista projekteista

(Leffingwell 2011, 6).

Joustamattomuus prosessissa siis aiheuttaa sen, että mikäli järjestelmämäärittelyt ovat puutteelliset tai muuttuvat, ajautuu prosessi ongelmiin. Tästä johtuen myös kommunikaation ja palautteen puute aiheuttaa sen, että projekti viivästyy tai pahimmassa tapauksessa joudutaan perumaan kokonaan. On itsestään selvää, että mikäli tilaaja antaa toimittajalle puutteelliset määrittelyt on lopputulos harvoin tavoitellun kaltainen. Selitys ei kuitenkaan aina ole niin yksinkertainen. On oleellista ymmärtää, että tilaaja ei aina edes itse ymmärrä, mitä sovellukseltaan haluaa. Kyse ei välttämättä ole ylimalkaisuudesta. Järjestelmämäärittelyjä tehtäessä on kaikkein oleellisinta, että tilaaja ymmärtää, mitä on tilaamassa, ja toimittaja ymmärtää, mitä on toimittamassa. Vesiputousmalli ei mahdollista jatkuvaa seurantaa sille, että tilaaja näkisi, mitä toteuttaja tekee. Prosessin ensimmäinen vaihe siis korostuu. Määrittelyt täytyy siis tehdä niin, että molemmat ovat tekemässä samaa asiaa. Mikäli näin ei ole, tulee määrittelyihin muutoksia ja vesiputous -kehitysmallin tiukka aikataulu rikkoutuu. Charles Cobb kirjoittaa kirjassaan, että vesiputousmallista sovelluskehitystä ajaa suunnitelmaohjaus, joka perustuu hyvin vahvasti etukäteen suunnitteluun ja kontrolliin ja niissä on hyvin vähän joustoa. Näitä menetelmiä hän kutsuu suunnitelmaohjatuiksi tai

ennustaviksi, koska niissä käytetään vahvasti etukäteissuunnittelua projektin aikataulun ennustamiseen (Cobb 2011, 5).

Leffingwell puolestaan kuvaa vesiputousmallia hallitsevaa toimintaprosessia rautakolmio -ansaksi.



Kuva 3. Rautakolmio (Leffingwell 2011, 7)

Rautakolmiolla Leffingwell tarkoittaa tilannetta, jossa käytetään noudatetun suunnitelman prosessia, jossa vaatimusvaihe ohjaa toteutusta. Leffingwellin mukaan aiemmin mentiin ansaan, kun oletettiin, että osa määrittelyistä voitaisiin päätellä etukäteen siten, että toteutusprojektin budjetti pystyttäisiin ennustamaan jossain määrin etukäteen. Tämä taas johti kiinteähintaisten sopimusten tekemiseen, joissa tilaaja maksaa toteuttajan määrittelemän kustannuksen tietyssä ajassa ja toteuttaja on sitoutunut toteuttamaan tilaajan toiveet niiden puit-

teissa. (Leffingwell 2011, 7). Kuten aiemmin on todettu, ohjelmistoprojektit ovat vain harvoin staattisia kokonaisuuksia, joissa määrittelyiden lukkoon lyöminen aikaisessa vaiheessa mahdollistaa sujuvan ja toimivan kehitysprosessin. Leffingwellin mukaan juuri tämä kiinteiden määrittelyiden ajatus osoittautui monen projektin epäonnistumisen syyksi.

2.2 Vesiputousmalli käytännössä

Onko vesiputousmalli siis pohjimmiltaan huono sovelluskehitysmenetelmä? Jim Highsmith käsittelee vesiputousmallia kirjassaan Agile Project Management; Creating Innovative Products. Hän kirjoittaa, että vaikka vesiputousmallisia tapoja vastaan hyökätään niiden yksinkertaisuuden takia ne jatkavat vaikuttamistaan sopimusprosesseissa ja johdon perspektiiveissä. (Highsmith, 2010, 11) Vesiputousmallin ymmärtäminen on siis Highsmithin mukaan ensiarvoisen tärkeää sovelluskehitystyössä, koska malli vaikuttaa yhä vahvasti tilaaja - toimittaja -prosesseihin. Vesiputousmalli sopiikin hyvin niin sanottujen kiinteähintaisten sopimusten tekemiseen. Kiinteähintaisella sopimuksella tarkoitetaan toimittajan tekemää tarjousta tuotettavasta ohjelmistosta, jonka hinta- ja aikatauluarvio on kiinteä. Chemuturi kirjoittaakin kirjassaan, että kiinteähintaisissa sopimuksissa ei ole mitään ihmeellistä ja niissä ei ole mitään epätavallista. Kiinteähintaiset sopimukset sopivat hyvin asiakkaan tarpeita vastaavia tuotteita tekevien ohjelmistotuottajille ja ne tarjoavatkin näitä sopimuksia vain, jos ovat saaneet tarkat vaatimukset ja niiden tuottamat tarjoukset sisällyttävät usein myös korkealaatuisen suunnitelman toteutusta varten (Chemuturi 2009, 14). Tämä sopii vesiputousmalliin hyvin vahvan alkudokumentaation ja suunnittelun takia. Se ei kuitenkaan poista vaatimusta ammattitaitoisesta ja toisensa tuntevan tilaajasta ja toimittajasta. Etenkin toimittajalla on oltava pitkä kokemus tilaajan järjestelmistä sekä ymmärrys siitä, mitä tilaaja milläkin järjestelmävaatimuksella hakee. Myös Coplien, James, Bjørnvig ja Gertrude kirjoittavat kirjassaan, että mikäli kehittämistoiminta sujuu vesiputousmallilla hyvin voi sitä yhä käyttää. Jos tunnet määrityksesi ja ne eivät muutu niin vesiputoumalli on ihan hyvä. (Coplien ja Bjørnvig 2010, 216).

Vesiputousmallissa on tiettyjä etuja ketteriin sovelluskehitysmenetelmiin verrattuna. Agarwal ja Tayal listaavat sekä positiivisia, että negatiivisia puolia kirjassaan seuraavasti:

Vesiputousmallin edut:

- se on lineaarinen malli
- se on segmentoitu mallii
- se on systemaattinen ja peräkkäin toteutettava
- se on yksinkertainen
- siinä on asianmukainen dokumentaatio

Vesiputousmallin haitat:

- on vaikeaa määritellä kaikki vaatimukset projektin alussa
- malli ei ole sopiva minkäänlaisiin muutostilanteisiin
- toimivaa versiota järjestelmästä ei nähdä kuin vasta projektin elinkaaren loppupuolella
- se ei sovellu hyvin isoihin projekteihin
- se vaatii raskasta dokumentaatiota
- siinä ei voida palata takaisin, kun sovelluksen elinkaari on kehityksessä
- siinä ei ole esimerkkimallia selkeästi siitä mitä asiakkaat haluavat
- siinä ei ole riskianalyysia
- mikäli jossain vaiheessa tehdään virheitä, niin hyvää sovellusta ei pystytä tekemään
- se on dokumentointiohjattu prosessi, joka vaatii muodollista dokumentaatiota jokaisen vaiheen lopussa

(Agarwal ja Tayal 2008, 31)

Heidän listaamansa asiat vesiputousmallin hyvistä puolista, toimivat hyvin noudatetun suunnitelman prosessissa. Ongelmana on vain haittojen suuri määrä. Itse en kuitenkaan allekirjoita, kaikkia heidän mainitsemiaan haittoja. Esimerkiksi sitä, että vesiputousmalli ei soveltuisi isoihin projekteihin, tai jos jossain vesiputousmallin vaiheessa tehdään virheitä, ei hyvää sovellusta pystytä tekemään.

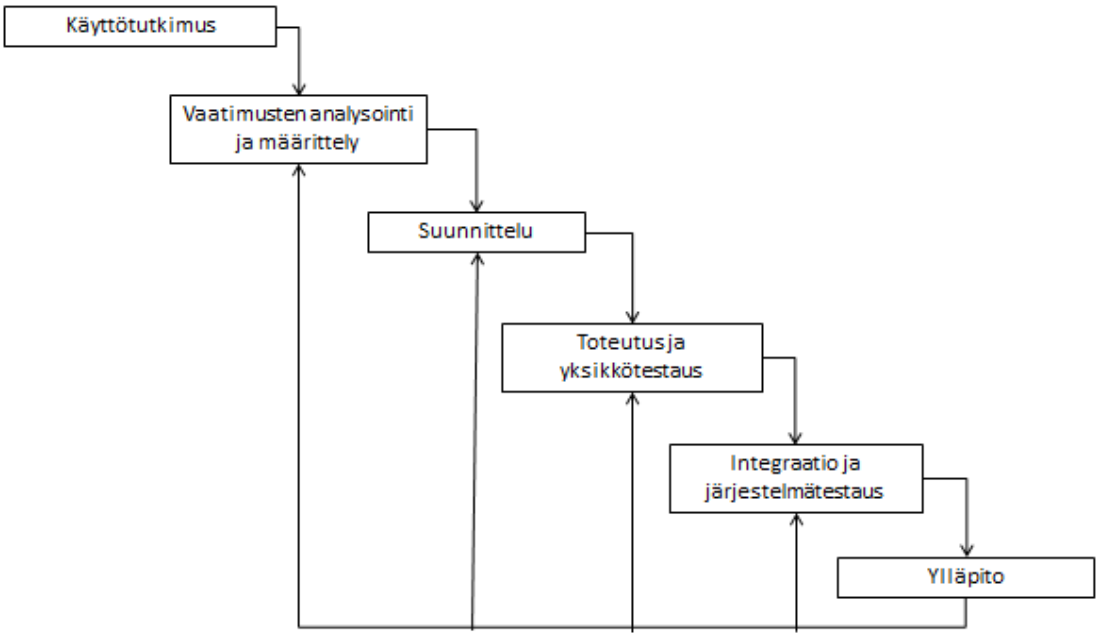
Olen itse osallistunut isoihin vesiputousmallisiin projekteihin, joissa on havaittu virheitä vasta sovelluksen testausvaiheessa, ja virheet on saatu korjattua onnistuneesti. Ongelma liittyy enemmänkin siihen, että vesiputousmallissa virheiden korjaus on erittäin työlästä, koska koko järjestelmäkokonaisuus on toteutettu ennen testausta, ja virhe voi olla jossain syvällä, muuten toimivassa koodissa.

Negatiivinen asenne vesiputousmallia kohtaan juontaa juurensa vahvasta polarisaatiosta vesiputousmallin- ja ketterien menetelmien puolestapuhujien keskuudessa. Cobbin kirjassa on lainaus Boehmin ja Turnerin havainnoista asiaan liittyen. He toteavat, että valitettavasti sen sijaan, että pyrittäisiin löytämään tapoja tukea toisiaan, nämä kaksi sovelluskehityksen lähestymistapaa, pitävät toisiaan vastapuolina nollasumma pelissä. Ketterien menetelmien kannattajat herjaavat perinteisten mallien kannattajia, ja syyttävät heitä sovelluskehityksen epäinhimillistämisestä, ja prosessin palvonnasta. Tähän vesiputouksen kannattajat vastasivat syytöksillä keskinkertaisuudesta, huonosta laadusta ja siitä, että vakavassa liiketoiminnassa pidetään vain hauskaa (Cobb 2011, 7). Yhteisen näkemyksen puute johtaakin siihen, että kirjallisuus liittyen sovelluskehitykseen, on usein vahvasti joko tietyn kehitystavan puolella, tai sitä vastaan.

Cobb mainitsee myös kirjassaan syyksi välirikkoon sen, että ketterä liike oli periaatteessa vallankumous byrokraattisia vesiputoustyylisiä prosesseja vastaan, ja ketterä yhteisö halusi sijoittaa itsensä mahdollisimman kauas niistä toimintatavoista. Hänen mukaansa molemmilla on paljon opittavaa toistensa työskentelytavoista, ja painottaakin sitä, että projektin metodologian valinnassa on yrityksissä otettava huomioon se, että valittu metodologia on linjassa yrityksen liiketoimintastrategian, toimintakulttuurin ja liiketoimintaympäristön kanssa, samalla huomioon ottaen yksittäisten projektien riskit (Cobb 2011, 8). Tämä vahvistaa näkemystä siitä, että mikäli vesiputousmalli toimii hyvin yrityksessä, on tarpeetonta alkaa tekemään suuria muutoksia kehitysprosessiin vain sen takia, että ketterät menetelmät on otettava väkisin käyttöön.

2.3 Kohti ketteriä kehitysmenetelmiä

Vesiputousmallin huono menestys modernissa sovelluskehitystoiminnassa johti muutokseen sovelluskehitysajattelussa. Näiden pohdintojen seurauksena alettiin kehittää niin sanottuja iteratiivisia menetelmiä. Rajib Mall kuvaakin teoksessaan muokattua vesiputousmallia, jossa on ketteristä menetelmistä tuttuja piirteitä.



Kuva 4. Iteratiivinen vesiputousmalli (Mall 2013, 41)

Hänen mukaansa verrattuna klassiseen malliin iteratiivinen vesiputousmalli antaa palautetta jokaisen vaiheen jälkeen, kun se on toteutettu. Palautteen avulla mahdollistetaan virheiden korjaamiset, jotka on tietyssä vaiheessa tehty, mikäli ne havaitaan vasta vaiheen jälkeen (Mall 2013, 41). Malli siis ohjaa vielä tarkasti vaiheittain miten projekti viedään loppuun, mutta perinteisestä mallista poiketen, se mahdollistaa reagoinnin muuttuviin tilanteisiin paremmin. Tämänkin mallin haasteeksi muodostuu se, että toteutettavat osakokonaisuudet, ovat erittäin suuria toteutettavaksi nopeatempoisesti ja tehokkaasti.

Sovelluskehityksessä huomattiin, että tarvittiin pienempiä osakokonaisuuksia eli, joita oli helpompi hallita ja toteuttaa. Näiden vaatimusten pohjalta luotiin ketterät kehitysmenetelmät. Barbee Davis luettelee kirjassaan miksi ketterät menetelmät ovat kasvattaneet niin paljon suosiotaan maailmalla. Hänen mukaansa kyse ei ole ohimenevästä muoti-ilmiöstä vaan ketterät menetelmät auttavat niitä käyttäviä:

- Ohjaamaan liiketoiminnan ketteryyttä
- Vastaamaan nopeasti muuttuvaan liiketoimintaympäristöön
- Kohottamaan projektien kohdentamista liiketoimintastrategioihin parhaan sijoitteun pääoman tuoton saavuttamiseksi
- Vaikuttamaan työntekijöiden palkkaamiseen sekä työpaikassa pysymiseen

(Davis 2012, 2)

Hänen mukaansa nämä tavoitteet saavutetaan olemalla joustavampia tavassa, jolla johdamme projekteja. Ketterä -sana liitetään usein sovelluskehitykseen, mutta hänen mielestään puhtaimmassa muodossaan se tarkoittaa kykyä olla joustavampi ja pysyä mukana muutoksessa (Davis 2012, 2). Davis siis määrittelee ketteryyden koko organisaation toiminnan perusajatuksena. Sitä voidaan soveltaa sekä ohjelmistotuotannossa, projektinhallinnassa että yrityksen strategisten maalien asettamisessa. Myös yrityksen koko organisaatio voi olla ketterä. Clark määrittelee ketterän organisaation sellaiseksi organisaatioksi, joka pystyy luomaan itselleen kilpailuetua kyvyllä sopeuttaa sen henkilöstö ja prosessit koko ajan muuttuviin markkinavaatimuksiin. Tämä saavutetaan kasvavassa määrin teknologisten innovaatioiden avulla (Clark ja Baker 2004, 119). Tämän tutkimuksen tarkoituksena on hahmottaa ketterän kehitysmallin käyttöönottoa tutkimassani yrityksessä.

3 KETTERÄT KEHITYSMENETELMÄT

Ketterien menetelmien lähtökohtana pidetään niin sanottua ketterää julistusta. Tämä julistus tehtiin vuonna 2001, jolloin suuri joukko ketterien menetelmien metodologian kehittäjiä kokoontui niiden henkilöiden kanssa, jotka käyttivät ketteriä menetelmiä, keskustellakseen oliko parempia tapoja tehdä sovelluskehitystä (Leffingwell 2011, 12) Tämä julistus on pohjana ketterissä menetelmissä vielä nykyäänkin, kun uusia menetelmiä kehitetään. Ketterässä julistuksessa korostettiin seuraavia kohtia:

- Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
- Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
- Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
- Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

(Ketterän ohjelmistokehityksen julistus (http://agilemanifesto.org/iso/fi/)

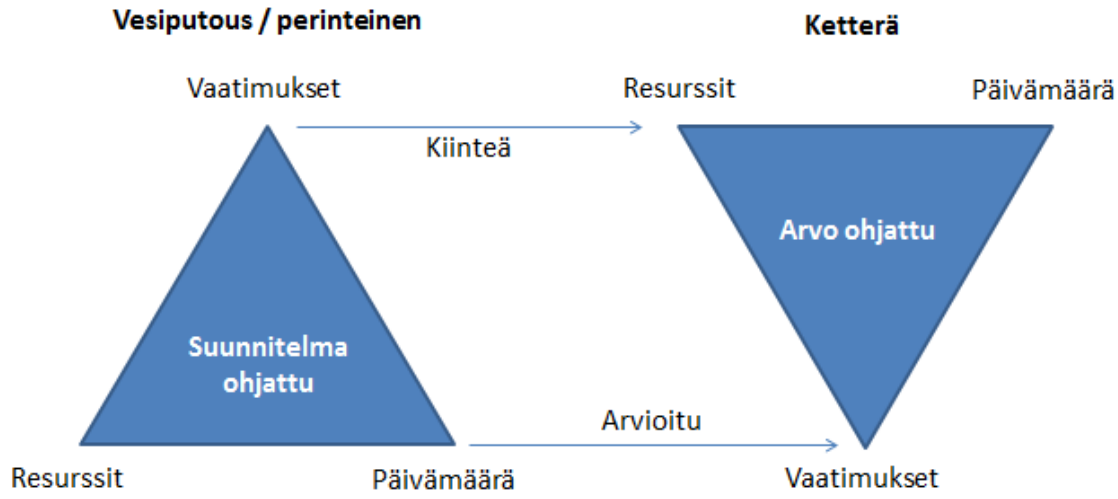
Ketterän julistuksen tarkoituksena oli siis täysin päinvastainen ajatus kuin vanhassa vesiputousmallisessa ohjelmistokehityksessä. Vesiputousmallin pääperiaatteita oli tarkka suunnittelu ja prosessin noudattaminen. Myös dokumentaation merkityksen vähentäminen ja asiakkaan kanssa keskustelu eivät olleet vesiputousmallissa kovinkaan oleellisessa asemassa. Ketterä julistus ei siis noudattanut perinteistä asiakas -toimittaja ajatusmallia, jossa asiakas kertoo mitä haluaa ja toimittaja toteuttaa tarpeen parhaaksi kokemallaan tavalla ja keskustelut onnistumisesta käydään vasta, kun sovellus on tuotannossa.

3.1 Pois noudatetusta suunnitelmasta ja rautakolmiosta

Ketterät menetelmät pyrkivät loiventamaan jyrkkää tilaaja toimittaja suhdetta kommunikaation avulla. Vaikka ketterissä menetelmissä on selkeät prosessit pyritään niissä poisoppimaan noudatetun suunnitelman -prosessista ja toteutusvaiheen muutoksien pelosta. Leffingwell kirjoittaa teoksessaan, että ketterissä menetelmissä on periaatteellisia eroja vaatimusmäärittelyihin verrattuna

vanhaan vesiputousmalliin. Ketterissä ei esiinny hänen mukaansa perinteisiä sovellus vaatimus määrittelyjä, suunnittelumäärittelyjä tai muita näiden kaltaisia. Sen lisäksi olettamus siitä, että toimittajan vastuulla on toteuttaa "kaikki se tavara" rajatussa aikataulussa ja rajatuilla resursseilla on poissa. (Leffingwell 2011, 16)

Tämä mahdollistaa Leffingewellin mukaan rautakolmion eliminoimisen ja siirtymisen noudatetun suunnitelman prosessista arvon ohjaukseen. Alla olevassa kuvassa on kuvattu, miten ketterät menetelmät muuttavat ohjausprosessia järjestelmätoteutuksessa.



Kuva 5. Vesiputous ja ketterä menetelmä (Leffingwell 2011, 16)

Leffingwellin mukaan ketterän menetelmän käyttöönotto mahdollistaa sen, että mikäli päivämäärä ja vaatimukset joutuvat vastakkainasettelutilaan päivämäärä voittaa. Tämä tarkoittaa sitä, että vaatimukset eivät koskaan pakota lisäämään käytettävissä olevaa aikaa eli ylitöitä ei ketterissä menetelmissä tehdä. Sopimus siitä, että ylitöitä ei tehdä taas korjaa, kaksi haastetta, jotka ovat arvioitu aikataulu ja resurssit. Vaatimuksilla taas on ketterissä menetelmissä kelluva ja muuttuva rooli, koska ne ovat alttiita muutoksille. (Leffingwell 2011, 16) Myös Cobb kirjoittaa kirjassaan ketterien menetelmien järjestelmämäärityksen ominaisuuksista. Hänen mukaansa ketterät menetelmät vaativat enemmän

joustavuutta käyttäjien tarpeille ja vaatimuksille, kun projekti etenee. Tämän lisksi vaaditaan paljon alempaa painotusta ennakkoon suunnitteluun ja kulujen ja aikataulujen kontrollointiin (Cobb 2011, 5). Tämä tarkoittaa siis sitä, että vaatimukset ja määrittelyt voivat muuttua, mutta ketterissä menetelmissä ei olemassa olevaan toteutukseen laiteta enää uusia ominaisuuksia kesken toteutuksen, vaan niitä voidaan pitää jatkokehityksenä, kun projekti on siirtynyt ylläpitoon.

3.2 Ketterät menetelmät ja kommunikointi

Ketterän julistuksen tärkeimpänä kohtana voidaan mielestäni pitää kommunikaation lisäämistä tilaaja - toimittaja suhteessa. Sujuva kommunikaatio mahdollistaa sen, että tilaaja ja toimittaja pysyvät ajan tasalla siitä, mitä toinen järjestelmältä haluaa ja missä toteutusvaiheessa järjestelmä on. Tämä on erityisen tärkeää siksi, että kuten todettua ketterissä menetelmissä ei ideaalitilanteissa tehdä ylitöitä, vaikka vaatimukset saattavat muuttua. Kommunikaation lisäämiseen liittyy kuitenkin kaksi harhakuvitelmaa, jotka usein liitetään ketteriin menetelmiin.

Ensimmäinen näistä on toimivan ohjelmiston rakentaminen kattavan dokumentaation sijaan. Ketterää kehitystapaa ymmärtämätön saattaa käsittää tämän siten, että mitään dokumentaatiota ei tarvita, vaikka todellisuudessa ketterässä kehitysmallissa tehdään erittäin kattavaa dokumentaatiota, joka on vain muodoltaan erilaista kuin vesiputousmallin dokumentaatio. Toinen kohta, joka saatetaan ymmärtää väärin, on muutokseen vastaamisen tärkeys suunnitelmassa pitäytymisen sijaan. Ketterä kehitysmalli pyrkii siihen, että tilaaja saa juuri sellaisen sovelluksen, jollaista on halunnut. Tämä vaatii tietyllä tavalla muuttuvia ja eläviä järjestelmämäärittelyjä, joita voidaan tarvittaessa tarkistaa. Se ei kuitenkaan tarkoita sitä, että toimittaja lähtee tuottamaan ohjelmistoa ilman minkäänlaista suunnitelmaa tilaajan puolelta.

Jim Highsmith kuvaa kirjassaan erittäin hyvin, mitä ketterässä sovellusmallin projektin hallinnassa pyritään saavuttamaan. Hän listaa viisi kohtaa, joiden yhtymäkohdat on helposti johdettavissa ketterästä julistuksesta.

1. Jatkuva innovointi - tämänhetkisten asiakasvaatimusten toimittaminen
2. Tuotteen sopeutumiskyky - tulevaisuudessa tehtävien asiakasvaatimusten toimittaminen
3. Nopeampi markkinoille pääsy - markkinoiden aikaikkunoiden tavoittaminen ja sijoitetun pääoman tuoton parantaminen
4. Henkilö ja prosessi sopeutumiskyky - kyky vastata npeasti tuote- ja liiketoimintamuutokseen
5. Luotettavat tulokset - liiketoiminnan kasvun ja tuottavuuden tukeminen

Viisi avain liiketoimintatavoitetta ketterässä kehityksessä (Highsmith 2010, 10)

Jos mietitään ketterän julistuksen perusperiaatteita eli yksilöitä ja kanssakäymistä, toimivaa ohjelmistoa, asiakasyhteistyötä sekä vastaamista muutokseen, on helppoa ajatella, että yllä olevat vastaavat näihin tavoitteisiin. Kommunikaatio korostuu erityisesti jatkuvassa innovoinnissa, jolla pyritään havainnoimaan asiakasvaatimukset tuotteen sopivuudessa sekä henkilöiden, ja prosessien sopivuudessa. Ketterä kehitys lähteekin hyvin pitkälti liikkeelle avoimesta kommunikaatiosta ja läpinäkyvyydestä. Holcombe mainitseekin kirjassaan, että hänen mielestään kommunikaatio on kaikkein tärkein asia ketterissä menetelmissä. Hänen mukaansa ketterät menetelmät ovat ennen kaikkea yhteistyötä, kommunikaatiota ja ihmisten kunnioittavaa ja luotettavaa kohtelua. Kaikki ongelmat ovat ratkaistavissa jollain tavalla, vaikka ratkaisu ei aina ole se mitä olisi odottanut. Joissain tapauksissa käytännön läheinen tapa edetä sattaa olla kivulias (Holcombe 2008, 65)

Dialogin merkitys kaikilla tasoilla on erittäin tärkeä ketterän projektin onnistumisen kannalta. Vesiputousmallissa ei kommunikaation merkitystä ole kuvattu tai sitten kommunikaatio on ollut enemmän yksisuuntaista ja hyvin läpinäkymätöntä. Kuten aiemmin on todettu, moni vesiputousmallin projekti ei päättyessään ole tuottanut asiakkaan haluamaa lopputulosta. Ongelma on ilmeinen, jos toi-

mittaja ei ymmärrä mitä asiakas loppujen lopuksi haluaa ja isompiin ongelmiin päädytään, jos tilaaja ei osaa kuvata toiminnallisuutta tai sovellusta, jota on tilaamassa. Kun yhdistetään tämänlainen keskustelemattomuus toteutusprosessiin, jossa asiakas pääsee katsomaan valmista tuotetta vasta sitten, kun testausvaihe alkaa, on ongelmia tiedossa.

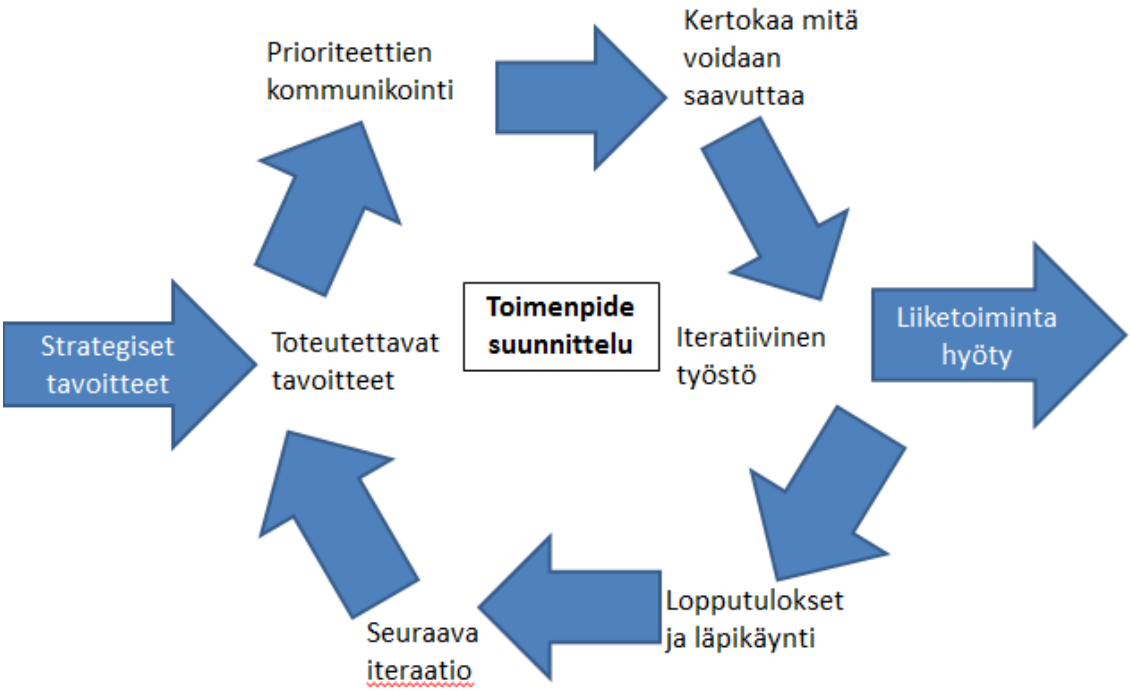
Tehokas kommunikaatio vähentää myös toteuttavan portaan epävarmuutta, mahdollistaa epävarmuuden hyväksymisen ja saattaa jopa saada toteutustiimin haluamaan muutoksia vaatimusmäärittelyihin. Jim Highsmith kuvaa teoksessaan yhtä ketterän kehitystiimin ominaisuutta, verraten sitä tuotteiden ominaisuuksiin. Hänen mukaansa, jos haluamme sopeutuvia tuotteita, täytyy myös ihmisten ja prosessien sopeutua muuttuvaan markkinatodellisuuteen. Jos halutaan sopeutuvia tuotteita, täytyy ensin rakentaa tiimejä, joiden jäsenet hyväksyvät muutoksen, eivätkä näe sitä esteenä, jota vastustetaan, vaan osana kukoistavaa ja dynaamista liiketoimintaympäristöä (Highsmith, 2010, 11). Ketterä kehitysmalli on siis syntynyt Highsmithin mukaan tarpeesta luoda joustavia tuotteita. Hänen mukaansa joustavien tuotteiden tekeminen vaatii myös niitä toteuttavilta joustavuutta ja tämän vuoksi ketterissä tiimeissä ja ympäristöissä työskentelevätkin parhaiten joustavat yksilöt, jotka eivät pidä muutosta uhkana, vaan osana kukoistavaa yritysympäristöä.

3.3 Nopeammin toteutuksesta tuotantoon

Ketterän kehitysmallin vahvuuksiin kuuluu myös nopeampi markkinoille pääsy. Ketterässä kehitysmallissa pidetään toteutuksen koosta riippuen säännöllisiä esimerkiksi viikottaisia, palavereja, joissa arvioidaan toteutustyön menossa olevaa vaihetta, toteutuksen haasteita ja mahdollisia muutoksia vaatimusmäärittelyihin. Tämä mahdollistaa reagoimisen muutokseen ja parantaa luottamusta tilaajan ja toimittajan välillä. Mikäli asiakkaan vaatimukset muuttuvat, päästään muutosta arvioimaan heti. Tämä mahdollistaa sen, etteivät muutokset aiheuta tilanteita, joissa toimittaja joutuu toteamaan, että sovelluksen rakenteen muuttaminen on prosessina tai kustannuksena niin suuri, ettei sitä kannata lähteä

toteuttamaan. Yleinen tilanne, jossa ollaan jouduttu toteamaan, että tiettyä muutosta ei kannata tehdä järjestelmiin sen kalleuden tai prosessin vaikeuden takia, saattaa johtaa hankalien ja työläiden alaprojektien tekoon, joiden kustannukset sekä ajankäytön että rahoituksen osalta ovat suuria. Avoin keskustelu asettaa myös toimittajalle vastuun siitä, että kommunikoidut muutostarpeet otetaan huomioon ja niiden vaikutukset arvioidaan. Goodpasture mainitseekin kirjassan, että ketterissä menetelmissä muuttumattomien määritysten ongelma on, että kirjoittajalla täytyisi olla tarpeeksi tietoa kirjoittaa täydellinen suunnitelma alusta asti. Hänen mukaansa ratkaisu saattaa lopulta ruokkia tarvetta. (Goodpasture 2010, 141). Ketterissä menetelmissä oletuksena onkin, että toteutuksen aikana käytävissä keskusteluissa määritykset tulevat muuttumaan ja tämä on hyväksyttävää.

Cooke kuvaa kirjassaan ketterää toteutusta reagoivaksi suunnitteluksi. Hänen mukaansa reagoivaan suunnitteluun osallistuu kaksi osapuolta, jotka ovat liiketoiminnan omistajat ja toteutustiimi, ja yhdessä ne muodostavat ketterän tiimin. Cooken mukaan reagoivassa suunnittelussa liiketoiminnan omistajat kommunikoivat toteutustiimille strategiset tavoitteensa osana iteraatioiden suunnittelukokousta ja toteutustiimi saavuttaa nämä tavoitteet realististen ja saavutettavien aktiviteettien kautta, joita he hallitsevat. Strategiset tavoitteet saavutetaan kuuden ydin toimenpiteen kautta.



Kuva 6. Toimenpide suunnitteluprosessi (Cooke 2010, 155)

Työvaiheita hän kuvaa seuraavasti:

- toteutettavat tavoitteet: liiketoiminnan omistajat purkavat strategiset tavoitteet pienemmiksi toimenpide ja liiketoimintatavoitteiksi ja kommunikoivat nämä tavoitteet toteutustiimille osana iteraation suunnittelukokousta.
- prioriteettien kommunikointi: liiketoiminnan omistajat tunnistavat korkeimman prioriteetin liiketoimintatavoitteet iteraation suunnittelukokouksessa
- kertokaa mitä voidaan saavuttaa: toteutustiimi neuvoo liiketoiminnan omistajia iteraation suunnittelukokouksessa siitä kuinka paljon he pystyvät mielekkäästi toteuttamaan seuraavassa iteraatiossa
- iteratiivinen työstö: toteutustiimi ottaa työn alle iteraatioon sovitut työt. Ideaalitilanteessa liiketoiminnan omistajat ovat tavoitettavissa työstön ajan antaakseen palautetta.

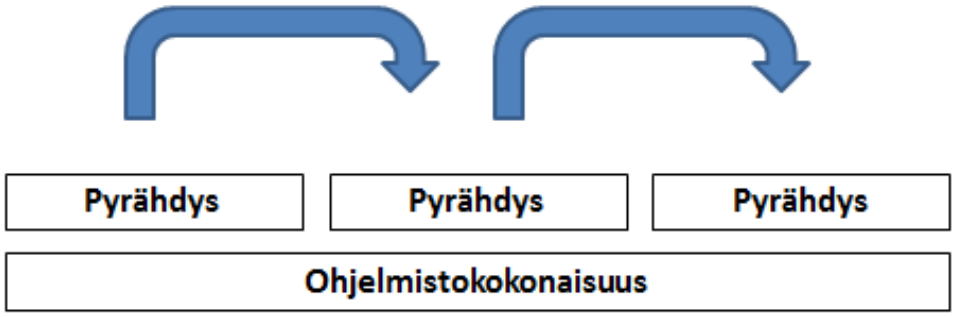
- jokaisen iteraation päätteeksi toteutustiimi esittelee tuotostaan liiketoiminnan omistajille tuotosten läpikäyntisessiossa
- läpikäynnin perusteella liiketoiminnan omistajat tunnistavat korkeimman prioriteetin toimenpiteet seuraavaan iteraatioon

Kyseessä on siis jatkuva prosessi, joka pyrkii jokaisessa uudessa työstövaiheessa tunnistamaan tärkeimmät tehtävät asiat. Työstösyklit ovat lyhyitä, joten prioriteettien muuttuessa, voidaan niiden toteutusaikataulua muuttaa.

Ideaalitilanne, jossa kyetään toimimaan luotettavassa ja avoimessa toimittaja - tilaaja suhteessa, luo luotettavan lopputuloksen. Avoin kommunikaatio, muutokseen haastava toimittaja ja projektiin sitoutunut tilaaja saavuttavat yleensä projektin lopputuloksena huomattavasti toimivamman sovelluksen kuin vesiputousmallissa.

3.4 Scrum

Scrum on yksi ketterien kehitystapojen alamalli ja varmaankin yleisimmin käytössä oleva. Scrumin ajatuksena on tehtävän sovellustyön pilkkominen mahdollisimman pieniin osakokonaisuuksiin, joita kutsutaan pyrähdyksiksi. Nämä osakokonaisuudet on helppo toteuttaa ja hahmottaa laajojen kokonaisuuksien sijaan. Toteutettava ohjelmistokokonaisuus puolestaan koostuu pyrähdyksistä. Alla on yksinkertaistettu kuvaus siitä, miten osakokonaisuudet muodostavat ohjelmistokokonaisuuden.



Kuva 7. Scrumilla toteutettu projekti

Kehitystöitä hallitaan aiemmin mainitulla tuotteen työlistalla, johon on kerätty toteutuksessa olevat työt, mahdolliset uudet työt, muutokset jo olemassa oleviin toimintoihin ja virheet sovelluksessa.

Scrumin teoriaan kuuluu oleellisena osana määritellä erilaisia rooleja ja prosesseja, jotka muodostavat yhdessä toimivan kokonaisprosessin, jota voidaan kutsua Scrumiksi.

Leffingwell kuvaa kirjassaan Scrumin ominaisuudet ja roolit seuraavasti:

- Työ tehdään pyrähdyksissä, jotka ovat aikasidonnaisia 30 päivän tai sitä lyhyempiä iteraatiotita.
- Pyrähdyksen työmäärä on aiemmin määritelty. Kun pyrähdyksen laajuus on päätetty, lisätoiminnallisuuksia ei voida enää lisätä muiden kuin kehitystiimin päätöksestä.
- Kaikki tehtävä työ on määritelty tuotteen kehityslistalla, joka sisältää uudet toimitettavat määrittelyt, järjestelmävirheet ja infrastruktuuri- ja suunnittelu aktiviteetit.
- Scrum Master mentoroi voimaannutettua, itseorganisaoituvaa ja itselleen vastuullista tiimiä, joka on vastuussa jokaisesta toimituksesta ja onnistuneesta pyrähdyksen lopputuloksesta.
- Tuoteomistajan rooli on toimia asiakkaan sijaisena.
- Päivittäinen seisontapalaveri on pääasiallinen kommunikointimenetelmä.

- Aikaikkunoihin asetetaan vahva fokus. Pyrähdykset, seisontapalaverit, tuotantoonsiirto katselmukset ja muut sen tapaiset kokoukset pidetään ennalta sovittuina aikoina.
- Tyypillinen Scrum -ohjeistus tukee 90 päivässä markkinoille saatavia tuottantoonsiirtoja, joihin sisältyy noin kolme pyrähdystä, jotka on määritelty noin 30 päivän mittaisiksi.

(Leffingwell 2011, 15)

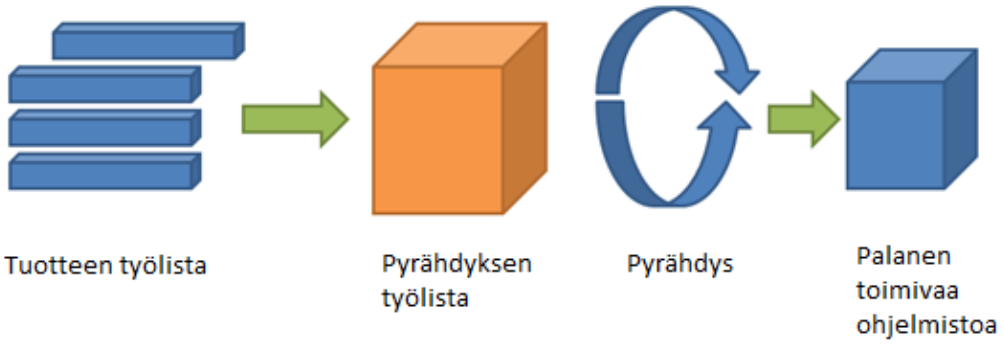
Kyseessä on siis malli, jossa odotetaan tiimiltä itseltään työkuormansa säätelyä ja ja itseohjautuvuutta. Toteutustiimin pitää pystyä arvioimaan itse, kuinka paljon työtä he saavat mahdutettua yhteen pyrähdykseen ja mahdollisuuksien mukaan lisäämään siihen uusia töitä.

Scrum Masterin rooli muistuttaa perinteistä IT -projektipäällikön roolia. Hän toimii toteutustiimin töiden mahdollistajana. Hän ei kuitenkaan määritä, missä järjestyksessä tai miten tiimin työt toteutetaan, vaan hänen tehtävänään on keskustella tilaajan kanssa ja toimia yhteistyössä tilaajan tuoteomistajan kanssa sekä mahdollistaa se, että toteutustiimi saa tehdä työtään rauhassa.

Tuoteomistajan rooli on toimia projektin omistajana. Hänen tehtäviinsä kuuluu keskustella projektin vetäjän eli Scrum Masterin kanssa projektin etenemisestä ja mahdollisista haasteista. Tuoteomistaja on tilaajan edustaja sovellusprojektissa. Hänen tehtäviinsä kuuluu kommunikoida liiketoiminnan vaatimukset toteutustiimille, keskustella Scrum Masterin kanssa siitä, miten tilaaja pystyy parhaiten varmistamaan sen, että toteutustiimi pystyy toteuttamaan vaaditun järjestelmäratkaisun, ja kommunikoida tilaajan edustajille projektin etenemisestä. Tuoteomistajan tärkein tehtävä on kuitenkin tuotteen työlistan ylläpito ja päivittäminen. Tuoteomistaja siis päättää ylätasolla, missä järjestyksessä toteutustiimi lähtee viemään eteenpäin toteutettavan sovelluksen osakokonaisuuksia ja millaisia muutostöitä sovellukseen voidaan toteutuksen aikana tehdä.

3.5 Scrum -prosessi

Aiemmin esittelin, millainen prosessi oli käytössä vesiputousmallissa. Scrumin yhtä pyrähdystä voisi kuvata virtaviivaistettuna vesiputousmallina pienoiskoossa. Alla oleva kuva on mukailtu Dean Leffingwellin teoksesta ja se kuvaa hyvin miten yksi pyrähdys toteutetaan scrumissa.



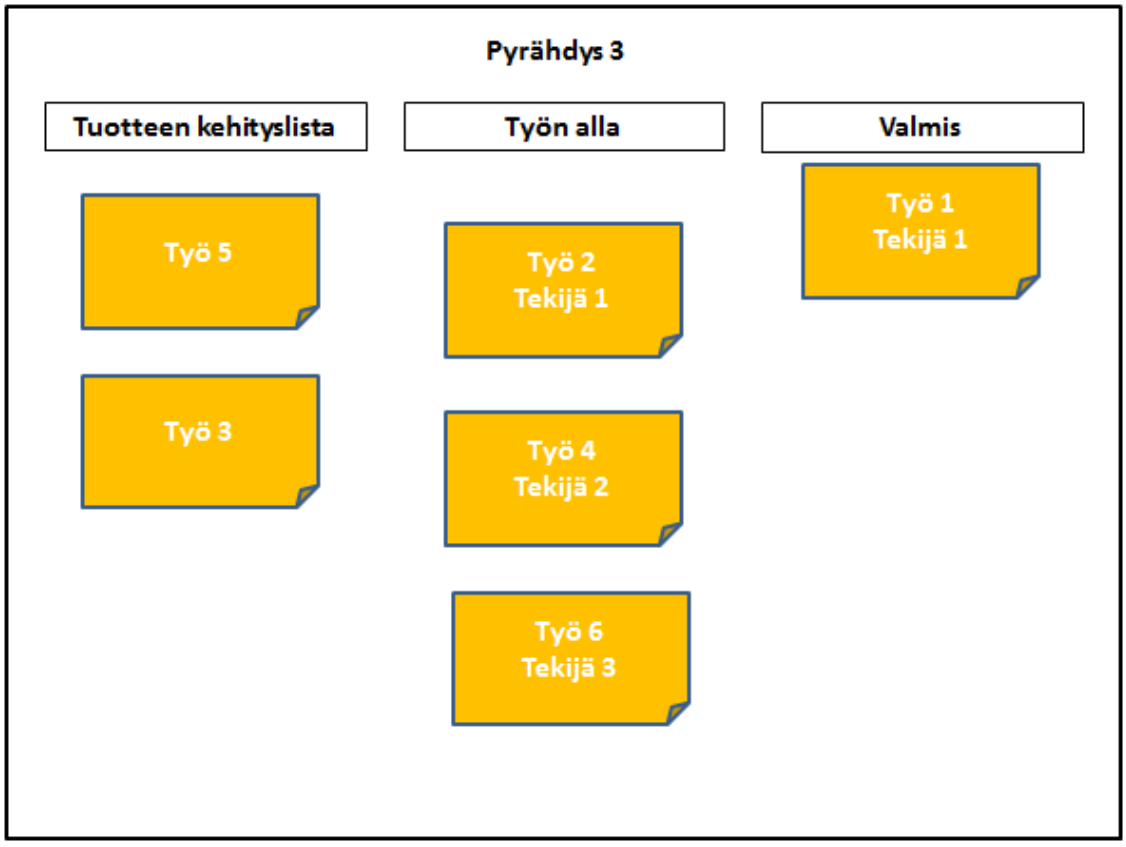
Kuva 8. Scrumilla toteutettu projekti, yhden pyrähdyksen sisältö

Kaikki siis alkaa siitä, että tuoteomistaja valitsee tuotteen työlistalta sopivat palaset, joita kutsutaan tarinoiksi, toteutettavaksi pyrähdykseen. Yleensä tarinat valitaan niin, että pyrähdyksen päätteeksi pidettävässä tarkastelussa on jotain konkreettista esiteltävää toimittajan muille edustajille. Hyvä esimerkki tästä on jokin perustoiminnallisuus toteutettavasta sovelluksesta.

Kun toteutettavat tarinat on valittu, siirretään ne pyrähdyksen työlistalle, joka on kuin tuoteomistajan työlista alatasolla. Ennen varsinaista pyrähdystä pidetään yleensä sen suunnittelukokous, johon osallistuvat tuoteomistaja, Scrum Master sekä toteutustiimi. Suunnittelukokouksen tarkoitus on koota erikseen jokaiseen pyrähdykseen toteutettavat työt pyrähdyksen työlistalle. Tiimi on vastuussa tuotteen työlistalta tulevien töiden pilkkomisesta toteuttamisen kannalta sopivan

kokoisiksi tarinoiksi pyrähdyksen työlistalle ja onkin tärkeää, että tiimi osaa jakaa isommat kokonaisuudet pienemmiksi palasiksi.

Tämän jälkeen ne viedään toteutustiimin seinälle. Seinä onkin yksi tärkeimpiä ketterien menetelmien työn etenemisen seurantatyökaluja. Seinä kuvaa aina toteutuksessa olevaa pyrähdystä ja siihen on koottu kaikki toteutustiimin työt, jotka siihen ollaan toteuttamassa. Peter Saddington määrittelee sen kirjassaan esimerkiksi yksinkertaisena prosessina, jossa on kolme kaistaa: tuotteen kehityslista, työn alla ja valmis (Saddington 2012, 74).



Kuva 9. Seinä

Oheisessa kuvassa on esimerkki siitä, miltä toteutustiimin seinä saattaa näyttää pyrähdyksen numero 3 osalta. Työt viisi ja kolme ovat edelleen tuotteen kehityslistalla ja niiden toteutusta ei ole vielä aloitettu. Työt kaksi neljä ja kuusi ovat työn alla ja työ yksi on valmistunut. Jokaiseen työhön on merkitty myös sen tekijä, jotta seinällä käyvä toteutustiimin jäsen näkee heti, mitä kenelläkin on työn

alla. Kokonaisuutena työt muodostavat yhden pyrähdyksen, joka valmistuu, kun kaikki työt ovat siirtyneet kohtaan valmis.

Pyrähdys on tehtävien töiden toteutusjakso, jonka aikana tiimi toteuttaa, testaa ja pyrähdyksen työlistalle viedyt työt. Pyrähdyksen tuloksena saavutetaan palanen toimivaa ohjelmistoa, joka voidaan esitellä tilaajan edustajille pyrähdyksen katselmuksessa. Jatkuva testaaminen onkin ketterissä menetelmissä yksi kaikkein tärkeimmistä periaatteista. Vanhoissa menetelmissä, kuten vesiputousmenetelmässä, testaus suoritettiin vasta toteutusvaiheen jälkeen. Tämä ehdollisuus aiheutti ongelmia, mikäli jo toteutusvaiheessa oltiin aikataulusta myöhässä. Ketterissä menetelmissä sen sijaan testausta tehdään jatkuvasti toteutuksen rinnalla. Toteutustiimin saadessa jonkin sovelluksen osan valmiiksi, voidaan se heti antaa testaajalle katsottavaksi virheiden varalta. Tästä johtuen ketterät menetelmät eivät ole niin sensitiivisiä aikataulumuutoksille verrattuna vesiputousmalliin.

Tutkimassani yrityksessä on ohjelmistotuotanto toteutettu siten, että IT toteutustiimi pilkkoo pyrähdykseen tulevat työt pienemmiksi tarinoiksi, arvioi töiden koon, toteuttaa pyrähdyksen ja tekee alustavan rakennetestauksen tehdyille palasille. Varsinainen liiketoiminta on kuitenkin vastuussa pyrähdyksen tuotantotestaamisesta ja siitä, että tehdyt ominaisuudet ovat virheettömiä.

Tämän yksinkertaistetun kuvauksen lisäksi sisältyy yhteen pyrähdykseen huomattava määrä osaprosesseja. Tämän opinnäytetyön kannalta on kuitenkin oleellista ymmärtää, miten Scrum toimii ylätasolla, koska kyseinen toteutustapa on käytössä tutkimassani yrityksessä.

3.6 Haasteita ja väärinymmärryksiä Scrum -työskentelytavan käyttämisessä

Samalla tavalla kuin vesiputousmallissa on ketterissäkin menetelmissä omat haasteensa ja hankaluutensa sekä suunnittelun että toteutuksen osalta.

Mikäli ketterään kehitysmalliin osallistuville henkilöille on epäselvää, mikä heidän roolinsa Scrumissa on, tulee vastaan ongelmia. Hyvänä esimerkkinä voi-

daan antaa Ken Schwaberin kirjassa oleva yliohjaava Scrum Master. Yksi Scrumin perusajatuksista on, että tiimi on vapaa valitsemaan, missä järjestyksessä se toteuttaa sille annetut työt. Schwaber kertoo kirjassaan seuraavan esimerkin, miten yhdessä yrityksessä hoidettiin toteutustiimin viikottainen stauspalaveri. Scwaberin mukaan palaverissa Scrum Master otti esiin listan ja ryhtyi lukemaan siitä eri henkilöiden nimiä kysellen ovatko he tehneet listassa heille merkityt kehitystyöt (Schwaber, 2004, 54). Kyseessä on melko yleinen virhe Scrum Masterilta, joka ei ole sisäistänyt, millä tavalla Scrumin työt järjestetään ja mikä hänen roolinsa on töiden järjestämisessä. Scrum Master ei suinkaan kerro toteutustiimin jäsenille, missä järjestyksessä ja mitä heidän tulee ottaa pyrähdyksen aikana milloinkin ottaa työn alle. Scrum Masterin tehtävä on vain pitää huolta siitä, että tuoteomistajan kanssa sovitut sovelluksen palaset tulevat tehdyiksi pyrähdyksen aikana. Yliohjaava Scrum Saster saattaa olla entinen tiiminvetäjä, joka on tottunut aiemmin järjestelemään tiimin työt ja kertomaan sen jäsenille, millaisia ominaisuuksia heidän tulee missäkin vaiheessa toteutusta ottaa työn alle.

Scrumin yksi oleellisimpia ominaisuuksia toteutustiimin kannalta on itseohjautuvuudentarve ja siihen liityvä mahdollisuus. Tämän tarpeen mahdollistaa se, että itse tiimi on huomattavasti autonomisempi kuin tiimit, jotka työskentelivät aiemmin vesiputousmallin kanssa. Roman Pichler kuvaa kirjassaan tiimin ominaisuuksia kolmella sanalla: "itseorganisoituva, ristiin toimiva ja pieni" (Pichler 2011, 7). Itseorganisoitumisen ongelma voi ilmetä ainakin jo edellä mainitulla tavalla, mikäli tiimin Scrum Master on yliohjaava. Tällöin tiimillä ei ole mahdollisuuksia suorittaa työn jakoa itse vaan se on riippuvainen siitä, miten Scrum Master työt tiimin sisällä jakaa. Toinen läheisesti yllämainittuun yliohjaamiseen liittyvä ongelma on tiimi, joka odottaa ohjausta. Tämä on selkeästi näkyvissä yrityksissä, joissa IT -toteuttajat ovat pitkän linjan tekijöitä ja työskentelytavat ovat muodostuneet hyvin vakiintuneiksi vuosien varrella.

On luonnollista, että tiimin sisällä tietyt tekijät erikoistuvat tekemään tiettyjä järjestelmän osakokonaisuuksia ja kun rutiini näiden tiettyjen palasten tekemiseen kasvaa, niin kyseinen tekijä tekee jatkossakin mielellään kyseisiä töitä. Tällai-

nen työskentelytapa ei lähtökohtaisesti toimi ketterien menetelmien kanssa. Ketterissä menetelmissä on tärkeää, että tiimin kaikki jäsenet pyrkivät jakamaan osaamista siten, että he kysenevät tekemään kaikkia työlistalla olevia sovelluksen tai järjestelmän palasia. Tätä tarkoitetaan ristiin toimivalla tiimillä. Mikäli tiimin sisällä on sanaton sopimus siitä, että tietyt toteuttajat tekevät vain tiettyjä osakokonaisuuksia, tulee toteutuksessa haasteita. Tämä näkyy selvimmin siinä, ettei osaaminen jakaudu tiimin sisällä. Kuitenkin yksi tärkeimmistä ketterän tiimin ominaisuuksista on se, että se on ristiin toimiva.

Tutkimassani yrityksessä on sovelluksien kehittämisessä pitkät perinteet ja ketterien menetelmien lanseerauksen yhteydessä muutama vuosi sitten oli selvästi nähtävissä, että osaaminen oli sirpaloitunut tiettyjen alueiden osalta vain muutamille tekijöille. Nykyään tilanne on parempi ja osaamisen jakamisesta on tullut jo arkipäivää. Tietyillä haastavammilla tai virheherkemmillä alueilla on kuitenkin nähtävissä yhä, että työn tekee parhaiten sen alueen osaava toteuttaja.

"Kaikki tai ei mitään" -ajattelu on myös yleistä ketterien menetelmien käyttäjien keskuudessa. Puhdas Scrum kuitenkin harvemmin istuu sellaisenaan kaikkeen sovelluskehitykseen ja kaikkiin sovelluskehityksen tilanteisiin. Cobb kirjoittaa kirjassaan, että joitain Scrumin käyttäjiä vaivaa juuri tällainen "kaikki tai ei mitään" -ajattelu, jossa perusajatuksena on, että Scrumia käyttävän organisaation on pakko toimia ketterällä tavalla sen kaikissa vaadituissa muodoissa ja rajanveto on niin tiukka, että joko niiden mukaan toimitaan tai sitten menetelmää ei käytetä ollenkaan. Cobb kirjoittaa, että Scrumin ja vesiputousmallin välissä on kuitenkin useita eri mahdollisuuksia toteuttaa sovelluskehitystä ketterästi (Cobb, 2011, 15). Myös Peter Schuh kirjoittaa kirjassaan Integrating Agile Development in the Real World, että ei ole väärin yhdistää eri elementtejä traditionaalisista kehitysmenetelmistä ja ketteristä menetelmistä. Hän kuitenkin toteaa, että edessä on todennäköisesti epäonnistuminen, jos projekti, joka on perustettu joillekin muille kuin ketterille periaatteille, ilmoittaa yhtäkkiä, että se on ketterä projekti (Schuh 2004, 59)

Tutkimassani yrityksessä on pyritty soveltamaan ketterää menetelmää siten, että se ottaa huomioon järjestelmien ja sovellusten kompleksisuuden ja sovel-

luskerrosten monimutkaisuuden. Lopputuloksena on saatu toimiva ketterä kehitysprosessi, joka ei noudata puhdasta Scrum -ajatusmaailmaa, mutta lainaa sieltä juuri niitä elementtejä, jotka tukevat tällaisessa ympäristössä tehtävää sovelluskehitystä.

Kehittäjän kannalta varmaankin pahin mahdollinen asia, joka voidaan ymmärtää ketterissä menetelmissä väärin, on tekemisen nopeuttaminen. Ketterät menetelmät usein markkinoivat itseään tehostavina ja nopeuttavina toimintamalleina. Tämä ei kuitenkaan tarkoita sitä, että monen tuhannen työtunnin työ saadaan ketterillä menetelmillä pienennettyä muutaman sadan tunnin mittaiseksi. Cobbin mielestä moni ohjelmistoprojekti on tullut tunnetuksi "kuoleman marssina". Hän lainaa kirjassaan Edward Yourdonia, joka määrittelee "kuoleman marssi" -projektin sellaiseksi projektiksi, jonka parametrit ylittävät normin ainakin 50 prosentilla. Se tarkoittaa hänen mukaansa, että projektille on asetettu ainakin seuraavat rajoitukset:

- Aikataulu on puristettu puoleen siitä, mitä voitaisiin saada rationaalisen arviointiprosessin tuloksena. Tällä tarkoitetaan sitä, että projekti, joka normaalisti vaatisi 12 kuukauden ajan, pitää saada valmiiksi kuudessa kuukaudessa tai aiemmin.
- Henkilöstö on supistettu vähempään kuin puoleen siitä, mitä tämän kokoiseen ja laajuiseen projektiin normaalisti määrätään.
- Budjetti ja siihen liittyvät resurssit on leikattu puoleen.
- Projektin toiminnallisuuksiin, ominaisuuksiin, suorituskykyvaatimuksiin tai muuihin teknisiin aspekteihin liittyvät asiat ovat kaksi kertaa niin suuret kuin normaalitapauksissa.

(Cobb 2011, 16)

Ketterät menetelmät nopeuttavat sovelluskehitystä, mutta ne eivät ole mikään pikaratkaisu, jos yritys tai organisaatio haluaa nopeuttaa tiettyä ohjelmistoprojektia lennosta tai tehdä koko kehityssalkustaan "ketterän". Myös Jamie Lynn Cooke mainitsee kirjassaan Agile productivity unleashed ketterien menetelmien väärinkäyttämisestä. Hänen mukaansa on tapauksia, joissa organisaatio on

ponnistellut ketterien menetelmien käyttöönoton kanssa aiemmin ymmärtämättä sen perustana olevia periaatteita. Esimerkiksi organisaatio siirtyy käyttämään ketterää kehitysmenetelmää, mutta vaatii yhä kaiken työn etukäteen sovittavaksi määrityksillä. Todelliset ketterät organisaatiot ymmärtävät, että reagoiva suunnittelu on arvokasta vain, kun organisaatio on asemassa, jossa se pystyy omaksumaan meneillään olevaa tehtävää työtä sen edetessä. Muuten iteratiivinen työskentelytapa muuttuu vain lyhyemmiksi tuotantoonsiirtosykleiksi, joilla on samat perusrajoitukset ja ketterät lähestymistavat saavat huonon maineen, kun tämä rajoitus realisoituu. (Cooke 2010, 144). On siis täydellinen väärinymmärrys alkaa istuttamaan ketterää kehitysprosessia yritykseen tai organisaatioon ymmärtämättä, millaisia vaatimuksia se asettaa kehitystiimille, sen resursseille tai projektien aikatauluille. Ketterä kehitys vaatii muutosta myös yritykseltä kokonaisuudessaan, ei vain kehitystiimin työskentelytavoilta.

Ketterien kehitysmenetelmien käyttöönotto ei ollut täysin selkeää tutkimassani yrityksessä, kun toimintatavan käyttöönottoa aloitettiin. Virheistä ja ongelmista otettiin kuitenkin oppia ja tällä hetkellä prosessi toimii hyvin.

Tyypillisin väärinymmärrys, joka liittyy ketteriin menetelmiin, on dokumentaation merkityksen unohtaminen. Ketterät menetelmät pyrkivät etäännyttämään itsensä vanhoista suunnitelmaohjatuista sovelluskehitystavoista. Ketterän kehityksen manifestissa jopa mainitaan, että sillä halutaan luoda toimivaa ohjelmistoa kattavan dokumentaation sijaan. Tämä saattaa joidenkin korvaan kuulostaa siltä, että ketterissä menetelmissä ei tarvitse tehdä dokumentaatiota. Tästä ei kuitenkaan ole kysymys. Ketterissä menetelmissä dokumentaatio muodostuu erilaisista asioista. Vesiputousmallissa pyrittiin mahdollisimman tarkkaan kuvaamaan, mitä tilaaja haluaa etukäteen, minkä pohjalta toimittaja lähti rakentamaan ohjelmistoa. Tapaa tehdä voisi kutsua "syteen tai saveen" -menetelmäksi ja kuten todettua, mahdolliset virheet dokumentaatiossa rikkoivat hyvin usein koko ohjelmistoprojektin rakenteen ja aikataulun.

Ketterät menetelmät taas painottavat kommunikaation merkitystä. Sen sijaan, että tilaaja kirjoittaa toimittajalle sadan sivun mittaisen järjestelmävaatimuksen, jota lähdetään toteuttamaan, on ketterien menetelmien mukaan tarkoituksen

mukaisempaa, että tilaaja ja toimittaja istuvat saman pöydän ääreen keskustelemaan. Sen sijaan, että toteutustiimin jäsenet tekevät heille määrättyä järjestelmän osaa yksin, kehotetaan ketterissä menetelmissä jatkuvaan dialogiin toteutustiimin kesken. Goodpasture kirjoittaa kirjassaan, että keskustelu ja väittely henkilöiden kesken tunnistetaan hyväksytyksi toimintatavaksi monelle muodolliselle dokumentille. Dokumentointi on yhä tarpeellista ja sen merkitys saa kasvaa, kun projekti laajenee. (Goodpasture 2010,10). Hän siis painottaa molempia muotoja tasapuolisesti. Päivittäiseen työskentelyyn liittyvissä asioissa riittää keskustelu kollegan tai tilaajan edustajan kanssa, mutta suurissa muutoksissa projektin sisällä on tärkeää päivittää myös dokumentaatio ajan tasalle. Myös Cobb kirjoittaa heti kirjansa alussa miten dokumentaatio ketterissä menetelmissä ymmärretään usein. Se ymmärretään kurittomana menetelmänä, jossa vain kirjoitetaan koodia ilman suunnittelua, dokumentointia tai mitään kurinalaista metodologiaa siitä, miten se kuuluu tehdä. (Cobb 2011, 1). Näin ei tietenkään ole. Ketterät menetelmät vaativat usean kirjoittajan mukaan huomattavasti enemmän tekijöiltään kuin vanhat menetelmät. Ketterät menetelmät vaativat yhteistyötä ja hyvää kommunikointia tiimin kesken sekä tilaajan ja toimittajan välillä.

Ketterän kehitystavan käyttämisessä voi myös ilmetä ongelmia, jos yrityksessä on paljon vanhoja kerrostuneita järjestelmiä ja vanhoja järjestelmäratkaisuja. Ketterä kehitystapa mielletään perinteisesti uusien sovellusten tekemiseksi uusilla sovelluskielillä ja teknologioilla. Nämä asiat voivat jonkun mielestä sopia huonosti ketterien menetelmien ajatukseen siitä, että uusia palasia sovelluksesta viedään tuotantoon jatkuvasti. Tutkimassani yrityksessä on paljon vanhoja kerrostuneita järjestelmiä, jotka vaativat usein päivitysten yhteydessä versiointia. Versioinnilla tarkoitetaan, että järjestelmiin liittyvät sovellukset viedään kertarysäyksellä tuotantoon sovitun viikonlopun aikana. Miten tämä siis sopii ketterään ajatteluun? Ketterät menetelmät eivät sulje pois versioajattelua sovelluskehityksessä. Schuh toteaa kirjassaan, että ketterän tiimin tehtävä on tuottaa sovitusti sovelluksen palasia, joita testataan ketterien menetelmien mukaan jatkuvasti. Sovelluksen valmistuessa pyrähdyksen päätteeksi se voidaan viedä tuotantoon, mikäli se on osa järjestelmäversiota tai se voi jäädä odottamaan

hyllylle (Schuh 2004, 22). Ketterät menetelmät ovat siis luonteeltaan joustavampia kuin vanhat menetelmät, joissa etukäteissuunnittelu määritti hyvin tarkkaan aikaikkunan tuotantoonsiirroille. Ne antavat enemmän mahdollisuuksia ajoittaa, milloin eri palaset viedään tuotantoon. On kuitenkin syytä huomata, että järjestelmäriippuvaisuuksista johtuen on melko harvinaista, että sovellus valmistuu paljon ennen versiopäivää ja jää vain odottamaan lupaa tuotantoon siirtoon.

Tutkimassani yrityksessä on tiukka versioaikataulu, joka suunnitellaan tarkasti yleensä vuoden alkupuolella. Tähän "versiokelloon" pyritään sovittamaan kaikki vuoden aikana suunnitellut järjestelmätyöt. IT -osaston tehtävä on laskea, kuinka suurilla resursseilla kukin versio toteutetaan ja millaisia töitä niihin mahtuu. Useasti käy niin, että tiettyyn versioon saattaa jäädä ylimääräistä kehitysaikaa, muttei se riitä kokonaisen sovelluksen tekemiseen. Tällöin pitää pohtia, voidaanko joitain toisista sovelluksista riippumattomia osia toteuttaa, testata ja viedä tuotantoon tulevaa järjestelmäversiota ajatellen. Usein näin pystytään tekemään. Tämä vähentää sovelluskehityksen tyhjäkäyntiä ja parantaa tehtävän työn laatua, koska sovelluksen osat, jotka on tehty aiemmin, tulevat testattua kahteen kertaan. Ensimmäisen kerran silloin, kun sovelluksen palanen viedään tuotantoon, muttei oteta käyttöön. Toinen testauskierros tapahtuu, kun sovellusta ollaan ottamassa käyttöön esimerkiksi seuraavassa versiossa.

4 HAASTATTELUJEN JA TUTKIMUKSEN TOTEUTUS

Tutkimuksen ketterän kehityksen käyttöönotosta tutkimassani yrityksessä toteutin puolistrukturoituina haastatteluina loppuvuoden 2014 aikana. Tilastokeskus määrittelee strukturoidun haastattelun seuraavanlaisesti: "Strukturoitu haastattelu tai paremminkin strukturoitu haastattelulomake tarkoittaa lomaketta, jossa kysymykset ja niihin tulevat vastausvaihtoehdot on rakennettu etukäteen tarkasti. Sen enempää haastattelijalla kuin vastaajallakaan ei ole lainkaan vapausasteita tulkinnoille. Sekä kysymys että siihen tulevat vastausvaihtoehdot on annettu ennalta." (https://www.stat.fi/virsta/tkeruu/04/01/) Toteuttamani haastattelut erosivat hieman strukturoidusta haastattelusta siten, että en ollut määritellyt valmiita vastauksia kysymyksiin. Haastattelutyyppi sopi tähän tutkimukseen hyvin tutkimusmuotona, koska lopputulosten kannalta haastattelun kokonaisteeman pystyi helposti jakamaan kolmeen eri osa-alueeseen, jotka olivat:

1. Vanhan kehitystavan käyttökokemukset
2. Uuden kehitystavan käyttökokemukset
3. Ketterän kehitystavan käyttö päivittäisessä työssä (ylläpito tai muu päivittäinen työ)

Tutkimassani yrityksessä oli jo vuosia ollut käytössä sisäinen projektinhallinnan työkalu, jolla hallinnoitiin koko projektin elinkaarta. Kyseinen työkalu oli kehitetty vanhan sovelluskehitysmallin aikana, joten nyt haluttiin selvittää, miten uusi kehitysmalli soveltuu yhteen vanhan projektinhallinnan työkalun kanssa ja miten kahden erityyppisen työkalun yhteen sovittaminen oli onnistunut. Projektinhallintaprosessiin ei tehty mainittavia muutoksia ketterien menetelmien käyttöönoton yhteydessä.

Toinen kiinnostuksen kohde oli selvittää, miten tutkimassani yrityksessä eri projektin vaiheissa työskentelevät henkilöt kokivat uuden kehitysmallin käyttöönoton ja millaisia kokemuksia kehitysmallin puitteissa työskentelystä oli saatu. Samalla haluttiin saada jonkinlaista käsitystä siitä, miten uusi kehitysmalli koettiin verrattuna vanhaan kehitysmalliin.

Tämän lisäksi haluttiin tietoa siitä, onko tutkimani yrityksen työntekijöillä yleensä selvyyttä siitä, miten ketterät kehitysmenetelmät toimivat, miten niiden käyttö on muuttanut kehitysprosessia ja onko työntekijöillä jonkinlaisia väärinymmärryksiä ketterien menetelmien käytöstä. Mainitsin jo aiemmin kappaleessa 3.6 mahdollisia väärinymmärryksiä ketteristä menetelmistä. Tällaisia väärinymmärryksiä saattaa syntyä tahattomasti, kun menetelmien kokonaiskuvaa ei ymmärretä kokonaisuudessaan ja saadun tiedon perusteella pyritään hahmottamaan, mistä uudessa kehitysmenetelmässä on kyse. Kuitenkin aina siirryttäessä uuteen kehitysmenetelmään on odotettavissa jonkin verran muutosvastarintaa, joka voi kärjistyessään aiheuttaa jopa tahallisia väärinymmärryksiä.

Toinen mahdollisuus väärinymmärryksiin on yrityksissä, joissa on pitkät perinteet jonkin kehitysmenetelmän käytöstä. Uuden kehitysmenetelmän käyttöönotto voidaan kokea haastavaksi ja uuden opettelu aikaa vieväksi aikatauluherkässä sovelluskehitysmaailmassa. Tällöin saatetaan helposti sortua tekemään asioita niin kuin niitä tehtiin vanhan kehitysmenetelmän aikaan, koska menetelmällä työskennelleet kokevat, että pystyvät paremmin hahmottamaan kokonaisuuden ja pysyvät paremmin aikataulussa käyttäen vanhaa menetelmää. Tietyn tyyppiset henkilöt saattavat myös kokea uuden kehitysmallin uhaksi omalle työlleen tai kokevat työskentelyn uudella kehitysmenetelmällä huonommaksi kuin vanhalla menetelmällä.

Ketterien menetelmien käyttöönotto aiheutti myös tutkimassani yrityksessä uusien prosessiroolien käyttöönottoa. Kuten aiemmin, ketterissä menetelmissä on käytössä tiettyjä toisiaan tukevia rooleja, jotka ovat oleellinen osa prosessia. Tuoteomistaja liiketoiminnan puolella vastaa siitä, mitä kehitystiimi alkaa toteuttamaan tuotteen kehityslistalta ja Scrum Master toimii tiimin töiden mahdollistajana ja tuoteomistajan vastaparina toimittajan puolella. Tutkimuksessani halusin myös selvittää, miten näiden kahden uuden roolin käyttöönotto on luonnistunut niihin valituilta henkilöiltä ja miten he kokevat roolinsa uudessa kehitysmallissa.

4.1 Kysymyskategoriat

Kategorioiden asettelun avulla pyrin hahmottamaan, millaisia kokemuksia tutkimani yrityksen eri tasoilla työskentelevillä henkilöllä oli vanhasta kehitystavasta ja millaisena he kokevat uuden kehitystavan. Kategorioiden asettelulla ei pyritty hakemaan vahvasti käyttäjien omia mielipiteitä kehitysmallien toimivuudesta tai siitä, kumpi kehitysmalli oli heidän mielestään toimivampi, vaan pyrin saamaan haastateltavan kuvaamaan parhaiten, millainen prosessi aiemmin oli ja millaisena he näkevät uuden prosessin. Kategoriat valitsin myös siksi, että koin niistä saatavan aineiston olevan hyödyllistä myös tutkimani yrityksen sisällä johtopäätöksissäni. Kategoriat ja kysymykset kävin läpi tutkimani yrityksen yhteyshenkilöni kanssa ennen haastatteluiden aloittamista.

4.2 Haastateltavien valinta

Haastateltavat valitsin siten, että pyrin saamaan mukaan haastateltaviin jonkun jokaisesta projektin työvaiheesta, sekä siten, että mahdollisimman monet eri roolit projekteissa olisivat edustettuina. Mahdollisimman monen roolin mukaan ottaminen mahdollisti sen, että sain kattavan kokonaiskuvan siitä, miten ketterän menetelmän käyttöönotto oli työskentelytapoja ja prosesseja muuttanut ja millä tavalla haastateltavat kokivat työskentelytavan käytön. Samalla pystyin vertailemaan eri organisaation osissa työskentelevien henkilöiden näkemyksiä siitä, miten ketterien menetelmien käyttöönotto toimi tutkimassani yrityksessä yleensä. Oleellista oli myös, että haastateltavilla oli kokemuksia tutkimani yrityksen vanhasta kehitystavasta ja että he työskentelivät tällä hetkellä uuden kehitystavan kanssa.

Aihe itsessään oli kaikille haastateltaville tuttu ja ketterien menetelmien käyttöönotto oli tuonut joitain uusia aspekteja jokaisen haastateltavan päivittäiseen työhön ja projekteissa työskentelyyn, joko muuttuneiden työtapojen johdosta tai uusien roolien omaksumisen näkökulmasta.

Haastattelutilaisuuden itsessään pidin epäformaalina haastatellen yhtä henkilöä kerrallaan pystyäkseni itse keskittymään aiheeseen ja myös siksi, että haastateltava voisi keskittyä vain esitettyihin kysymyksiin. Haastattelut toteutin kasvokkain, koska koin, että rauhallisen tilan varaaminen haastattelua varten helpotti itseäni sekä haastateltavaa keskittymään haastattelutilanteeseen parhaiten ja antoi mahdollisuuden pohtia vastauksia kysymyksiin kaikessa rauhassa. Yksinhaastattelu sopi myös parhaiten tähän tarkoitukseen, koska ajattelin, että kahden hengen haastatteluissa voi toinen osallistuja vaikuttaa toisen haastateltavan vastauksiin tai haastateltavat saattaisivat johdatella toinen toisiaan. Haastattelujen pituus oli noin 30 minuuttia, jonka aikana sain hyvin käytyä läpi aihealueet.

4.3 Aineiston keruu

Aineiston keruun päätin toteuttaa nauhoittamalla haastattelut ja hoitaa niiden purkamisen myöhemmin säästääkseni aikaa ja pystyäkseni keskittymään haastattelutilanteeseen parhaimmalla tavalla. Mikäli olisin pyrkinyt kirjoittamaan haastateltavien vastaukset samanaikaisesi haastattelun aikana, olisi haastattelutilanteeseen tullut tarpeettomia keskeytyksiä ja tilanteita, joissa olisin joutunut palaamaan tiettyihin aihekohtiin todennäköisesti uudestaan. Ennen haastattelun alkua kysyin jokaiselta haastateltavalta erikseen suostumusta haastattelun nauhoitukseen, johon kaikki suostuivat.

Ennen haastattelun alkua kerroin haastateltaville, mikä haastatteluni aihe oli ja millainen teemahaastattelun näkökulmasta oli haastattelun tavoite. Haastateltavat vastasivat esitettyihin kysymyksiin hyvin omien kokemustensa perusteella avoimesti ja vastaukset kysymyksiin vaihtelivat tutkimuksen kannalta sopivasti, vaikka paljon yhteneviä näkemyksiä oli havaittavissa. Erilaiset näkökulmat tulivat hyvin esille riippuen haastateltavan työtehtävästä organisaatiossa. Haastattelutilanteet pysyivät teemahaastattelun näkökulmasta hyvin vapaamuotoisina. Välillä haastateltavasta riippuen aiheesta voitiin poiketa myös hieman.

4.4 Tutkijan oma asema ja sen vaikutuksen tiedostaminen

Työskentelen itse yrityksessä sovellusasiantuntijana ja haastateltavien kanssa olen päivittäin tekemisissä ammatillisesti. Suurin osa haastatelluista on minulle tuttuja henkilöitä. Pyrin eliminoimaan haastateltavien ohjaamisen tai omien näkemysteni painottamisen haastatteluissa, seuraamalla tekemääni haastattelurunkoa tarkasti, kysyen haastateltavilta heidän toimenkuvaansa relevantteja kysymyksiä. Haastattelujen purkamiseen olen käyttänyt suoria lainauksia haastateltavien kommenteista, kun viittaan vain yhden henkilön vastukseen tiettyyn kysymykseen.

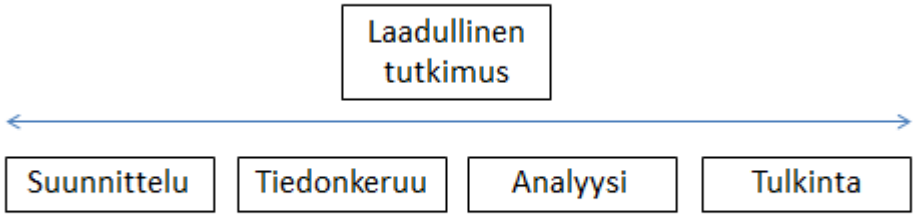
Haastattelun päätteeksi kerroin haastateltaville, että pyrin kysymyksillä hahmottamaan tutkimani yrityksen koko kehitysputken sekä vanhan että uuden kehitysmallin osalta ja mitä muutoksia päivittäiseen työhön uuden kehitysmallin käyttöönotto oli heidän arjessaan tuonut ja olinko onnistunut siinä. Samanaikaisesti haastateltavat saivat antaa palautetta siitä, kokivatko he, että jotain aihealuetta voisi painottaa enemmän tai jonkin aihealueen kysymyksiä lisätä. Palautetta sain jonkin verran, mutta pääsääntöisesti haastateltavat olivat tyytyväisiä kysymyksiin ja siihen, millaista tietoa niillä pyrittiin keräämään.

Vaikka haastattelujen tarkoituksena ei ollut kartoittaa haastateltavien mielipiteitä kehitysmallien paremmuudesta, toivat he usein esille selkeästi mielipiteensä siitä, kumpi kehitysmalli toimi heidän mielestään paremmin tai huonommin tai mitkä osat kummassakin kehitysmallissa olivat nykyään paremmin ja mitä osia ei olisi tarvinnut muuttaa.

4.5 Aineistoanalyysi

Tekemäni tutkimus kuuluu laadullisiin tutkimuksiin. Laadullisella tutkimuksella tarkoitetaan "mitä tahansa tutkimusta, jonka avulla pyritään "löydöksiin ilman tilastollisia menetelmiä tai muita määrällisiä keinoja". (Kananen 2012, 29) Kvalitatiivisen menetelmän valitseminen johtuu Kanasen mielestä usein siitä, että

ilmiöstä ei tiedetä paljoa (Kananen 2012, 29) Laadullinen tutkimuksen prosessi voidaan hänen mukaansa jakaa neljään vaiheeseen.



Kuva 10. Laadullisen tutkimuksen prosessikaavio (Kananen 2012, 93)

Suunnitteluvaiheessa pyritään Kanasen mukaan määrittelemään mm. tutkimusongelma, joka alkaa ongelman havaitsemisella ja määrittelemisellä. (Kananen 2012, 63) Kvalitatiivisen tutkimuksen tiedonkeruumenetelmistä Kananen kirjoittaa, että niistä yleisimmät ovat havainnointi, teemahaastattelu ja erilaiset dokumentit. (Kananen 2012, 93) Kananen kirjoittaa aineiston analysoinnista, että siitä voidaan etsiä esim. rakenteita, prosesseja ja malleja, mikä saavutetaan esimerkiksi lukemalla aineisto useamman kerran ja miettimällä, mitä teksti haluaa sanoa ja mitä se tarkoittaa. (Kananen 2012, 113) Neljänneksi Kananen mainitsee laadullisen tutkimuksen tulkinnan. Hän kirjoittaa, että laadullisen tutkimuksen tulkinta tarkoittaa sitä, että tutkijan on saava selville mikä on tiivistetyn aineiston sanoma, mikä voi olla esim. jokin aineistosta löytyvä rakenne. (Kananen 2012, 115). Nämä vaiheet läpikäymällä tutkija pyrkii hahmottamaan ratkaisun tutkimusongelmaan ja esittelemään sen tulkinnan valmistuttua.

Tutkimukseni haastatteluaineiston käsittelymenetelmänä käytettiin vastausten kategorisointia. Kategoriat sisältävät tietojärjestelmäprojekteille tyypillisiä vaiheja sisältöluokkia. Näitä teema- ja sisältöluokkia edustavat tutkimusosion alaotsikot sekä vanhan, että uuden kehitysmallin osalta. Tutkimus ei tähdännyt teoreettisen vaan käytännölliseen kontribuutioon, joten siinä ei kehitetty näitä sisältöluokkia yleisempiä teema- tai sisältöluokkia. Johtopäätöksissä esitelty vanhan

vesiputousmallin ja uuden ketterän kehitysmallin vertailu todennäköisine syineen ja seurauksineen edustaa jossain määrin tällaista ylemmän tason teemasisältöjen löytämiseen pyrkimistä. Lukija voi havaita tämän vertaamalla liitteen 1 tutkimuskysymyksiä tutkimustulosten 6-7 teemoihin ja sisältöluokkiin sekä johtopäätös- ja suositusosion lyhyeen taulukkovertailuun.

5 TUTKIMUKSEN TULOKSET

5.1 Johdanto

Tässä osiossa raportoin tekemäni haastattelut ja kerron niiden tulokset. Luku on jaettu kolmeen eri alalukuun tekemieni haastatteluiden osa-alueiden mukaisesti. Ensimmäisenä kerron, miten haastateltavat kokivat vanhan kehitysmallin käytön ja millaisia kokemuksia heillä oli työskentelytavasta. Tämän jälkeen siirryn uuden kehitysmallin käyttökokemuksiin ja uuden kehitysmallin vaikutuksiin haastateltavien päivittäisessä työssä. Viimeisenä esittelen johtopäätökset ja suositukset, jotka voidaan tehdä haastateltavien vastausten perusteella.

Haastatteluiden purun hoidin kuuntelemalla kaikki nauhoitteet läpi ja kirjaamalla samalla haastateltavien vastauksia kysymyksiin yhteen Excel -tiedostoon. Excelin jaoin kolmeen osaan haastattelun osa-alueiden mukaisesti. Tällä tavoin sain parhaiten kirjoitettua haastattelun tulokset tutkimusraporttiin käsittelemällä yhden osa-alueen kerrallaan.

5.2 Vanhan kehitystavan käyttökokemukset

5.2.1 Vanha kehitysmalli ja aikataulutus

Yrityksessä on ollut aiemmin käytössä hieman muokattu vesiputousmalli ennen ketterien menetelmien käyttöönottoa. Jokainen haastateltavistani mainitsi tämän asian tiedustellessani, millainen kehitysmalli yhtiössä oli käytössä ennen ketterien menetelmien käyttöönottoa. Sovellusvaatimusten muotona toimi pitkä kirjallinen dokumentti ja töistä ei usein saanut työmääräarviota sovelluskehittäjiltä ennen kuin kaikki yksityiskohdat oli kirjoitettu ylös määrittelydokumenttiin. Prosessista haastateltavat mainitsivat myös sen, että vaatimusmäärittelyä kirjoitettiin yleensä IT -osaston ja liiketoiminnan kanssa yhdessä. Aikataulutuksien kanssa tekemisissä ollut haastateltava mainitsi myös, että töiden tekeminen oli sekavampaa, koska yrityksessä ei ollut selkeää prosessia siihen, mitä kautta IT

-osasto sai työtilauksia. Tämä taas johti siihen, että IT -osaston tiiminvetäjät joutuivat selvittämään, mitkä työt olivat prioriteettilistan kärjessä. Haastateltu järjestelmäkehittäjä toikin esille, että

> *"työt tulivat hyvin pitkälti tiiminvetäjän kautta, mutta myös osittain sähköpostin kautta"*.

Vanhassa kehitystavassa käytössä ollut IT -tiiminvetäjän rooli oli paljon merkittävämpi IT -tiimin töiden aikataulutuksessa, koska viime kädessä hän keskusteli ja päätti, mitä työtä tiimi alkoi työstää.

Prosessi oli myös paljon työläämpi ja vaati paljon muutakin dokumentaatiota kuin vaatimusmäärittelyn. Yrityksen ulkoisella toimittajalla aiemmin työskennellyt haastateltava kertoi, että "kehitysmenetelmä perustui Tieto -object malliin, joka oli kuitenkin prosessina vesiputousmalli. Etenkin loppuvaiheessa oli erittäin tarkkaa, mitä piti olla missäkin vaiheessa valmiina. Tarkka "checklist". Käytössäoli siis erittäin tarkat tarkastuslistat, jotka tuli täyttää prosessin eri vaiheissa. Hän kertoi myös, että

> *"Projektien statuspalaverit pidettiin. Statusraportit allekirjoitettiin aina IT projektipäällikön ja ohjausryhmän vetäjän toimesta, eli siis formaali prosessi. Varsinainen seuranta oli kuitenkin mutu plus tuntimäärä, joka oli yhtä kuin kulut yhteensä"*.

Prosessi oli siis hyvin byrokraattinen, mikä on yleistä tilaaja - IT -toimittaja suhteissa, joissa toiminnan perustana toimii allekirjoitettu sopimuspaperi.

Myös testausvaihe oli suunnittelulta huomattavasti raskaampi kuin ketterissä malleissa. Ulkoisella toimittajalla aiemmin työskennellyt mainitsikin, että

> *"Testitapausten tekeminen oli paljon muodollisempaa. Templatet oli jokaiseen. Jenkkityyppiset prosessit. Samat tiedot on yhtäkkiä kolmessa tai neljässä kappaleessa. Testitapausten piti olla tosi tarkkaan määriteltyjä. Aika nopeasti huomattiin, että ne tehtiin kevyempinä. Projektipäällikkö ohjasi hyvin tarkasti sitä miten paljon*

> *esim. QCta käytettiin. Document driven development oli käytössä eli vahva dokumentointi"*.

Tällainen käytäntö on yleinen tilanteissa, joissa yrityksellä on ulkoinen IT -toimittaja, koska jokainen muutos merkitsee laskutettavaa aikaa toimittajan osalta. Projektityöskentelystä todettiin myös, että liiketoiminnan vaatimusdokumenttia käytettiin usein hyödyksi varsinaisen järjestelmämäärittelyn tekemisessä. It -tiimin vetäjä totesikin, että

> *"määrityksen lukemisella pääsi melkein testaamaan"*.

Joissain tapauksissa vaatimusdokumentit oli kirjoitettu siten, että niissä oli jo määritelty järjestelmän toiminnallisuuksia. Tällainen käytäntö muodostuu usein yrityksissä, joissa liiketoiminnan resurssit ymmärtävät jonkin verran toteutettavien järjestelmien kokonaisuuksista ja tuntevat niiden käyttöliittymät. Tällöin on luonnollista, että vaatimusta tehdessä mietitään jo mahdollisia käyttöliittymämuutoksia tai rajapintoja.

5.2.2 Järjestelmäversioiden sisällönhallinta

Varsinainen järjestelmätöiden aikataulutus toimi aiemmin haastateltavien mielestä pääsääntöisesti hyvin. Tämä johtui IT -toteuttajan mielestä siitä, että tutkimassani yrityksessä on ollut aina ja on yhä toimiva järjestelmän versiointi, eli tiettyyn versioon tehtiin tietty määrä IT -työtä, joka mitattiin työtunteina. Hän totesikin, että

> *"seuraaviin versioihin voitiin siirtää tavaraa. Lisäyksiä ja poistoja tehtiin. Versioajattelu mahdollisti sen"*.

Sovittua määrää työtunteja ei version osalta lähtökohtaisesti koskaan ylitetty, vaan se oli kiinteä. Mikäli jokin työ vaati vaatimusten osalta laajennuksia tai tarkennuksia, onnistui tämä myös joustavasti eli yrityksessä oli aiemmin myös toimiva järjestelmätöiden muutoshallinnan prosessi. Projekteissa työskennellyt haastateltava kertoi, että

> *"mikäli sovellukseen tai järjestelmään haluttiin tehdä muutoksia toteutuksen aikana, asiasta keskusteltiin. Mikäli koettiin, että muutos oli välttämätön tehdä, niin se yleensä myös tehtiin"*.

Vaadittavan muutoksen painoarvo liiketoiminnan näkökulmasta oli siis avainasemassa, jos se haluttiin mukaan vielä tuotantoon vietävään versioon.

Järjestelmäversioiden sisällöstä päätti versio-ohjausryhmä, johon osallistuivat yhden haastateltavan mukaan

> *"P&P ja jakelun management. Välillä myös kaikkien lajien edustajat olivat yhdessä mukana tietyssä vaiheessa"*.

Tämä työryhmä pyrki keräämään joka vuosi niin sanottua "toivelistaa" sekä jakelulta että tuoteyksiköltä, muttei prosessi ollut kovin toimiva. Ryhmässä mukana ollut haastateltava olikin sitä mieltä, että

> *"yli vakuutuslajien ajattelua ei ollut. Joka vuosi kerättiin toivelista, jossa oli versio-ohjausryhmäkäsittely. Siinä ei kuitenkaan päästy kovin pitkälle"*.

Yksi sovellusasiantuntijoista mainitsi myös, että

> *"päätökset teki isompi päätösryhmä. Työt olivat kuitenkin niin valmiita, että päätösryhmä toimi enemmänkin kumileimasimena"*.

Tämän tyyppinen aikataulutus viittaa vahvasti siiloutuneeseen organisaatioon, jossa eri yksiköt priorisoivat sisäisesti omia kehityshankkeitaan välittämättä muiden yksiköiden kehityshankkeista. Tämä johtaa usein tilanteeseen, jossa määrätyillä resursseilla toteutettavaan järjestelmäversioon ei mahdu kaikkia toiveita ja tällöin priorisointi muuttuu pitkäksi perusteluprosessiksi, jossa jokainen organisaatio perustelee, miksi juuri heidän työnsä on kaikkein tärkein.

5.2.3 Vaatimusten määrittely ja läpikäynti

Vaatimusdokumentin valmistuessa järjestettiin usein käsittely, jossa pyrittiin saamaan vastauksia avoimiin kysymyksiin. Palavereihin osallistunut haastateltava totesi kuitenkin, että

> *"tehtiin pitkiä määrittelydokumentteja, jotka toteutettiin hyvin pitkälti vaatimusten perusteella. Keskustelua oli välillä hieman liian vähän liiketoiminnan kanssa"*.

Tämä taas saattoi johtaa väärinymmärryksiin ja korjauksiin, kun vaatimusta lähdettiin toteuttamaan vaatimusdokumentin mukaan. Vaatimusten käsittelyt sitoivat paljon resursseja prosessiin ja tekivät suunnittelusta aikaa vievää, koska tiettyyn järjestelmäversioon pystyttiin toteuttamaan vain tietty määrä työtunteja vaativia järjestelmätöitä. IT -tiimien vastuulla oli myös antaa tuotetuista vaatimuksista työmääräarviot. Työmääräarvioiden antamisesta yksi kehittäjä mainitsi, että

> *"IT arvioi töiden koon. Arviointi oli aiemmin tarkempaa. Nykyään ei niinkään"*. Hän mainitsi myös, että *"tehtävät kokonaisuudet olivat suurempia ja tämän vuoksi vaikeampia hallita"*.

Ken Whittaker kertoo vesiputousmallin määrittelyistä kirjassaan, joka kuvaa hyvin myös tutkimani yrityksen tilannetta aiemmin. Hänen mukaansa yksi vesiputousmallin määrittäviä tekijöitä on määrittely ja analysointi etupainotteisesti. Tämän lisäksi projektit, joissa käytetään vesiputousmallia nojaavat vahvasti testaukseen ja lopulliseen järjestelmäintegraatioon. Tämä lisää hänen mukaansa projektin riskiä, joka kasvaa projektin kompleksisuuden kanssa, joka taas vaikuttaa kokonaisaikatauluun (Whitaker 2010, 253).

Vaatimusten ja määrittelyjen lisäksi myös muu dokumentointi projekteissa oli paljon muodollisempaa vanhassa kehitysmenetelmässä. Yksi haastatelluista sovellusasiantuntijoista totesi, että

> *"Dokumentointi on tehty Word -dokumentointina. Myös käyttötapauksia on tehty sekä toimintokuvauksia testauksen tueksi, vaikka ne eivät olleet niin hyviä testaajien mielestä".*

Tarkasta dokumentaatiosta huolimatta niiden hyötykäyttö oli hankalaa. Näistä muutama haastateltava kertoikin, että vaikka dokumentaatiota tehtiin paljon, niin kaikki dokumentaatio oli sekaisin ja mikäli jotain dokumenttia tarvittiin, oli sitä usein haastava löytää.

5.2.4 Vesiputousmallin projektisuunnittelu ja työn etenemisen seuranta

Vesiputousmalli painottaa paljon tarkkaa etukäteissuunnittelua, joka on herkkä suunnitteluvirheille. Yrityksessä suunnittelu vesiputousmallisissa projekteissa tehtiin aina hyvin, muttei se aina riittänyt. Yksi haastateltavista kertoikin, että

> *"suunnittelu perustui täysin siihen, että määriteltiin kaikki valmiiksi ja sen jälkeen lähdettiin toteuttamaan. Joskus tuli ongelmia siinä, että maailma oli saattanut muuttua välissä. Kun kaikki kirjoitettiin, saattoi olla myös epäselvää oliko määrittely tarpeeksi tarkka"*.

Tällöin ilmaantui ongelmia projektin toteutuksen loppupäässä, mikä on hyvin tyypillistä vesiputousmallille. Suunnitteluhaasteet näkyivät myös kehitystyön tyhjäkäyntinä. Tutkimassani yrityksessä on erittäin paljon toisistaan riippuvaisia järjestelmiä ja sovelluksia, jotka tarvitsee sujuvan kokonaisprosessin näkökulmasta toteuttaa samanaikaisesti. Etupainotteinen suunnittelu vaati, että nämä rajapinnat pitivät olla tunnistettuina ennen projektin aloitusta, jotta toteutustiimit pystyivät tekemään kehitystyötä sujuvasti. Näin ei kuitenkaan aina ollut ja tästä johtuen

> *"tyhjäkäyntiä oli, koska yli järjestelmärajojen ei osattu hahmottaa rajapintoja"*

sanoi yksi ydinjärjestelmien toteuttajista. Toisaalta toinen IT -toteuttaja oli sitä mieltä, että

> *"tyhjäkäyntiä ei ollut niin kamalasti, koska töitä on aina ollut niin paljon. Joskus on saattanut olla"*.

Eli mikäli projektin eteneminen pysähtyi jonkin teknisen esteen tai määrittelypuutteen takia, tehtiin pois tekemättä jääneitä järjestelmäkorjauksia tai tuotannosta löytyneitä virheitä. Moni haastateltava tunnisti ongelmaksi sen, että eri järjestelmien riippuvaisuuksia ei aina kyetty tunnistamaan etukäteen kokonaisprosessissa. Riippuvaisuuksien tunnistaminen kompleksissa sovellusympäristössä on erittäin tärkeää tilanteissa, joissa pyritään suunnittelemaan etupainotteisesti ja mikäli tätä ei pystytä tekemään, niin tehtävän järjestelmätyön tuntimäärä tulee kasvamaan ja aikataulu venymään.

Työn etenemisen raportointi ja seuranta oli tutkimassani yrityksessä vesiputousmallin käytön aikana selkeästi strukturoitu, mutta prosessi itsessään ei ollut haastateltavien mukaan tehokas.

Yksi liiketoiminnan edustaja totesi, että

> *"jokaiselle projektille asetettiin IT -projektipäällikkö, joka hoiti seurannan sekä IT -osastolle että liiketoiminnalle"*.

Liiketoiminnan puolelta tiedettiin myös kertoa, että

> *"projektin ohjausryhmälle tehtiin säännölliset statusraportit projektin etenemisestä ja mikäli työn etenemisessä havaittiin ongelmia, oli projektipäällikön vastuulla hankkia lisää resursseja tekemään toteutusta"*.

Projektipäällikön vastuulla oli myös pitää säännölliset statuspalaverit ja seurata, miten käytettyjen työtuntien toteuma piti paikkansa. Ajoneuvovakuutusten kehittäjä totesikin, että

> *"aiemmin laskettiin paljon enemmän toteutuneita tunteja, koska IT oli toimittajan roolissa"*.

Työpalavereihin osallistui myös enemmän henkilöitä sekä IT -osastolta että liiketoiminnasta. Palavereja pidettiin säännöllisesti, mutta ne olivat tehottomia eikä niiden pitämiseen ollut selkeää prosessia.

> *"Töiden etenemistä seurattiin, muttei kovin tarkasti. Palavereja pidettiin, muttei selkeää prosessia. Palaverit olivat tehottomia. Saatettiin istua kahdeksi tunniksi alas ja luettiin testaustyökalusta virheenkuvauksia"*

totesi yksi ydinjärjestelmien kehittäjistä. Varsinaisen projektien etenemisen seurannan lisäksi oli siis käytössä paljon työntekijöiden aikaa vieviä prosesseja, jotka olisi voitu käyttää toteutustyöhön tai muuhun suunnitteluun.

5.2.5 Järjestelmien testaus

Sovellus- ja järjestelmäkehityksen yksi tärkeimmistä vaiheista on testaus. Vesiputousmallissa testaus on ajoitettu prosessin viimeiseksi osaksi, ja siksi usein kärsii eniten aikatauluviivästyksistä. Testaaminen sujui aiemmin haastateltavien mielestä pääsääntöisesti hyvin, mutta vastauksissa oli paljon hajontaa. Se kertoo erilaisista toimintamalleista eri sovellusalueilla. Yksi IT -toteuttaja kertoi, että

> *"testitapaukset on tehty aina testaustyökaluun"*.

Toinen IT -toteuttaja kuitenkin totesi, että

> *"testitapauksia ei tehty, vaan testaus oli enemmän muistinvaraista"*.

Testausresursoinnista todettiin, että

> *"testihenkilöiden hankinta oli aina projektipäällikön vastuulla ja isompiin projekteihin löytyi aina testaajat". "Aiemmin oli myös helpompaa saada ylimääräisiä testaajia projekteihin"*,

yksi haastateltavista tiesi kertoa. Testaus on aina ollut vesiputousmallissa haaste, koska se on ajoitettu toteutuksen loppupäähän. Tähän IT -osastolla työskentelevä haastateltava totesikin, että

"testaus jousti aina toteutukseen liittyvissä viivästyksissä".

Myös liiketoiminnassa työskentelevä haastateltava vahvisti tämän. Hän sanoi, että

"*Vesiputousmallissa ongelmat esiintyivät aina loppuvaiheessa"*.

Davis ja Radford kritisoivatkin vesiputousmallia, koska vesiputousmalli olettaa, että suunnitelma voidaan muuntaa toimivaksi ratkaisuksi. Tämä ei heidän mukaansa aina ole kuitenkaan mahdollista, mikä voi selvitä vasta, kun odottamattomia esteitä ilmenee. He jatkavat, että nämä esteet ilmenevät esimerkiksi suorituskykyongelmina ja kasautuvat koko toteutusvaiheen ajan, aina lopulliseen tuotteeseen asti. Nämä ongelmat saattavat ilmetä toteutusvaiheessa tai vasta testausvaiheessa, jolloin käyttäjät näkevät ne ensimmäistä kertaa. Tällöin voi kuitenkin olla liian aikaa vievää tai kallista kehittämisen elinkaaressa palata takaisin ja korjata nämä virheet. (Davis ja Radford 2014, 149). Kuten jo aiemmin todettiin, on vesiputousmalli erittäin herkkä aikatauluhäiriöille. Erityisesti tilanteissa, joissa toteutus on viivästynyt, mikä johtaa testausvaiheen aikavajeeseen. Tällöin on mahdollista, että lopulliseen tuotantosovellukseen pääsee virheitä, jotka ovat erittäin aikaa vieviä ja kalliita korjata myöhemmässä vaiheessa sovelluksen elinkaaren aikana.

5.2.6 Toteutuksen resursointi ja henkilöriskit

Resursointi saattaa vesiputousmallisessa kehittämisessä muodostua haasteeksi, koska vesiputousmallin prosessi on tarkkaan määritelty. Henkilöriski kasvaa tällaisissa tilanteissa ja aikatauluviivästykset aiheuttavat siinä vakavia häiriöitä. Menetettyä aikaa kurotaan vesiputousmallissa yleensä kiinni ottamalla aikaa seuraavasta toteutusvaiheesta, jolloin kokonaisuus kärsii. Tutkimassani yrityksessä on resursointi toiminut haastateltavien mukaan hyvin vesiputousmallin käytön aikana. Yksi IT -osaston haastateltava totesi vesiputousmallin resursoinnista, että

> *"vesiputousmalli oli helppo toteuttajien kannalta, koska vastuut ovat selkeät ja kuka tekee mitäkin".*

Vesiputousmallissa ei siis ole kuvattuna osaamisen jakamiseen liittyviä elementtejä. Tämä tarkoittaa käytännössä, että samat henkilöt voivat toteuttaa samoja järjestelmän osia projektista toiseen, jolloin he erikoistuvat tiettyyn järjestelmän osaan. Resursointi on tällöin tavallaan automaattista, koska purettaessa tehtävää toteutusta konkreettiseksi tekemiseksi ovat tiettyjen osa-alueiden tekijät aina selvillä. Aiemmin toteutustiimin vetäjänä toiminut IT -osaston työntekijä totesikin, että

> *"tiimit resursoivat itse. Esimiehiltä pyydettiin resurssia lisää, jos oli tarvetta".*

Tässä valossa vesiputousmallin tekeminen vaikuttaa erittäin selkeältä ja toimivalta. Tämän tyyppiseen prosessiin liittyy kuitenkin erittäin suuri henkilöriski esimerkiksi sairastapauksissa. Yhden IT -osaston haastateltavan mielestä

> *"sairastapauksissa ei välttämättä tuuraajaa löytynyt. Ihmiset oli kiinnitetty enemmän tiettyihin töihin".*

Myös muut IT -osaston toteuttajat vahvistivat, että sairastapaukset ja kesälomat olivat aiemmin sellaista aikaa, jolloin toteutustyö saattoi jäädä laahaamaan paikalleen, koska vastuuhenkilö oli poissa.

Sama toimintamalli oli käytössä myös liiketoiminnan puolella. Sovellusasiantuntijat olivat osaamisensa puolesta vahvasti tietyn sovelluksen tai järjestelmän osaajia. Liiketoiminnan puolella työskentelevä haastateltava totesikin, että

> *"pyrittiin siihen, että vastuu pyrittiin siirtämään pikku hiljaa, jos koulutettiin uusia, mutta yhdellä henkilöllä ei käytännössä ollut kahta vastuualuetta".*

Liiketoiminnan puolelta todettiin myös, että

"tämä liittyy osittain yrityksen ikärakenteeseen".

Tutkimassani yrityksessä on paljon vuosia osaamistaan tietyllä sovellus- tai järjestelmäalueella kartuttaneita työntekijöitä, jotka ovat tottuneet tekemään tai toteuttamaan määrityksiä omille sovellus- tai järjestelmäalueilleen.

Ketterissä kehitysmenetelmissä pyritään loiventamaan vahvaa henkilöriskiä, koska menetelmien peruslähtökohtana toimii osaamisen jakaminen. Tällöin esimerkiksi sairastapauksissa pystytään toteuttajalle saamaan sijainen melko helposti, jos sairastuneen henkilön toteuttama järjestelmän osa on kokonaisuuden kannalta kriittinen.

Kokonaisuutena haastateltavista suurin osa totesi kuitenkin, että resursointi toimi vanhan kehitysmallin aikana hyvin ja työt saatiin pääsääntöisesti tehtyä ajallaan. Mikäli projektin seurannassa huomattiin, että kaikkea työtä ei saatu tehtyä ajallaan, niin mahdollisuutena oli myös karsia tehtävän työn määrää kaventamalla sovelluksen ominaisuuksia. IT -osaston työntekijä totesikin haastattelussa, että

> *"toteutustiimillä oli aina mahdollisuus tehdä arviointia siitä, ehtiikö jokin toteutustyö versioon".*

Mikäli nähtiin, että se ei ollut mahdollista, siirrettiin työt muutoshallintaan.

Davis mainitsee kirjassaan vesiputousmalliin liittyen, että toimiva muutoshallinta on erittäin tärkeää vesiputousmallisessa työskentelytavassa. Hänen mukaansa tämä johtuu huomattavasta aikaerosta käynnistyksen, kehittämisen ja toimeenpanemisen välillä. Tämän lisäksi hän kirjoittaa, että muutoshallinta korostuu, koska vesiputousprosessi ei vaadi kontaktia liiketoiminnan kanssa elinkaaren aikana (Davis 2013, 216). Toimiva muutoshallinta ei kuitenkaan helpota vesiputousmallisien projektien kulujen kasvamista tilanteissa, joissa sovelluksesta löytyy virheitä toteutuksen aikana. Klimczak kirjoittaakin, että etukäteen ennusta-

minen vesiputousmallissa aiheuttaa sen, että virheen korjaamisen kustannus kasvaa exponentiaalisesti projektin elinkaaren aikana. Virheen korjaamisen kustannus saattaa nousta kymmenkertaiseksi toteutuksen aikana sen sijaan, että se oltaisiin havaittu jo suunnitteluvaiheessa (Klimczak 2013, 121). Muutoshallinnan suhteen onkin otettava huomioon, että sovelluksen ylläpitoon siirtämisen jälkeen tehtävät muutokset ovat aina pois uusien sovellusten toteutusajasta.

Järjestelmien ylläpidossa ei haastateltavien mukaan olekaan koskaan ollut ongelmia. Tuotannon virheiden korjaaminen on ollut aina prioriteeteista ensimmäinen. Mikäli tuotantojärjestelmissä havaittiin virheitä, tulivat ne aina IT -osastolle korjattavaksi korkeimmalla prioriteetilla, kertoi yksi IT -osaston haastateltavista.

5.2.7 Projektien dokumentointi ja jälkiseuranta

Kuten aiemmin on jo todettu vesiputousmallin ennalta määritelty tarkka toteutustapa saattaa aiheuttaa pahoja viivästyksiä projekteissa aikataulujen pettäessä. Nämä viivästykset korvataan yleensä ottamalla aikaa muista prosessin vaiheista, esimerkiksi testausvaiheesta. Kysyin yhtenä kysymyksenä IT -osaston haastateltavilta, miten aikataulun viivästymiset hoidettiin projekteissa vanhan kehitystavan aikana? Vastaukset olivat melko samanlaisia. Vaihtoehtoina oli ylitöiden tekeminen, joita tehtiin paljon enemmän kuin uuden kehitysmallin käyttöönoton jälkeen. Muina vaihtoehtoina mainittiin ominaisuuksien karsiminen versioon vietävästä sovelluksesta, töiden vieminen tuotantoon versiopäivän jälkeen tai koko järjestelmäversion vetäminen takaisin ja tuotannon jatkaminen vanhalla versiolla. Yksi liiketoiminnan edustaja totesi kuitenkin, että

> *"Yleensä hoidettiin viikonlopun tai seuraavan viikon aikana. Koskaan ei jäänyt mitään viemättä".*

Järjestelmäversiot saatiin siis hyvin tuotantoon myös vanhan kehitystavan aikana.

Yksi tärkeimmistä projektinhallinnan työkaluista on sen toteuman arviointi ja vertaaminen kustannushyötyanalyysissa annettuun arvioon. Projektien to-

teumia, onnistumisia ja epäonnistumisia voidaan myös analysoida, ja niitä voidaan hyödyntää tulevien projektien suunnittelussa ja toteutuksessa. Tutkimassani yrityksessä ei ainakaan haastateltavien mielestä ole aiemmin tehty kovinkaan syvällisiä projektien toteumien arviointeja. Pitkään IT -osastolla työskennellyt toteuttaja oli sitä mieltä, että

> *"loppuraportteja on tehty, muttei tietoa onko käytetty hyväksi*".

Haastateltavista vain yksi mainitsi, että

> *"loppuraportteja on joistain projekteista tehty, muttei aina. Esimerkiksi konversiossa näin on tehty, ja sitä myöhäisemmissä konversioissa katsottiin läpi kokemuksia ja koitettiin oppia siitä".*

Muista haastateltavista moni totesi, että dokumentaatiota tehtiin ja tehdään yhä paljon, mutta sen valmistuttua kukaan ei käytä niitä mihinkään. Projekteista saatu oppi onkin yhden IT -osaston haastateltavan mielestä

> *"toteumaseurantaa tehtiin plus - miinus kahdeksan prosentin vaihteluvälillä. Projektin onnistumista ei seurattu. Loppuraportteja tehtiin ja niihin kirjattiin, mitä opittiin, mutta ei nitä käytetty koskaan vaan se oli enemmän hiljaista tietoa".*

Yrityksen sovelluskehityshenkilöstö siis oppi itse välttämään suuremmat sudenkuopat ja tunnistamaan ne erilaisista projekteista.

5.3 Uuden kehitystavan käyttökokemukset

5.3.1 Uuden kehitystavan käyttöönotto

Uusien prosessien käyttöönotto yhtiössä, jossa on ollut kauan käytössä jokin malli tehdä töitä, on aina haaste organisaatiolle. Sovelluskehityksessä isoin haaste organisaatiolle ketterien menetelmien lanseerauksessa on vanhasta

pois oppiminen ja muutosvastarinta. Alan Kelly toteaa kirjassaan, että uudella tavalla tekeminen tarkoittaa yksinkertaisesti sitä, että asioita tehdään eri tavalla kuin aiemmin. Tämä taas tarkoittaa, että on vaikeampaa arvioida kuinka kauan tekeminen kestää, mikä taas johtaa siihen, että ensimmäisellä kerralla tekeminen kestää todennäköisesti kauemmin kuin tehdessäsi sitä vanhalla tavalla. (Kelly 2008, 166). Tällaisessa tilanteessa epävarmuus saattaa kasvaa ja ihmiset kokevat epävarmuutta uutta työtapaa kohtaan.

Tutkimassani yrityksessä ketterien mallien käyttöönotto hoidettiin siten, että ohjelmistoprojekteihin osallistuvalle henkilöstölle koulutettiin hyvissä ajoin ketterien menetelmien teoriaa workshoppien kautta ja kerrottiin millaisesta muutoksesta oli kysymys. Lähestymistapa oli lähtökohtaisesti hyvä. Workshopeissa oli kuitenkin alkuvaiheessa havaittavissa, että niin sanotun "puhtaan ketteryyden" ajatukset määrittelyiden muodosta ja merkityksestä saattoivat aiheuttaa väärinymmärryksiä kuulijoissa. Asian läpiviennistä vastuullinen IT -osaston työntekijä muistelikin, että

> *"neljä vuotta sitten, kun ketteristä kehitysmenetelmistä puhuttiin ensimmäisiä kertoja, minun kerrottiin todenneen, että ketterissä menetelmissä ei tarvitse tehdä dokumentaatiota".*

Tämän ajatuksen hän kumosi ja mainitsikin, että kyseessä on melko tyypillinen väärinymmärrys. Ketterissä menetelmissä arvostetaan dokumentaatiota ja se kuuluu osaksi prosessia. Sillä on vain erilainen rooli kuin vanhoissa kehitysmenetelmissä. Mario Moreira kuvaa kirjassaan dokumentaation merkitystä ketterissä menetelmissä sellaiseksi mitä arvostetaan prosessin, työkalujen ja suunnittelun lisäksi. Näitä konsepteja katsellaan kuitenkin eri näkökulmasta ketterässä kontekstissa. Tällä Moreira tarkoittaa sitä, että niitä ei tulisi automaattisesti tehdä vain prosessin takia. Hänen mukaansa niiden tarve tulee yksilöiden interaktion kautta (Moreira 2010, 54). Tällaisia väärinymmärryksiä ei kuitenkaan enää ole tutkimassani yrityksessä havaittavissa, kun malli on ollut käytössä muutaman vuoden ajan. Dokumentaatiolla on edelleen vahva rooli yrityksessä projektien suunnittelussa ja vaatimusten sekä määritysten tekemisessä.

5.3.2 Uusi kehitysmenetelmä ja aikataulutus

Kysyttäessä haastateltavilta, millainen ketterä kehitysmenetelmä tutkimassani yrityksessä on käytössä, ovat vastukset hyvin samanlaisia. Käytössä on sovellettu ketterä malli, jossa on piirteitä ketteryydestä, mutta puhtaaseen ketteryyteen ei tällä hetkellä tai todennäköisesti tulevaisuudessakaan päästä kovin nopeasti. Käytössä oleviksi ketterän kehitysmallin ominaisuuksiksi mainitaan muun muassa säännölliset seinäpalaverit, toteutettavien osakokonaisuuksien pilkkominen pienempiin osiin ja pyrähdyksen tarkastelu. Tämän lisäksi käyttöön on otettu ketteristä menetelmistä tuttuja rooleja, kuten tuoteomistaja ja Scrum Master. Ketterien menetelmien näkökulmasta tällaisen sovelletun mallin käyttö on toivottavaa. Puhtaaseen ketteryyteen harvemmin päästään ja yrityksen tai organisaation pakottaminen siihen ei ole edes suositeltavaa. David Rico toteaakin kirjassaan, että ketteryydellä on monia eri ominaisuuksia. Näitä ovat esimerkiksi tuotantotiimin kanssa samoissa tiloissa työskentelevät asiakkaan edustajat, pariohjelmointi, testauksen ohjaama toteutus, määritteleminen ja erityisesti julkaisusuunnittelu ovat hänen mukaansa toimivia esimerkkejä. (Rico 2009, 45). Hän kuitenkin painottaa jokaisen näiden kohdalla oikeanlaisia ja oikeanaikaisia painotuksia (Rico 2009, 35 - 41).

Syiksi miksi tutkimassani yrityksessä ei päästä puhtaaseen ketteryyteen, haastateltavat mainitsevat seuraavia asioita: vanhat järjestelmät, toteuttajien kesken on yhä olemassa selkeät vastuualueet, koko organisaatio ei ole ketterä ja eri tuotealueiden tietynlainen siiloajattelu, jossa ei välttämättä seurata kokonaiskuvaa, kun uusia toiminnallisuuksia kehitetään. Kaikki haastateltavat olivat kuitenkin yhtä mieltä siitä, että uusi kehitysmalli on ketterä ja tutkimassani yrityksessä on pyritty ottamaan siitä yritykselle parhaiten sopivat toimintamallit, joiden käyttöönotto on pääosin onnistunut.

Ketterissä malleissa lähestymistapa vaatimusmäärittelyyn ja uusien kehitysideoiden läpivientiin on erilainen kuin perinteisissä menetelmissä. Määritysten merkitys on ketterissä menetelmissä tärkeä, mutta niiden muoto on erilainen kuin vesiputousmallissa. Määrittelyiden osalta yksi IT -toteuttajista totesikin, että

> *"Ideat tulevat vähän raakileina ja arviointi on vähän vaikeaa. Usein niissä on liian karkea taso. Vesiputousmalli oli selkeämpi arvioinnin kannalta".*

Tuoteomistajan rooli ketterissä menetelmissä on töiden aikataulutuksen takia tärkeä kehityshankkeiden läpiviennissä, koska hän vastaa siitä, mitä töitä toteutustiimi lähtee toteuttamaan. Haastateltavat myös sanoivat, että juuri ideasta toetutukseen prosessi on selkeytynyt, koska nyt tehtävään valitut tuoteomistajat hoitavat hyvin töiden aikataulutukseen liittyvät ennakkoselvitykset ja kehitysideoita ei enää juurikaan tule suoraan IT -toteutustiimille. Toisen IT -toteuttajan mielestä tekeminen on selkeämpää, koska työt

> *"menevät Product Ownereiden kautta. Aiemmin niitä tuli enemmän suoraan ITlle"*.

Tuoteomistajan rooli ja sen selkeneminen organisaatiossa ovatkin yksinkertaistaneet dialogiprosessia yrityksessä IT -osaston ja liiketoiminnan välillä.

5.3.3 Järjestelmäversioiden sisällönhallinta

Ketterän kehityksen perusajatuksiin kuuluu muutos ja sen hyväksyminen. Tämän vuoksi toisin kuin vesiputousmallissa ketterissä menetelmissä ei koeta haastavaksi kesken toteutuksen muuttuvia vaatimuksia. Thomas Myer sanoo kirjassaan, että Scrum tunnistaa, että asiakkaat haluavat muuttaa mieltään siitä mitä haluavat. Joskus muutokset ovat subjektiivisia ja joskus arvaamattomia, toisinaan ne aiheutuvat markkinoiden paineesta (Myer 2008, 26). Tutkimassani yrityksessä koettiin muutokset vaatimuksissa kesken toteutuksen haastaviksi, koska ne usein vähensivät testaukseen varattua, aikaa tai aiheuttivat toteutustiimissä ylitöitä. Ketteriin menetelmiin siirryttyään haastateltavat kokivat, että töiden hallitseminen on helpompaa pyrähdysajattelun takia. Yksi sovellusasiantuntija totesikin, että

> *"nykyään toteutustiimi pystyy reagoimaan paremmin muutoksiin, koska työ tehdään pyrähdyksissä, koska tällöin voidaan viedä eri sovelluksissa tehdä rinnakkain eri ominaisuuksia".*

Myös IT -toteuttajat olivat samaa mieltä ja mainitsivatkin, että nykyään töitä on helpompi hallinnoida, jos pyrähdykseen tulee niitä lisää. Ketterät menetelmät antavat myös tuotantotiimille enemmän päätäntävaltaa pyrähdyksiin tehtäviin töihin. Mikäli pyrähdykseen varattu työmäärä on täynnä, on toteutustiimillä mahdollisuus sanoa uusille töille, etteivät ne mahdu enää mukaan. Kaksi IT -toteutustiimin jäsentä sanoikin, että aiemmin heillä oli mahdollisuus sanoa, että työn alla olevaan pyrähdykseen ei enää mahdu töitä. Heidän mukaansa toimintamalli on kuitenkin muuttunut siten, että joitain töitä otetaan työlistalle. Töiden aikataulutukseen on myös tullut muutoksia.

> *"Kokonaisajattelu on tullut paremmin esille. Tuoteyksikön sisällä myös ajattelu on selkeytynyt. Pyrimme yhteiseen näkymään. Kaikki riippuu kuitenkin siitä, mitä ylin johto haluaa painottaa".*

Aikataulutuksessa on siis vahvasti pyritty pääsemään eroon vanhasta siiloajattelusta, jossa eri tuoteyksiköt painottavat omia töitään aikataulutuksessa.

5.3.4 Toteutuksen resursointi ja henkilöriskit

Käytännön toteutukseen on tullut myös paljon muutoksia uuden kehitysmallin myötä. Kaikki IT -osaston järjestelmäasiantuntijat sanoivat, että toteuttajat saavat nykyään valita työnsä, kun aiemmin tiiminvetäjät jakoivat työt toteuttajille ja päättivät, missä järjestyksessä töitä lähdettiin tekemään. Myös töiden pilkkominen pienemmiksi palasiksi on tullut mukaan toteutusprosessiin uuden kehitysmallin käyttöönoton jälkeen. IT -osaston haastateltava kertoi myös, että

> *"töiden jakaminen on helpompaa. Aiemmin tiimin omissa palavereissa jaettiin työt tiiminvetäjän toimesta"*.

Uusi kehitystapa onkin onnistunut karsimaan tarpeettomia palavereja, joihin käytettiin aiemmin aikaa. Toteuttajien itse valitessa työnsä myös yhteisvastuu

on kasvanut, koska tekijä joutuu nyt itse arvioimaan pystyykö hän toteuttamaan valitsemansa työn. Tähän liittyen yksi IT -toteuttaja olikin sitä mieltä, että

> *"nykyään töiden tekeminen on myös toteuttajan kannalta helpompaa, koska voin itse arvioida pystynkö tekemään työn ajallaan".*

Töiden koon arvioimisessa on kuitenkin vielä tehtävää. IT -osastolta kerrottiin myös, että

> *"liian isojen järjestelmän osien tekemisessä on perinteisen mallin peruja, koska vanhassa mallissa tehtiin koko maailma valmiiksi".*

Sujuva ketterä kehittäminen vaatii, että toteutettavat kokonaisuudet pidetään tarpeeksi pieninä, jotta prosessi etenee sujuvasti ja pyrähdykset saadaan toteutettua.

Ketteriin menetelmiin oleellisesti kiinnittyvä osaamisen jakaminen ei toteudu tutkimassani yrityksessä täydellisesti. Yksi IT -toteuttajista olikin sitä mieltä, että heillä

> *"on edelleen käytössä selkeät osa-alueet, jotka ovat tiettyjen kehittäjien vastuulla".*

Toimintamalliin ajaa hänen mukaansa kiire. Ketterien menetelmien käyttöönoton yhteydessä ajoneuvovakuutustiimissä oli pyritty tekemään töitä siten, että selkeitä vastuualueita ei enää olisi, mutta kiire on ajanut tekijät toteuttamaan muutoksia vanhan mallin mukaisesti. Tähän toinen IT -osaston haastateltavista mainitsi, että

> *"työjakoa on mahdollista tehdä, mutta henkilöriski on olemassa ja vastuualueet ovat selkeät. Parannusta on tullut".*

Hänen mielestään tässä suhteessa IT -toteutustiimien henkilöriski on pienentynyt aiemmasta, koska osaamisen jakamista on saatu jonkin verran tehtyä. IT -osaston toinen haastateltava oli asiasta siitä samaa mieltä, että

> *"tietämys on kasvanut, mutta valmius tehdä ei ole kasvanut samassa suhteessa".*

Liiketoiminnan puolen haastateltava taas sanoi, että osa vakuutusjärjestelmiin liittyvistä toteutustöistä voidaan nykyään tehdä myös muissa IT -tiimeissä, jos resurssit ovat varsinaisessa tiimissä tiukilla. Hän totesikin, että

> *"On otettu töitä toiselta sovellusalueelta tehtäväksi".*

Liiketoiminnan puolella osaamisen jakaminen on haastateltavien mukaan onnistunut hyvin. Toisen liiketoiminnan edustajan mielestä hänen yksikössään

> *"Osaamisen laajentamista on onnistuttu tekemään".*

Tämän myös samassa tiimissä työskentelevä liiketoiminnan edustaja vahvisti. Hänen mukaansa

> *"liiketoiminnan puolella on tullut lisää rooleja eli heillä on sovellusasiantuntijan rooli ja tuoteomistajan rooli. Tietämys on varmaan kasvanut sen suhteen, mitä mikäkin työ vaatii. Se mitä voidaan toimittaa, menee koko ajan realistisempaan suuntaan. Osaamista ja kokonaiskuvan hahmottamista on lisännyt myös uusien roolien omaksuminen ketterien menetelmien käyttöönoton yhteydessä".*

Samassa tiimissä työskentelevä sovellusasiantuntija kertoi kuitenkin, että hänen mielestään yksikössä

> *"ei ole osaamista määritellä asioita yli sovellusalueiden. Ei ole aikaa perehtyä ja henkilöriski on aika korkea".*

5.3.5 Ketterän mallin projektisuunnittelu ja etenemisen seuranta

Yksi ketterien menetelmien tärkeimpiä prosessihyötyjä on kommunikaation parantaminen. Kommunikaation parantamiseen liittyy oleellisesti myös turhien pa-

laverien karsiminen ja läpinäkyvyyden lisääminen siten, että kaikki tiimin jäsenet näkevät, mitä muut ovat tekemässä seinäpalaverien kautta. Kokonaisuutena tässä on haastateltavien mukaan onnistuttu tutkimassani yrityksessä. Seinäpalaverit ovat lisänneet läpinäkyvyyttä tekemisessä ja turhista seurantapalavereista on osittain päästy eroon. Liiketoiminnan edustaja olikin sitä mieltä, että

> *"projektipäällikön näkökulmasta seinäpalaverit helpottavat projektin seuraamista, koska ihmiset joutuvat avautumaan enemmän siitä, mitä he ovat tekemässä".*

Seinäpalaverin tärkein tarkoitus onkin saada jaettua tietoa mahdollisimman nopeasti toteutustiimin sisällä. Resnickin, de la Mazan ja Bjorkin mukaan päivittäinen seinäpalaveri onkin 15 minuutin mittainen tapaaminen, jonka aikana keskustellaan koordinoinnista, riippuvuuksista ja esteistä. Palaverin pitäminen lyhyenä pitää tiimin liikkeellä ja estää sitä hukkaamasta aikaa (Resnick, Steve, de la Maza, Michael, Bjork ja Aaron 2011, 27). Vaikka kommunikaatio onkin tutkimassani yrityksessä parantunut, niin IT -osastolla koetaan yhä haastavaksi, että kommunikaatio ei ole vieläkään sillä tasolla, että IT -tiimit tietäisivät, mitä muilla tiimeillä on työn alla. Tämä kertoo siitä, ettei ketterää mallia ole vielä saatu käyttöön kaikilla sovellusalueilla.

Myös projektien kokonaisseuranta on haastateltavien mielestä parantunut kommunikoinnin näkökulmasta, koska vastuut seuraamisesta ja reagoinnista ongelmiin ovat selkiytyneet. Liiketoiminnan edustaja totesikin, että

> *"projekteissa projektipäällikkö seuraa onko projekti aikataulussa. Mikäli jokin asia ei etene niin Scrum Master on vastuussa asian tuonnista esille ja eteenpäin viennistä"*.

Toteutuksen kokonaisuuden etenemisestä ovat vastuussa Scrum Master ja tuoteomistaja.

Ketterien menetelmien perusajatus on, että toteutustiimillä on tasainen työkuorma pyrähdyksestä toiseen. Tämä taas mahdollistaa sen, että tiimi on jatku-

vasti työllistetty ja ylitöitä ei tarvitse tehdä. Tutkimassani yrityksessä tämä ei vielä toteudu täydellisesti. IT -osaston työntekijä kertoi haastattelussa, että

> *"toteutustiimeillä on nykyään mahdollisuus kieltäytyä lisätöistä, jotka tulevat tiimille kesken pyrähdyksen toteutuksen. Mahdollisuutta ei kuitenkaan käytetä hyväksi, vaan ylimääräiset työt tehdään ylitöinä".*

Ylitöiden tekemisen pakollisuus johtuu IT -osaston edustajan mielestä myös

> *"järjestelmän kompleksisuudesta".*

Melkein kaikki haastateltavat kuitenkin totesivat, että toteutustyön tyhjäkäynnin osuus on vähentynyt huomattavasti ketterien menetelmien käyttöönoton jälkeen. Tämä vähentää ylitöiden tarvetta, koska menetettyä toteutusaikaa ei tarvitse kuroa kiinni ylitöinä. Syyksi tyhjäkäynnin vähenemiseen mainitaan kommunikaation ja läpinäkyvyyden parantuminen sekä tästä johtuva prosessin tehostuminen.

5.3.6 Projektien dokumentointi, järjestelemävaatimukset ja jälkiseuranta

Dokumentointi on ketterissä menetelmissä huomattavan erilaista kuin vanhassa vesiputousmallissa. Kuten aiemmin totesin, dokumentoinnin ymmärtäminen ketterien menetelmien näkökulmasta voi olla haastavaa ja onkin yksi väärinymmärretyimpiä ketterien menetelmien prosessin osia.

Tutkimassani yrityksessä sovellusten ja järjestelmien määrittely tehdään yhä hyvin pitkälti vanhan kehitysmallin mukaisesti. Liiketoiminnan edustaja kertoi, että

> *"käytössä on sekalainen malli. Ei puhdasta Agilea. Kompleksisuus estää. Tehdään tarkkoja määrityksiä, jotka IT toteuttaa".*

Yrityksen järjestelmämaailma on niin kompleksinen ja kerrostunut, että on tarkoituksenmukaista tunnistaa etupainotteisesti eri järjestelmien rajapintoja.

Myös Chemuturi ja Thomas tunnistavat dokumentoinnin tärkeyden erityisesti isoissa projekteissa. Heidän mukaansa dokumentoinnilla on etunsa, koska dokumentoitu suunnitelma:

- on muiden luettavissa, mahdollisten siitä pois jääneiden asioiden tunnistamiseksi, mikä voi johtaa suunnitelman parantumiseen.
- se toimii viitteenä projektin sidosryhmille
- sen avulla voi mahdollistaa toteutuksen ohjausta ja suoriutumisarviointia
- sen avulla voi mahdollistaa validointia ja suunnittelun sääntöarvoja, koska se tarjoaa suuntaviivat aikaansaatujen toteutusarvojen vertailemiseen

(Chemuturi, Murali, Cagley ja Thomas 2010, 70)

Kattava dokumentointi ei tietenkään ole tarkoituksenmukaista pienissä projekteissa, mutta isommissa siitä on etua. Liiketoiminnassa työskentelevä haastateltava mainitsi esimerkkinä ketterien menetelmien pilottiprojektin, jota oli vetämässä. Siinä pyrittiin melko puhtaaseen ketteryyteen, mikä tarkoitti määrittelyjen tekemistä tarpeen mukaan. Hän mainitsi, että

> *"kaikki meni hyvin siihen asti, kun asiat muuttuivat kompleksisiksi. Tämän jälkeen ajauduimme ongelmiin".*

Tällaisessa isommassa projektissa olisi siis varmasti ollut tarpeellista tehdä kattavampaa dokumentointia siitä, mitä ollaan tekemässä, jotta pystytään paremmin reagoimaan yllättäviin tilanteisiin. Myös aiemmista projekteista olisi varmaankin ollut hyötyä, jos niitä olisi käytetty referenssimateriaalina.

Kokonaisuutena dokumentoinnin ja järjestelmien testitapausten tekemisen taso ja prosessi on haastateltavien mielestä parantunut. IT -toteuttajat sanoivatkin, että nykyään toteutustyön dokumentaatio on osa kokonaisprosessia sen sijaan, että tehdyistä muutoksista pyrittäisiin tekemään dokumentaatiota projektin val-

mistuttua, kuten vanhassa kehitysmallissa. Myös Q&A Specialist -roolin luominen osana ketteriä menetelmiä on parantanut testitapausten laatua. Ketterien mallien kevyemmistä määrittelyistä yksi IT -osaston haastateltavista totesi kuitenkin, että usein

> *"liiketoiminnan vaatimus jää jalkoihin, kun asiat tarkentuvat. Kaikki vaatimuksen asiat eivät välttämättä ole selkeitä toteutuksen alussa. Testitapaukset tehdään. Kun tarkkaa määritystä ei ole, niin testitapaukset ovat hankalampia tehdä".*

Tämä kertoo osittain liian kevyesti laaditusta vaatimuksesta, jota olisi tullut tarkastella tarkemmin ennen sen lähettämistä IT -osastolle arviointiin.

Projektien seurannan ja toteuman arviointiin ei ole haastateltavien mielestä tullut juurikaan muutosta uuteen kehitysmalliin siirtymisen jälkeen. Liiketoiminnan edustaja totesikin, että

> *"Tuossa on kehitettävää. IT -tuntien toteutumista seurataan, mutta ei businesstunteja. Kustannushyötyä ei seurata. Kustannushyötyanalyysin toteuma puuttuu nykyäänkin. Hiljaisella tiedolla pystytään arvioimaan uusia töitä. Projektin jatkoseuranta puuttuu. Siinä on yhä kehitettävää".*

Liiketoiminnan käyttämän ajan seuranta on erittäin tärkeää, jotta resursointia pystytään säätämään tarpeen mukaan. Myös kustannushyödyn arviointi ja seuranta puuttuu projektin elinkaaresta. Yleinen vastaus projektien lopputuloksen arviointiin haastateltavilta oli, että se saatettiin joissain projekteissa tehdä ja toisissa taas ei. Mikäli arviointia tehtiin, niin projektista opittuja kokemuksia ei ainakaan käytetty hyväksi tulevissa projekteissa.

Projektien jälkiseuranta on erittäin tärkeää. Chemuturi ja Thomas kirjoittavat kirjassaan seurannan tärkeydestä. He kutsuvat jälkiraportointia projektin ruu-

miinavaukseksi ja suosittelevatkin sellaisen tekemistä kaikille projekteille riippumatta siitä epäonnistuivatko vai onnistuivatko ne. He jatkavat, että projektin ruumiinavauksen tarkoitus on tunnistaa kaikki sudenkuopat ja parhaat toimintatavat, jotka projektissa ovat tulleet vastaan. Parhaaksi toimintatavaksi he mainitsevat ruumiinavauksen tekemisen jokaiselle projektille (Chemuturi, Murali, Cagley ja Thomas 2010, 220)

Yleinen vastaanotto ketterien menetelmien käyttöönotolle oli haastateltavien vastausten perusteella erittäin positiivinen. Isoimmiksi parannuksiksi mainittiin kommunikaation ja läpinäkyvyyden paraneminen, prosessin selkeytyminen uusien roolien myötä ja se, että ketteryyttä ei ole pakotettu yritykseen, vaan siitä on valittu parhaiten tutkimalleni yritykselle sopivat osat, joista on saatu toimiva kokonaisuus.

6 JOHTOPÄÄTÖKSET JA SUOSITUKSET

Uuden menetelmän käyttöönotto on aina yritykselle iso muutos, joka saa aikaan yrityksen työntekijöissä sekä positiivisia että negatiivisia mielikuvia. Muutoksen johtaminen on yksi tärkeimpiä osaamisalueita uusien asioiden jalkauttamisessa eri organisaatioihin. Sovellusprojektit kattavat yleensä laajuutensa puolesta suuren osan yrityksen henkilöstöä ja tällöin uuden asian kommunikoiminen korostuu. Niinkin radikaali muutos kuin sovelluskehittämisen tavan muuttaminen aiheuttaa sovelluskehityksessä työskentelevien henkilöiden parissa varmasti paljon kysymyksiä.

Yksi yleisimmistä kysymyksistä on varmasti: "Miksi käyttää uutta menetelmää, kun vanha menetelmä toimii ihan hyvin?" Tähän kysymykseen vastaaminen on erityisen tärkeää, kun henkilöstöä lähdetään kouluttamaan, sillä mikäli siihen ei osata vastata on oppimisessa edessä ylämäki.

Keith Morrison kirjoittaa kirjassaan yksilön muutoksen vastaanottamiskyvystä, että se on erilainen riippuen siitä, miten henkilö kokee muutoksen. Sillä, onko muutos yksilön mielestä kiinnostava tai mullistava ja miten yksilö kokee yleensäkään muutoksen, on merkitystä (Morrison 122, 1998). Muutoksen hyväksyminen on siis erittäin paljon yksilön henkilökohtaisista ominaisuuksista riippuvainen.

Yksi ketterän kehityksen pääperiaatteista on prosessissa tapahtuvan muutoksen hyväksyminen. Sovelluskehitysmaailma on vesiputousmallin aikana toiminut hyvin pitkälti ennalta tarkkaan määriteltyjen lainalaisuuksien mukaan. Määrittelyt tehtiin etupainotteisesti ja prosessi ohjasi toimintaa sen päätepisteeseen asti. Ketterissä menetelmissä tällaisessa prosessiohjatussa maailmassa toiminut yksilö joutuu kasvokkain muuttuvien ja elävien vaatimusten kanssa. Tämä aiheuttaa epävarmuutta.

Tekemissäni haastatteluissa kävi ilmi, että vaatimusten taso on laskenut muutaman haastateltavan mielestä ketterien menetelmien käyttöönoton jälkeen. Kyseessä saattaa olla todellinen organisaatiossa tapahtunut muutos, jonka eteenpäin viejänä on ollut ketterien menetelmien ajatus siitä, että vaatimuksen ei tarvitse enää olla kymmenien sivujen mittainen dokumentti, jossa jokainen sovelluksen toiminnallisuus on määritelty kohta kohdalta. Toisaalta kyseessä saattaa olla yksilön oma näkemys, jota hän peilaa kokonaisprosessin muutokseen. Hän saattaa yksinkertaisesti kokea, että työmääräarvion antaminen on uuden kehitysmenetelmän ohjeistuksen mukaan hankalampaa kuin aiemmin ja hänen mielestään ongelma on määrittelyiden tasossa. Kuitenkin yrityksessä tehdään yhä vaatimuksia etupainotteisesti vesiputousmallin mukaan ja oman kokemukseni mukaan niiden taso ei ole mainittavasti laskenut.

Etupainotteisten vaatimusten tekeminen johtuu siitä, että tutkimassani yrityksessä ei ole käytössä puhdasta ketterää kehitysmallia. Kuten haastateltavat itsekin totesivat, kehitysmalli on hybridi, johon on poimittu ketterien kehitysmallien perusteista yrityksen prosesseja parhaiten tukevat elementit, mikä johtuu suurelta osin järjestelmäkerrosten ja rajapintojen monimutkaisuudesta. Vanhoihin järjestelmiin ja niiden aiheuttamaan taakkaan ei pystytä enää yrityksessä vaikuttamaan, vaan niiden kanssa on pystyttävä elämään. Uusien sovellusten kehittämisessä ja suunnittelussa on sen sijaan erittäin tärkeää pyrkiä mahdollistamaan ketterän kehittämisen käyttäminen mahdollisimman tehokkaasti jatkossa. Tämän vuoksi vanhasta poisoppiminen myös sovellusten ja järjestelmien suunnittelussa on tärkeää.

Yksi parhaiten onnistuneista ketterien menetelmien elementeistä on eri roolien käyttöönotto. Tuoteomistaja ja Scrum Master -roolien käyttöönotto on haastateltavien mielestä ollut yksinomaan hyvä asia. Aikatauluttaminen on selkeämpää ja toimivampaa. Toteutustiimeillä on myös rauha tehdä töitä, koska tuoteomistajan vastuulla on kanavoida tehtävät työt tiimeille ja keskustella niistä Scrum Masterin kanssa, rauhoittaen tiimin tekemään sitä, mitä se tekee parhaiten. Näissä kahdessa roolissa olevat henkilöt ovatkin ne, jotka joko mahdollistavat

tai rikkovat ketterän prosessin, ja näiden roolien vahvistaminen ja tukeminen onkin erittäin tärkeää.

Kaikkein tärkeimmäksi elementiksi, mikä on noussut esiin ketterien menetelmien myötä, nostan kommunikaation. Vesiputousmalli tunnistaa kommunikaation ainoastaan dokumentaation muodossa tuotteen kehittämisessä. Dokumentaatio ajaa vesiputousmallista kehitystä ja dokumentaatioon aiheutuneet muutokset korvataan uudella dokumentaatiolla. Ketterissä kehitysmenetelmissä korvataan dokumentaatiota keskustelulla. Dokumentaatiota ei tehdä vain dokumentoinnin takia, vaikka sillä on ketterissä prosesseissa oma roolinsa. Ketterät menetelmät perustuvat lyhyisiin kommunikoinnin sykäyksiin, jotka konkretisoituvat seinäpalaverien muodossa. Seinällä on nähtävissä lyhyellä katsauksella kaikki se, mitä toteutustiimi on tekemässä kyseisenä ajankohtana. Tämä lisää läpinäkyvyyttä ja tehokkuutta, kuten haastateltavista suurin osa totesikin. Seinäpalaverit ovat myös tehneet osan aiemmin käytetyistä projektien seurantapalavereista tarpeettomiksi, koska lyhyellä keskustelulla saavutetaan usein sama lopputulos kuin tunnin mittaisella projektin seurantakokouksella. Tätä kommunikaation tehostumista ja läpinäkyvyyden parantumista pidänkin yrityksessä parhaimpana ketterien menetelmien käyttöönoton saavutuksena.

Työtä on kuitenkin vielä tehtävä erityisesti osaamisen jakamisen alueella. Osaaminen ei kasva ympäristössä, jossa ainainen kiire ei mahdollista yksilöitä kokeilemaan uusia asioita. Onkin huolestuttavaa, että muutama haastateltavista totesi, että menetelmän käyttöönoton alkuvaiheessa oli mahdollista laajentaa omaa osaamistaan, mutta kiire on aiheuttanut sen, että toiminta on mennyt kohti vanhoja vesiputousmallin työskentelytapoja. Ilman osaamisen jakamista ja kehittämistä ei henkilöriski poistu, vaan kasvaa työvoiman ikääntyessä. Ikääntyvien työntekijöiden eläköityessä poistuu tietovaranto heidän mukanaan, jos se on vain hiljaisena tietona heidän päässään.

Projektin jälkiseuranta ja erityisesti kustannushyötyanalyysin arvion seuraaminen toteuman kautta on yksi tärkeimpiä projektin elinkaaren seurannan tehtäviä.

Kustannushyötyanalyysin arvion vertaaminen toteumaan antaa erittäin tärkeää tietoa siitä, miten yrityksessä on onnistuttu arvioimaan kuluja, aikataulua ja ennen kaikkea sitä, vastasiko projektin lopputulos asiakkaan tarpeeseen myynnin näkökulmasta. Vanhoista projekteista saadut opit tulisi dokumentoida ja oppeja tulisi reflektoida tulevien projektien suunnittelussa, jotta niihin käytettyä suunnittelu- ja toteutusaikaa saataisiin kavennettua ja toteutusta tehostettua.

Alla olevassa taulukossa esitetään tiivistetysti tutkimukseni havainnot. Sen avulla avulla pyrin havainnollistamaan tiivistetysti miten asioita tehtiin vanhan kehitysmallin aikana ja miten töiden tekeminen on muuttunut sen jälkeen, kun yritys siirtyi uuteen kehitysmalliin.

Taulukko 1. Johtopäätökset tiivistettynä

Vanha kehitystapa	Uusi kehitystapa
Työ tulivat tiiminvetäjien kautta tai sähköpostilla suoraan toteuttajille	Työt tulevat keskitetysti tuoteomistajien kautta
Projekteissa oli seurantapalaverit, mutta niiden pitämisestä ei koettu saatavan tarpeellista hyötyä tiedottamisen muodossa.	Projektin seurantapalavereja pidetään yhä, mutta ne ovat lyhyempiä ja juoksevat asiat hoidetaan lyhyillä seinäpalavereilla.
Järjestelmätöiden aikataulutus oli haastavaa eikä selkeää prosessia ollut.	Prosessi järjestelmätöiden aikataulutukseen on kehitetty.
Kaikilla toteuttajilla oli selkeät osa-alueet, joita he tekivät. Henkilöriski oli korkea esim. sairastapauksissa.	Osaamisen jakamista on pyritty tekemään ja siinä ollaan osittain onnistuttu.
Määrittelyt tehtiin etupainotteisesti ja pyrittiin kuvaamaan koko muutos kerralla.	Etupainotteisia määrittelyitä tehdään yhä, mutta niihin tulleista muutoksista käydään jatkuvaa dialogia
Toteutettavat kokonaisuuden olivat suuria ja vaikeampia hallita.	Pienet kokonaisuudet antavat mahdollisuuden arvioida itse ehtiikö jonkin työn tekemään
Projektein toteumaa ei seurattu ja vanhojen projektien loppuraportteja käytettiin vain harvoin uusien projektien suunnittelussa.	Projektin toteuman arviointia ei tehdä vieläkään. Uusien projektien suunnittelussa käytetään kertynyttä hiljaista tietoa.

Kokonaisuutena ketterien menetelmien käyttöönotto on onnistunut kohtalaisen hyvin tutkimassani yrityksessä. Ketterät menetelmät antavat sovelluskehitykseen uusia työkaluja ja läpinäkyvyyttä sekä tehostavat toteutusprosessia. On kuitenkin tärkeää, että omaksuttuja toimintamalleja ylläpidetään ja prosessissa työskenteleviä haastatellaan siitä, miten prosessi heidän mielestään toimii. Tällä tavoin voidaan pohtia uusien elementtien käyttöönottoa, jotka on nykyisestä ketterästä mallista jätetty pois.

LÄHTEET

Agarwal B ja Tayal S. 2008. Software Engineering. Laxmi Publications.

Chemuturi M. 2009. Software Estimation Best Practices, Tools, and Techniques : A Complete Guide for Software Project Estimators. J. Ross Publishing Inc.

Chemuturi M C ja Thomas M. 2010. Mastering Software Project Management : Best Practices, Tools and Techniques. J. Ross Publishing Inc.

Clark M. 2004. Business Success Through Service Excellence, Routledge.

Cobb C. 2011. Making Sense of Agile Project Management : Balancing Control and Agility, John Wiley & Sons.

Cooke J L. 2010. Agile Productivity Unleashed, IT Governance.

Coplien J. 2010. Lean Architecture : For Agile Software Development, John Wiley & Sons.

Davis B. 2012. Agile Practices for Waterfall Projects : Shifting Processes for Competitive Advantage, J. Ross Publishing Inc.

Davis B. 2013. Mastering Software Project Requirements : A Framework for Successful Planning, Development and Alignment. J. Ross Publishing.

Davis B ja Radford D. 2014. Going Beyond the Waterfall : Managing Scope Effectively Across the Project Life Cycle. J. Ross Publishing.

Goodpasture J C. 2010. Project Management the Agile Way : Making It Work in the Enterprise, J. Ross Publishing Inc.

Haikala I & Mikkonen T. 2011. Ohjelmistotuotannon käytännöt, Talentum.

Highsmith J. 2010. Agile Project Management; Creating Innovative Products, Pearson Education, Inc.

Holcombe M. 2008. Running an Agile Software Development Project, Wiley.

Kananen J, 2012. Kehittämistutkimus opinnäytetyönä. Juvenes Print.

Kasurinen J P. 2013. Ohjelmistotestauksen käsikirja, Saarijärven Offset Oy.

Kelly A. 2008. Changing Software Development : Learning to Become Agile. Wiley.

Ketterän ohjelmistokehityksen julistus (http://agilemanifesto.org/iso/fi/)

Klimczak E. 2013. Design for Software : A Playbook for Developers. John Wiley & Sons.

Leffingwell D. 2011. Agile software requirements; Lean Requirements Practices for Teams, Programs, and the Enterprise, Pearson Education, Inc.

Mall R. 2013. Fundamentals of Software Engineering. PHI Learning.

Moreira M E. 2010. Adapting Configuration Management for Agile Teams : Balancing Sustainability and Speed. John Wiley & Sons.

Morrison K. 1998. Management Theories for Educational Change. SAGE Publications Ltd. (UK).

Myer T. 2008. Professional CodeIgniter. Wiley.

Remenyi D & SherwoodSmith M. 1999. IT Investment : Making a Business Case. Routledge

Resnick S, de la Maza M, Bjork A. 2011. Professional Scrum with Team Foundation Server 2010. Wrox.

Rico D F & Sayani H H & Sone S. 2009. Business Value of Agile Software Methods : Maximizing ROI with Just-In-Time Processes and Documentation. J. Ross Publishing Inc.

Saddington P. 2012. Agile Pocket Guide : A Quick Start to Making Your Business Agile Using Scrum and Beyond, John Wiley & Sons.

Schuh P, 2004. Integrating Agile Development in the Real World, Charles River Media / Cengage Learning.

Schwaber K. 2004. Agile Project Management with Scrum, Microsoft Press.

Strukturoitu haastattelu (https://www.stat.fi/virsta/tkeruu/04/01/)

Viscardi S. 2013. Professional ScrumMaster's Handbook, Packt Publishing Ltd.

Whitaker K. 2009. Principles of Software Development Leadership : Applying Project Management Principles to Agile Software Development. Course Technology / Cengage Learning.

Wysocki R. 2013. Effective Project Management : Traditional, Agile, Extreme (7th Edition), John Wiley & Sons, Incorporated.

Liite1. kysymykset

Laajat teemat:

1. Vanhan kehitystavan käyttökokemukset.
2. Uuden kehitystavan käyttökokemukset.
3. Ketterän kehittämistavan käyttö päivittäisessä työssä (ylläpito tai muu päi vittäinen työ)

Vanhan kehitystavan käyttökokemukset:

1. Millainen kehitysmalli oli käytössä?
2. Miten kehitysideat saatiin ns. kehitysputkeen?
3. Miten idean eteenpäinvieminen tapahtui, kun se oli hyväksytty kehitysputkeen?
4. Oliko kehitysputki staattinen vai oliko mahdollista, ettei kehitystyötä viety eteenpäin mikäli todettiin, ettei se ole toteuttamiskelpoinen?
5. Voitiinko kehitystyöhön tehdä laajennuksia tai supistuksia tarvittaessa, jos todettiin ne tarpeellisiksi? Miten muutoksiin reagoitiin?
6. Miten hyväksytyt kehitystyöt aikataulutettiin järjestelmäversioihin? Kuka aikataulutuksen teki? Millaisista henkilöistä aikataulutuksen päätösryhmä koostui?
7. Miten resursointi kehitystöihin hoidettiin? ITssä? Liiketoiminnassa?
8. Miten resursointi toimi käytännössä?
9. Miten liiketoiminnan henkilöt valittiin toteuttamaan eri järjestelmämäärittelyjen suunnittelua? Olivatko yksittäiset henkilöt kiinni useissa projekteissa suunnitteluvaiheessa?
10. Miten toteutusta tehtiin käytännössä - Miten järjestelmäsuunnittelijat jaettiin toteuttamaan kehitystyön eri alueita?'
11. Miten työn etenemistä seurattiin?
12. Miten työn eteneminen mahdollistettiin, jos tuli esteitä tai resursointiongelmia?
13. Oliko kehitystöiden tekemisessä tyhjäkäyntiä?
14. Miten ylläpito sujui kehitystyön aikana? Miten virheitä korjattiin?
15. Miten kehitystöiden dokumentointi tehtiin?
16. Miten testitapaukset luotiin?
17. Miten testaus suunniteltiin ja resursoitiin? Kuka teki resursoinnin?
18. Miten mahdolliset viivästykset projekteissa hoidettiin?
19. Miten reagoitiin, mikäli tuotantoon nostovaiheessa havaitaan ongelmia?

20. Miten projektien toteutuma ja onnistuminen arvioitiin ja dokumentoitiin? O llaanko arviointia ja dokumentointia käytetty hyväksi tulevien projektien s uunnittelussa?
21. Muita huomioita?

Uuden kehitystavan käyttökokemukset:

1. Millainen ketterä malli on käytössä (scrum, lean)?
2. Miten kehitysideat saadaan uuden mallin mukaan ns. kehitysputkeen?
3. Miten idean eteenpäinvieminen tapahtuu, kun se on hyväksytty kehitysputkeen?
4. Onko kehitysputki staattinen vai onko mahdollista, ettei kehitystyötä viedä eteenpäin, mikäli todetaan, ettei se ole toteuttamiskelpoinen?
5. Voidaanko kehitystyöhön tehdä laajennuksia tai supistuksia tarvittaessa, jos todetaan ne tarpeellisiksi? Miten muutoksiin reagoidaan?
6. Miten hyväksytyt kehitystyöt aikataulutetaan versioihin? Kuka aikataulutuksen tekee? Millaisista henkilöistä aikataulutuksen päätösryh mä koostuu?
7. Miten resursointi kehitystöihin hoidettaan? ITssä? Liiketoiminnassa?
8. Miten resursointi toimii käytännössä?
9. Miten liiketoiminnan henkilöt valitaan toteuttamaan eri järjestelmämäärittelyjen suunnittelua? Ovatko yksittäiset henkilöt kiinni useissa projekteissa suunnitteluvaiheessa?
10. Miten toteutusta tehdään käytännössä - Miten järjestelmäsuunnittelijat jaetaan toteuttamaan kehitystyön eri alueita? Olivatko yksittäiset henkilöt kiinni useissa projekteissa toteutusvaiheessa?
11. Miten työn etenemistä seurataan?
12. Miten työn eteneminen mahdollistetaan, jos tulee esteitä ja resursointiongelmia?
13. Onko kehitystöiden tekemisessä tyhjäkäyntiä?
14. Miten ylläpito sujuu kehitystyön aikana? Miten virheiden korjaus onnistuu?
15. Miten kehitystöiden dokumentointi tehdään?
16. Miten testitapaukset luodaan?
17. Miten testaus suunnitellaan ja resursoidaan? Kuka tekee resursoinnin?
18. Miten mahdolliset viivästykset projekteissa hoidetaan?
19. Miten reagoidaan, mikäli tuotantoon nostovaiheessa havaitaan ongelmia?

20. Miten projektien toteutuma ja onnistuminen arvioidaan ja dokumentoidaan? Aiotaanko arviointia ja dokumentointia käyttää hyväksi tulevien projektien suunnittelussa?
21. Muita huomioita?

Ketterän kehittämistavan käyttö päivittäisessä työssä (ylläpito tai muu päivittäin en työ):

1. Onko ketterän mallin käyttöönotto tuonut muutoksia päivittäiseen työhön verrattuna vanhaan kehitysmalliin?
2. Onko resursointi muuttunut vanhasta?
3. Onko työtaakka kasvanut tai pienentynyt, kun on siirrytty uuteen kehitysmalliin?
4. Onko oman työnajan allokointi helpottunut tai vaikeutunut, kun on siirrytty uuteen kehitysmalliin?
5. Muita huomioita

Liite 2. haastattelut:

1. Johtaja, liiketoiminnan järjestelmäalueesta vastaava johtaja. (18.11.2014)
2. Manager, IT -osasto, ketterästä kehitysprosessista vastaava esimies. (05.12.2014)
3. Projektipäällikkö, tuoteyksikkö, pitkään tutkimassani yrityksessä toiminut projektipäällikkö, jonka vastuulla oli vetää läpi ensimmäinen ketterä projekti yrityksessä. (1.12.2014)
4. Tiiminvetäjä 1, liiketoiminta, liiketoiminnan järjestelmäalueen tiiminvetäjä, joka on myös vastuussa vakuuttamisalueen kehityshankkeiden aikataulutuksesta yrityksessä. Toimii tuotantosovellusten omistajan roolissa. (24.11.2014)
5. Testauskoordinaattori, IT -osasto, yrityksessä järjestelmien testauksesta vastaava koordinaattori. (26.11.2014)
6. Sovellusasiantuntija 1, liiketoiminta, ajoneuvovakuutuksista vastaava sovellusasiantuntija, joka toimii ajoneuvovakuutuksen tuoteomistajana. (10.11.2014)
7. Sovellusasiantuntija 2, liiketoiminta, myyntisovelluksesta vastaava sovellusasiantuntija, myyntisovelluksen tuoteomistaja, joka oli mukana tutkimani yrityksen ensimmäisessä ketterästi toteutetussa projektissa. (20.11.2014)
8. Sovellusasiantuntija 3, liiketoiminta, asiakasjärjestelmästä vastaava asiantuntija. Tutkimassani yrityksessä pitkään eri sovellusalueilla ja projekteissa työskennellyt asiantuntija. (09.12.2014)
9. Tiiminvetäjä 2, IT -osasto, tuotanto- ja ydinjärjestelmä sovellustiimin tiiminvetäjä ja tiimin Scrum Master. (24.11.2014)
10. It liiketoimintavastaava, IT -osasto, IT- järjestelmien omistaja ja tuotantosovellusten omistajan vastapari ITn puolella. (26.11.2014)
11. Järjestelmäasiantuntija 1, IT -osasto, ajoneuvovakuutussovelluksen ja ydinjärjestelmän kanssa pitkään työskennellyt kehittäjä. (28.11.2014)

12. Järjestelmäasiantuntija 2, IT -osasto, tuotanto- ja ydinjärjestelmän kanssa pitkään työskennellyt kehittäjä 1. (27.11.2014)
13. Järjestelmäasiantuntija 3, IT -osasto, tuotanto- ja ydinjärjestelmän kanssa pitkään työskennellyt kehittäjä 2. (28.11.2014)