Opinnäytetyö (YAMK)

Teknologiaosaamisen johtaminen

YTEJOS14

2019

Niko Mäkelä

TUOTEKEHITYSPROSESSIN TEHOSTAMINEN

– TERVEYDENHUOLLON TIETOJÄRJESTELMIÄ TOIMITTAVASSA OHJELMISTOYHTIÖSSÄ

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (YAMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Teknologiaosaamisen johtamisen Ko.

2019 | 100 sivua, 27 liitesivua

Niko Mäkelä

TUOTEKEHITYSPROSESSIN TEHOSTAMINEN

- Terveydenhuollon tietojärjestelmiä toimittavassa ohjelmistoyhtiössä

Kehittämishankkeen tavoitteena oli terveydenhuollon tietojärjestelmätoimittajana toimivan yrityksen tuotekehitysprosessin merkittävä nopeuttaminen työkuormaa pienentämällä sekä prosessia tehostamalla. Hankkeeseen kuului nykyisen prosessin ongelmien kartoittaminen, sekä ongelmakohtien kehittäminen. Ongelmakohtia tutkittiin mittareilla, jotka tutkivat läpivientiaikaa, laatua, sekä muutosten määrää. Lisäksi tutkimuksessa suoritettiin haastattelu, jolla pyrittiin hakemaan varmistusta oletettuihin ongelmakohtiin.

Tulokset osoittavat, että suurimmat syyt hävikille tuotekehitysprosessin aikana ovat asiakkaan tahtotilan tuotekehitykselle asti saaminen ja kehitystiimin ulkopuolisten toimijoiden prioriteettien kohtaamattomuus tuotekehityksen prioriteettien kanssa. Kiire ja tuoteomistajan hankala tavoittaminen koettiin aikataulun ja tuotteen laadun kannalta ongelmalliseksi.

Tuotekehitysprosessin työstämisessä yhdistetään ketterän kehityksen menetelmiä osaksi perinteisiä ohjelmistokehitysmenetelmiä. Merkittävänä haasteena prosessissa todettiin asiakkaan hankala tavoittaminen, mikä vaikeuttaa heidän sitouttamista prosessiin.

Kehittämishankkeessa tutkittiin ketterien kehitysmallien tarjoamia etuja perinteiseen ohjelmistokehitysmalliin verraten. Tuloksien pohjalta, työssä kannustetaan organisaation laajuiseen ketterän kehityksen kulttuurin valjastamiseen, yhteen projektiin kerrallaan keskittymiseen, tuoteomistajuuden uudelleen määrittämiseen, tilastoinnin lisäämiseen, pienempien ohjelmistoversioiden käyttöönottamiseen, sekä aiemmin tehdyn tehokkaampaan hyödyntämiseen. Hankkeessa ei käyttöönotettu uutta prosessia.

ASIASANAT:

Tuotekehitys, ohjelmistokehitys, ketterä kehitys, agile-menetelmät, lean-ajattelu, terveydenhuoltoala

MASTER'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Technology competence management

2019 | 100 pages, 27 pages in appendices

Niko Mäkelä

OPTIMIZING THE PRODUCT DEVELOPMENT PROCESS

- In a software company providing healthcare information systems

The aim of the development project for the healthcare information system company was to significantly speed up the company's product development process by reducing the workload as well as the lead time. The study included mapping the problems of the current process, and improved problem areas. Problem sections were investigated with indicators that investigate lead time, quality, and number of changes. In addition, the survey conducted an interview, which sought out to confirm the assumed problems.

The results show that the main reasons for the waste during the product development process is the miscommunication between product owner, team and the customer as well as synchronizing the priorities regarding development and other actors.

Pressure and difficulties getting in touch with the product owner was seen as a problem for the schedule and product quality. The development process combines agile methods with traditional software development methods. A major challenge in the process was getting in touch with the clients, making it difficult for them to commit to the process. The study explored the advantages of agile development models compared to the traditional software development models.

Based on the study results, a company-wide agile culture, concentration of one project at a time, redefining the role of product owner, increasing the amount of statistics measuring, a more frequent rhythm of smaller releases, as well as a more powerful usage of previously made projects is encouraged. This study did not introduce a new development process.

KEYWORDS:

Software development, agile methods, lean methods, healthcare

SISÄLTÖ

LIITTEET

KUVAT

KUVAAJAT

TAULUKOT

1 JOHDANTO

Potilastietojärjestelmä on laaja ohjelmisto- ja tietokantakokonaisuus, joka sisältää kaiken potilaan terveyteen ja hoitoon liittyvän tiedon. Toimintaympäristö on haastava, sillä työ sairaalassa on nopeatempoista ja jatkuvasti keskeytyvää. Tuotekehitysprojektin määritteleminen yhdessä suurien organisaatioiden, kuten sairaaloiden kanssa, on ongelmallista, sillä tuotteen tuleekin toimia yhteisesti useamman ison organisaation kesken. Terveydenhuollon tietotekniikan kehitysprojekteissa tulee huomioida, että ohjelman laatiminen ja koodaaminen on vain pieni osa toimivan ohjelmiston kehittämiseen ja käyttöönottoon tarvittavasta kokonaistyömäärästä. Kehitettävään tietojärjestelmään integroitavia potilastietojärjestelmiä on valtava kirjo. Tuotekehitystä tarjoavan toimittajan näkökulmasta, asiakkaan kanssa on usein haastavaa kommunikoida ja tuotteen määrittelyt saattavat elää projektin aikana ajoittain jopa todella paljon. Ympäristöön kaivataan siis ketterää toimijaa, jota taipuu nopeisiin muutoksiin. (Mäkelä 2006)

Terveyden ja hyvinvoinninlaitos (THL) on kirjoittanut keväällä 2018 julkaisemassa artikkelissaan, että terveydenhuollon tietojärjestelmät ovat kehittyneet viime vuosina nopeasti ja niistä on entistä enemmän apua potilaiden hoidossa. He mainitsevat artikkelissaan myös, että järjestelmät auttavat mm. suunnittelemaan, millainen hoito potilaalle sopii parhaiten. Vaikka järjestelmien älykkyyden kerrotaan lisääntyneen selvästi kolmen viime vuoden aikana ja potilastiedon siirto organisaatioiden välillä on lisääntynyt, edelleen on kuitenkin paljon potilasdataa, jota ei syötetä lainkaan tietojärjestelmiin. ja tietojärjestelmien käytössä on edelleen eroja riippuen sairaanhoitopiiristä. Hyvinvointialalla liikkuu paljon tietojärjestelmien toimittajia ja kilpailu on siten kovaa. Tuotekehityksen jatkuva parantaminen on hyväksi yrityksen resurssien käytön kannalta, mutta myös siksi, että avoimille markkinoille tulee päästä ensimmäisenä ennen kilpailijoita.

Tämä kehittämishanke toimii Turun ammattikorkeakoulussa suoritettavan ylemmän korkeakoulututkinnon opinnäytetyönä, Teknologiaosaamisen johtamisen -koulutusohjelmasta. Kehittämishankkeen tavoitteena on terveydenhuollon tietojärjestelmätoimittajana toimivan toimeksiantajan tuotekehitysprosessin merkittävä nopeuttaminen, työkuormaa pienentämällä sekä prosessia tehostamalla. Tuotekehitysprosessia haluttaisiin nopeuttaa jopa 30%. Tehtävänantoon sisältyy tehostavien työtapojen ja menetelmien löytäminen, sekä prosessin muuttaminen tehostuksen aikaansaamiseksi.

Uuden tehostetun prosessin käyttöönottaminen on kuitenkin rajattu kehittämishankkeen ulkopuolella tehtäväksi omaksi projektikseen.

1.1 Toimeksiantaja

Kehityshankkeen toimeksiantajaorganisaatiota, kutsutaan tässä työssä toimeksiantajaksi. Toimeksiantaja on terveydenhuollon tietojärjestelmätoimittaja, jonka tietojärjestelmät on kehitetty hoitopoluilla tapahtuvan hoidon kirjaamiseen ja laadun analysointiin. Yrityksen tavoite on kehittää hoitopolkuja ja parantaa potilaan saamaa hoidon laatua, mahdollistaa hoitodatan analysointi kliinisiin tarkoituksiin, sekä tukea hoitohenkilökuntaa heidän työssään parhaalla mahdollisella tavalla. Yritys on kasvanut tämän kehittämishankkeen aikana pienestä yrityksestä keskisuureksi yritykseksi. Yritys on ISO 13 485 -sertifioitu yritys.

1.2 Tutkimuksessa käytetyt menetelmät

Tutkimuksessa on käytettä sekä kvalitatiivisia että kvantitatiivisia menetelmiä. Kvantitatiivista tutkimusta on käytetty vain lähtötilanteen kartoittamiseen. Mittareiksi kvantitatiiviselle tutkimukselle on käytetty prosessin läpivientiaikoja sekä järjestelmävirheiden (eli bugien) ja määrittelymuutosten lukumäärää. Kvalitatiivinen tutkimus on tehty digitaalisella haastattelulomakkeella ja tutkimus on tehty tiettyjä olettamuksia vasten. Olettamukset pohjautuvat projektien retrospektiiveihin ja mm. kahvipöytäkeskusteluissa käytyihin keskusteluihin. Haastattelussa on käytetty seuraavia olettamia:

- Olettama 1: Tuotekehityksen menetelmät koetaan tarpeellisiksi, mutta niitä ei tehdä akateemisesti oikein.
- Olettama 2: Tuotekehityksen menetelmiä ei ymmärretä.
- Olettama 3: Lean-hävikkityypeistä odotus on tuotekehitysprosessin suurin ongelma.
- Olettama 4: Suurin tuotekehitysprosessissa odotusta aiheuttava ongelma on, että asiakkaan tahtotilaa ei saada tuotua tuotekehitykselle asti.

Haastatteluiden avulla on pyritty saamaan vahvistus asetetuille olettamuksille, jotta tuotekehitysprosessin kehityskohde saataisiin rajattua suppeammaksi.

1.3 Työn rakenne

Työn rakenne noudattaa yksinkertaistetusti seuraavanlaista kaavaa:

1. Minkälaista tuotekehityksen tulisi olla?
2. Millainen toimeksiantajaorganisaation nykyinen tuotekehitysprosessi on ja mitkä ovat sen ongelmakohdat?
3. Miten toimeksiantajan tuotekehitysprosessia tulisi tehostaa?

Työ lähtee liikkeelle teoriaosuudella. Luvussa 2 tutustutaan asiakkaan ympäristöön ja kuvataan millaisia erityispiirteitä ja ongelmia kohdistuu terveydenhuollonalan tuotekehitykseen. Teoriaosuus päättyy lukuun 3, jossa tutustutaan projektinhallintaan, ketterän tuotekehityksen menetelmiin, DevOps-kulttuuriin, sekä verrataan ketteriä menetelmiä perinteisempiin tuotekehitysmenetelmiin.

Teoriaosuuden päätyttyä keskitytään esittelemään toimeksiantajan tuotekehitysprosessia, sekä siihen liittyviä ongelmakohtia. Luvussa 4 käydään läpi toimeksiantajan tuotekehitysstrategian vaikutusta tuotekehitysprosessiin, esitellään itse tuotekehitysprosessi, sekä tuotekehitystä edeltävä määrittelyprosessi ja tuotekehitysprosessia jatkavat testausprosessi ja toimitusprosessi. Luvussa 5 esitellään tutkimuksessa käytetyt tutkimusmenetelmät, sekä esitellään haastattelututkimuksen vastauksia ja siinä tehtyjä havaintoja.

Luvut 6 ja 7 syventyvät siihen, miten toimeksiantajan tuotekehitysprosessia tulisi tehostaa. Luvussa 6 paneudutaan tarkemmin tutkimuksessa esille nousseisiin ongelmakohtiin ja niiden vahvistamiseen haetaan tukea aiemmin esitellystä teoriaosiosta. Luvussa 7 puolestaan tuotekehitysprosessin tehostamiseen esitetään konkreettisia ehdotuksia aiemmin esiteltyjä ongelmakohtia taklaamaan.

Lopuksi tehdään käytettyjen tutkimusmenetelmien ja tulosten luotettavuuden analysointi ja käsitellään myös lyhyesti tehostetut tuotekehitysprosessin käyttöönoton ja jalkautuksen epäonnistuimista.

2 TERVEYDENHUOLLONALAN OHJELMISTOJEN TUOTEKEHITYS

Terveydenhuollonalalla ohjelmistojen tuotekehitys tapahtuu usein varsin perinteisin kehitysmenetelmin. Palvelujen tulee toimia joka päivä vuorokauden ympäri, joten asennuksen aikaiset tietojärjestelmien käyttökatkot tulee pitää mahdollisimman lyhyinä. Terveydenhuollonalan toimijoiden normaalitoiminta ei saa häiriintyä uuden tuotteen käyttöönotosta tai käytössä olevan tuotteen päivityksestä. Usein joudutaan jopa käyttämään uutta ja vanhaa järjestelmää rinnakkain.

Ohjelmistotuotekehitys terveydenhuollonalalla on harvoin ketterää, sillä ohjelmistojen loppukäyttäjät ja tuotekehityshankkeeseen osallistuvat päättäjät ovat kiinni omassa työssään mm. leikkaussaleissa ja poliklinikoiden vastaanotoilla, mistä johtuen heitä on usein vaikeata tavoittaa ohjelmistokehityksen merkeissä. Tavoittamattomuus tekee tietojärjestelemien suunnittelusta aina hankalaa, oli käytettävä projektinhallintamenetelmä millainen tahansa. Työ sairaalassa on hektistä, nopeatempoista ja jatkuvasti keskeytyvää. Esimerkiksi kirurgisen sairaalan leikkausosastolla työskentelee sairaanhoitajia, lääkintävahtimestareita, apulaisosastonhoitajia ja osastonhoitajia. Ortopedian poliklinikalla puolestaan työskentelee mm. erikoislääkäreitä, erikoistuvia lääkäreitä, sairaanhoitajia, perushoitajia, lääkintävahtimestareita, osastosihteereitä, lähettejä sekä sairaala-apulaisia. Kokonaiskuvaa potilaan tilanteesta voi siis olla hankalaa hahmottaa, koska työvaiheet ovat pirstoutuneet eri ammattikunnille. Terveydenhuoltoalan henkilökunnalla ei myöskään ole perinteisesti sisältynyt tietotekniikkaa terveydenhuollon ammattilaisten peruskoulutusohjelmaan, minkä vuoksi tietojärjestelmien käyttö ja niiden kehittäminen voi osoittautua hankalaksi. (Mäkelä 2006, 136; Lähteenmäki 2006, 27)

Kari Mäkelä kuvaa kirjassaan Terveydenhuollon tietotekniikka – Terveyden ja hyvinvoinnin sovellukset, tyypillisen terveydenhuollon tietotekniikan kehitysprojektin etenemisen vaiheet seuraavanlaiseksi:

1. Projekti aloitetaan tehtävän määrittelystä eli pohditaan, mitä halutaan tehdä, muuttaa ja parantaa.
2. Määrittelyä seuraa toimivan ohjelmarungon laatiminen, jossa toivotut tavoitteet ja ominaisuudet pääpiirteissään toteutuvat. Tässä vaiheessa ohjelma ei vielä ole loppukäyttäjälle valmis, vaan se vaatii ammattilaisen apua.

3. Kun ohjelmarunko on esitelty, jatketaan iterointivaiheesta, jonka aikana toimiva ohjelmarunko käydään loppukäyttäjän kanssa läpi ja muokataan sitä hänen toiveidensa mukaisesti. Pieneltäkin tuntuvat muutokset saattavat aiheuttaa suuria muutoksia ohjelmakoodiin, mikä aiheuttaa rutkasti lisätyötä, sillä vanhaa koodia muokattaessa, tulee tehdä selväksi, etteivät uudet muutokset riko ohjelmiston muita toiminallisuuksia.
4. Muutosten jälkeen aloitetaan toimivan kehitysversion koekäyttö, jonka aikana varmistetaan, että ohjelma toimii kuten loppukäyttäjä sen toivoo toimivan.
5. Mikäli koekäytössä tai varsinaisessa tuotantokäytössä ilmenee loppukäyttäjälle uusia toiveita ohjelmalta, aletaan suunnitella ohjelman päivitysversiota, jolloin projekti käydään läpi uudelleen. (Mäkelä 2006, 136)

Kari Mäkelä huomauttaa myös, että terveydenhuollon tietotekniikan kehitysprojekteissa tulee huomioida, että ohjelman laatiminen ja koodaaminen on vain pieni osa toimivan ohjelmiston kehittämiseen ja käyttöönottoon tarvittavasta kokonaistyömäärästä ja että ohjelman lopullinen käyttöönotto terveydenhuollossa kestää huomattavasti kauemmin, kuin ohjelman laajuuden tai tarvittavan ohjelmointityömäärän perusteella voisi päätellä. Päätöksen teon ja määrittelyyn ongelmille ei Mäkelän mukaan yleisesti anneta normaalissa tietojärjestelmäkehityksessä tarpeeksi huomiota, mikä ei välttämättä pienissä hankkeissa haittaa, kun päättäjiä on korkeimmillaankin vain muutamia. Usein, mitä vähemmän on päättäjiä, sitä helpommaksi tulee myös projektin määritteleminen. Tehtäessä tuotetta suuremmalle organisaatiolle, kuten sairaaloille, ongelma korostuu, kun tuotteen tuleekin toimia yhteisesti useamman ison organisaation kesken. Mäkelä painottaa myös että määrittely on joskus jopa mahdotonta ilman iterointivaihetta ja että lähes kaikki tietojärjestelmäratkaisut tarvitsevat jonkinlaisen iterointivaiheen, eikä ilman riittävän hyvin toimivaa alustavaa ratkaisua ole myöskään mahdollista kehittää lopullista täydellisesti toimivaa ratkaisua. Mäkelä kirjoittaa kirjassaan myös komiteaongelmasta, jolla hän tarkoittaa nimenomaan sitä, että mitä suurempi projekti on, sitä enemmän siinä usein on asiantuntijoita, joiden suuri määrä vaikeuttaa päätöksen tekoa. Nyrkkisääntönä hän esittää että yksi kohtuullisesti toteutettu päätös on terveydenhuollon tietojärjestelmähankkeissa n. kymmenen kertaa parempi kuin sata erinomaista mutta toteutumatonta päätöstä. Mahdollisimman montaa asiantuntijaa pitää kuulla, mutta konkreettiset päätökset tulee syntyä pienen komitean toimesta. (Mäkelä 2006, 140)

2.1 Potilasdatan asettamat rajoitteet

Terveydenhuollonalan tietojärjestelmät tallentavat tietokantoihinsa erittäin paljon potilaisiin ja heidän hoitoonsa liittyvää potilasdataa, mikä tekee tallennettavasta tiedosta erittäin sensitiivistä. Tietojärjestelmiin ei voida tällöin tallentaa tietoa, jota siinä ei varsinaisesti käytetä. Jotta potilastietoja ei tarvitsisi syöttää useita kertoja ja useaan eri järjestelmään erikseen, pienikin tietojärjestelmä integroidaan usein useisiin muihin alalla toimiviin potilastietojärjestelmiin. Tiedon on siis oltava mahdollisimman rakenteellista, jotta mahdollistettaisiin sen vertailtavuus eri potilaiden ja eri järjestelmien välillä. Sen on myös oltava erittäin eheätä ja jatkuvasti saatavilla. Terveydenhuollonalan tietojärjestelmää määriteltäessä on siis pidettävä huoli erityisesti datan laadusta ja sen arvosta. Ongelmaksi määrittelyssä tuskin muodostuukaan arvostuksen puute dataa kohtaan, vaan sen rakenteellistamisen vaikeus ja laadun varmistus.

2.2 Asiakasrajapinnan ongelmat terveydenhuollonalan ohjelmistokehityksessä

Asiakasrajapinnan ongelmia terveydenhuollonalan ohjelmistokehitykseen liittyen on tämän hankkeen aikana kartoitettu haastatteluiden avulla (liite 2; liite 3), joiden pohjalta voidaan sanoa, että yksi yleisimmistä ongelmista asiakasrajapinnassa on, että ongelmaa ei ymmärretä. Ongelmiin saatetaan helposti suhtautua liian huolettomasti tai liian rankasti. Suurimmaksi haasteeksi koettiin myös kommunikaatio-ongelma, sillä terveydenhuollon- ja tietotekniikanalan toimijoilla on varsin omanlaatuinen sanastonsa ja kommunikointitapansa. Tuotekehitysprosessiin tämä ongelma vaikuttaa siten, että muutoksia tulee usein jopa selkeäksi ja varmoiksi pidettyihin asioihin.

Kari Mäkelä mainitsi kirjassaan Terveydenhuollon tietotekniikka (2006) päätöksentekokomitean kokoon liittyvistä ongelmista. Haastatteluiden pohjalta (liite 3) päästiin samaan tulokseen Mäkelän esittämien väitteiden kanssa siitä, että mitä suurempi komitean koko on, sitä hankalammaksi yhteisymmärryksen saavuttaminen tulee ja sitä vaikeampaa tulee siten myös määrittelystä. Mitä käytännönläheisemmistä asioista on kyse, sitä suuremmaksi ongelmat muodostuvat. Asiakkaan päässä toimii useita eri toimihenkilöitä useista eri yrityksistä ja laitoksista, sekä useista toimipisteistä ja koulukunnista (sairaanhoitajat, fysiatrit, ortopedit ja neurologit vain muutaman mainitakseni), joilla kaikilla on omia käytäntöjään. Asiakkaan prosesseissa on myös useita eri vaiheita ennen ja jälkeen hoitopäätöksen ja toimenpiteen tekemistä. Kirjavasta käyttäjäkunnasta ja

asiakkaan monimutkaisista prosesseista johtuen usein päädytäänkin tilanteeseen, että asiakas ei osaa määritellä omaa prosessiaan tai kuvata ongelmiaan. Asiakas voi myös keskittyä kehittämiseen liian kepeästi eikä mieti ongelmia, toimintoja ja prosesseja tarpeeksi tarkkaan, jolloin kehitysprojektin aikana luodaan usein isojakin muutostoiveita. Asiakasta on myös vaikeaa saada sidottua tuotteen kehitykseen mukaan, sillä aikataulut luovat usein ylitsepääsemättömän haasteen.

2.3 Tuotekehityksen ongelmat terveydenhuollonalan ohjelmistokehityksessä

Aiemmin mainitut datan asettamat rajoitukset ovat erityisesti tuotekehityksen ohjelmointivaiheessa esille tulevia haasteita. Olemassa olevan datan määrä on valtava, mutta sen tulee kuitenkin olla täydellisen eheätä ja aina saatavilla. Ongelmaksi muodostuu datan suhteen jatkuvat, joskus isotkin muutostoiveet, joihin on usein vastattava asiakasympäristön luonteen vuoksi välittömästi. Datan eheys ja saatavuus on erittäin hankalaa ja paljon työtä vaativaa turvata jatkuvasti muuttuvassa toimintaympäristössä ja ohjelmistoalustassa. Datan pohjalta tehtävä raportointi asettaa myös vaatimuksia datan yhdenmukaisuudelle ja rakenteisuudelle.

Tuotekehityksessä isoksi haasteeksi muodostuu myös muiden toimijoiden tietojärjestelmiin integroituminen, mikä on lähes poikkeuksetta välttämätöntä. Kerätty data on liikesalaisuutta, eikä siitä haluta luopua ilmaiseksi. Yhteistyökumppaneiden aikataulujen synkronoiminen on harvoin helppoa, mikä hankaloittaa jatkuvaa integroitumista. Toimittaminen on myös haastavaa, sillä asiakas haluaa usein omaan toimintaympäristöönsä räätälöidyn ratkaisun tuotteesta. Uutta versiota tuotteesta ei kuitenkaan voida ottaa käyttöön milloin halutaan eri sidosryhmien erilaisista päivitysrytmeistä johtuen.

Koska potilastietojärjestelmän dataa on erittäin paljon ja loppukäyttäjillä on eritasoinen tietotekninen osaaminen, yhdeksi suurimmista haasteista muodostuu myös käyttöliittymä ja sen käyttäjäystävällisyys. Asiakkaan päätelaitteet ja ohjelmistot saattavat olla todella vanhoja, eikä käyttäjäkunta ole välttämättä vielä sopeutunut sähköisiin järjestelmiin.

3 OHJELMISTOJEN KEHITYSMENETELMIÄ

Perinteisissä ohjelmistotuotantoprosesseissa projektin sisältö pyritään lukitsemaan suunnitteluvaiheessa. Perinteiset menetelmät soveltuvat parhaiten kertaluonteisille projekteille, joissa on vain pieni tai kohtalainen määrä etukäteen tuntemattomia muuttujia (Auer 2013; Davis 2012, 75)

Perinteisesti projektihallinnassa on tiettyjä vaiheita, jotka tehdään yleensä samassa järjestyksessä:

1. projektin asettaminen
2. projektin suunnittelu
3. projektin toteutus
4. projektin seuranta
5. projektin päättäminen. (Wysocki 2013, 101)

Perinteisesti ohjelmistoprojektin hankinta aloitetaan suunnittelemalla sille budjetti ja aikataulu, ennen kuin ensimmäistäkään koodiriviä on kirjoitettu. Projektin tärkeimmät kriteerit eli lopputuotteen käyttötarkoitus ja -kelpoisuus, käyttäjätyytyväisyys, sekä arvon tuotto jäävät tällöin huomiotta. (Järvenpää & Kovanen 2018)

3.1 Lean-ajattelu johtamisfilosofiana

Lean-ajattelu on johtamisfilosofia, jonka avulla pyritään poistamaan työhön liittyvät tuottamattomat toiminnot. Lean pyrkii parantamaan asiakastyytyväisyyttä, ja laatua, vähentämään kustannuksien määrää, sekä lyhentämään projektin läpivientiaikoja. Tärkeää on tunnistaa työhön liittyvä hävikki ja pyrkiä tehokkaasti poistamaan sitä. Hävikillä tarkoitetaan tuottamattomia toimintoja jotka hidastavat prosessia ja tuottavat tarpeettomia kustannuksia. Lean-tuotanto on saanut alkunsa Toyotan autotehtaalla jo 1970-luvulla. Leanin etuina ovat asiakaslähtöisyys, jatkuvan kehittämisen periaate, sekä kustannustehokkuus. Kantavana ajatuksena on tuottaa maksimaalisesti arvoa asiakkaalle ja karsia pois turha. Näin myös laatu paranee ja tehokkuus kasvaa, kun projektissa ei tehdä asioita turhaan. (Davis 2012, 30; Cobb 2015, 89; Poppendieck 2007, 25)

Kirjassaan Implementing Lean Software Development: From Concept to Cash (2007) Mary ja Tom Poppendieck kuvaavat millaiset asiat ohjelmistokehitysprojektissa ovat

hävikkiä ja luovat vastoinkäymisiä. Tällaisia asioita ovat mm. sellaiset asiat jotka luovat myöhästymisiä asiakkaalle, mutta jotka eivät tehdessä tai pois jätettäessä kuitenkaan vaikuta lopputulokseen:

- Osittain tehty työ: Sisältää kaiken työn jota ei ole viimeistelty, esim. työn joka ei ole dokumentoitu tai katselmoitu, sekä työt jotka ovat kehitysjonossa täysin aloittamatta. Osittain tehty työ vanhenee ja muuttuu arvottomaksi ajan kuluessa.
- Ylimääräiset prosessit: Prosessissa tehtävä ylimääräinen työ, joka ei tuota arvoa asiakkaalle. Tällaista työtä ovat esim. käyttämätön prosessin vaatima dokumentaatio tai katselmus joka ei tuota lisäarvoa lopputulokseen.
- Ylimääräiset toiminnot: Palveluun tai tuotteeseen tehdyt toiminnot joita organisaatio tai asiakas ei tarvitse, lisäävät testausta ja monimutkaistavat ylläpitoa.
- Tehtävän vaihdot: Kun työntekijä on osoitettu useaan projektiin joihin häneltä edellytetään jatkuvaa projektien ja niiden sidosryhmien hoitamista, kuluu paljon aikaa ja työtehoa tehtävien vaihtamiseen.
- Odotus: Kaikki työn seisauttava odottelu lisää projektin läpimenoaikaa ja syövät projektin arvoa asiakkaan näkökulmasta.
- Liikehdintä: Kaikki vaivannäkö joka nähdään työhön liittyvän informaation tai siihen liittyvän materiaalien liikutteluun. Hävikkiä syntyy esim. kun samaa informaatiota joudutaan jakamaan useaan kertaan. Myös kaikki työn siirtely tekijältä toiselle vaatii aina lisäkommunikointia.
- Puutteet, virheet ja vikatilanteet: Väärin tehty, viallinen, puutteellinen tai epäselvä toiminto, ominaisuus tai informaatio luo projektiin hävikkiä, koska ongelmatilanteiden selvittämiseen tarvitaan aina lisätyötä. Mitä kauemmin aikaa ongelman syntymisestä kuluu, sitä hankalammaksi ongelmatilanteesta elpyminen tulee.
- Epästandardi ja manuaalinen käsin tehtävä työ: Konfigurointi, testiympäristöjen pystyttäminen, käsin asennettavat palvelimet, räätälöidyt tuotteet, jne. luovat hävikkiä, sillä tehty työ ei ole uudelleen käytettävää.
- Sankarillisuus: Jotta maali saavutettaisiin, yksilöille ja tiimille voidaan asettaa epärealistisia tavoitteita, jotka voivat lopulta johtaa suunnittelemattomiin tilanteisiin, ylitöihin, sekä nopeisiin riskipitoisiin päätöksiin.

3.2 Projektikolmio

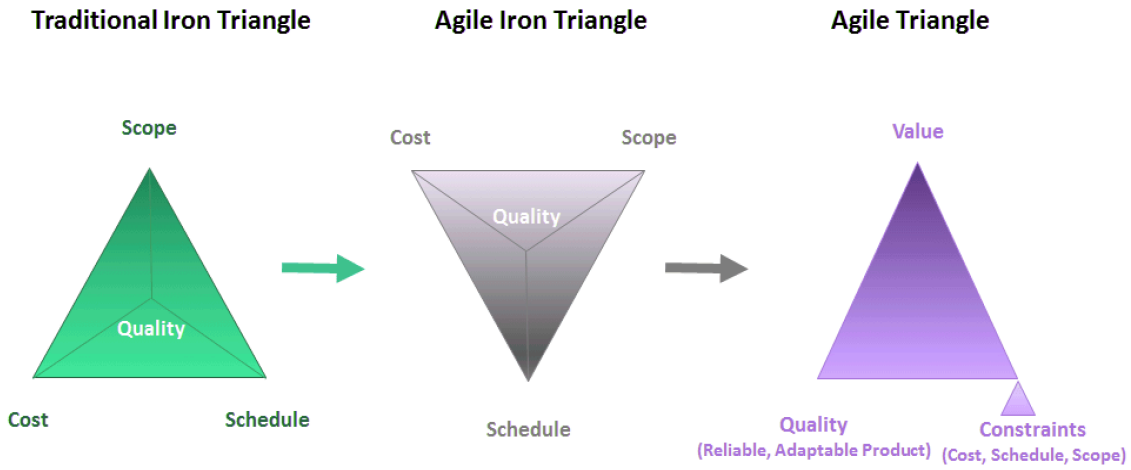
Projektikolmio on projektinhallinnan malli, jonka perinteisessä versiossa aikataulu, resurssit eli kustannukset ja ominaisuuksien määrä eli laajuus, esitetään kolmion sivuina (kuva 1). Usein ohjelmistoprojektia aloitettaessa pyritään lukitsemaan kaikki 3 ulottuvuutta. Projektikolmion on tarkoitus on kuitenkin osoittaa, että käytännössä vain 1 tai korkeintaan kaksi kolmion sivuista saadaan lukittua. Tuotteen ominaisuuksista tai laadusta tinkiminen vähentää tuotteen kykyä luoda arvoa eli tyytyväisiä ja maksavia asiakkaita. Laadusta tinkiminen vaikuttaa myös teknisen velan kasvuun, mikä johtaa myöhemmin ylläpidon ja jatkokehityksen kustannuksia. Ominaisuuksien lisääminen puolestaan pidentää aikataulua. (Highsmith 2010)

Tuotekehitysprojektin ja siinä kehitettävän tuotteen tarkoituksena on aina tuottaa arvoa asiakkaalle. Perinteisen projektikolmion suurin ongelma on se, ettei se tunnista tuotteen laatua tai sen tuottamaa arvoa projektin tavoitteena. Kun ohjelmistokehitystä on suoritettu perinteisesti suunnitelmavetoisesti ja projektikolmion ulottuvuuksista on lukittu ominaisuudet, on arvioitavaksi jääneet resurssit ja aikataulu. Tällöin tuotteeseen tehdään helposti turhia ominaisuuksia, jotka eivät tuota asiakkaalle arvoa, sillä usein projektin puolimatkassa jokin projektin alussa tehty olettamus osoittautuu vääräksi loppukäyttäjien käyttäytymisen perusteella. (Järvenpää & Kovanen 2018)

Ketterän kehityksen lähestymistavat miellettäessä projektikolmio kääntyy ikään kuin päälaelleen ja projektista lukitaankin resurssit ja aikataulu, jolloin suunniteltavaksi jäävät tuotteen ominaisuudet. Jim Highsmith esittelee kirjassaan Agile Project Management: Creating Innovative Products (2010) käsitteenä kuvassa 1 esiintyvän kaltaisen ketterän ohjelmistokehityksen projektikolmion (agilekolmio), jossa ulottuvuuksina ovat arvo, laatu ja rajoitteet. Rajoitteisiin sisältyvät mm. projektin aikataulu, budjetti ja tuotteen ominaisuudet. Koska ketterää ohjelmistokehitystä vedetään arvovetoisesti (eikä suunnitelmavetoisesti), myös ketterässä kolmiossa ajatuksena on erityisesti arvon ja laadun huomioiminen projektin tavoitteina. Projektia ei siis arvioidakaan enää ominaisuuksien lukumäärän perusteella, vaan laatua ja sen tuottamaa arvoa verrataan projektin rajoitteisiin. (Järvenpää & Kovanen 2018)

Ketterä lähestymistapa antaa teoriassa mahdollisuuden muuttaa projektin suuntaa annetun palautteen pohjalta. Käytännössä suunnan muuttaminen voi kuitenkin olla vaikeaa tai jopa mahdotonta esim. tilanteessa kun projekti on hinnoiteltu etukäteen. Ketterä

lähestymistapa on kuitenkin tehokas suoja riskejä vastaan sen ohjatessa projektia tavoite- ja vaikutuslähtöisesti. (Järvenpää & Kovanen 2018)



Kuva 1. Jim Highsmithin esittelemän kaltaiset ketterän projektikolmion vaiheet. (Highsmith 2010)

3.3 Tuotteen kehitysjono

Tuotteen kehitysjono (engl. product backlog) on priorisoitu lista käyttötapauksia ja toiminnallisuuksia, joita tuotteeseen on suunniteltu toteutettavan. Kehitysjonon sisältö muovautuu projektin kuluessa, mikä tekee ajantasaisesta kehitysjonosta projektinhallinnan perusedellytyksen. Kehitysjonon tulisi olla ainoa lähde tuotteeseen toteutettaville vaatimuksille ja muutoksille, jotta vaatimukset löytyisivät yhdestä paikasta ja priorisointi olisi mahdollista. Suoritettaessa ketterää kehitystä, tuotteen kehitysjonosta valitaan sprintin kehitysjonoon ne kohdat, jotka tuotekehitystiimi ennustaa pystyvänsä tekemään inkrementin aikana (kuva 4). (Cobb 2015, 68; Schwaber & Sutherland 2017, 15)

Tuotteen kehitysjonosta vastaa tuoteomistaja. Scrumia suoritettaessa scrummaster saattaa avustaa tuoteomistajaa kehitysjonon hallinnassa ja suunnittelussa. Vastuuseen sisältyvät kehitysjonon sisältö, saatavuus ja järjestäminen. Kehitysjonossa tulisi listata kaikki ne ominaisuudet, toiminnot, vaatimukset, parannukset ja korjaukset, jotka tullaan toteuttamaan tuleviin inkrementteihin. Kehitysjonoon listataan myös kaikki ne asiat mitä tuotteessa saatetaan tarvita, mutta kehitysjonon jalostamisen yhteydessä, tulisi ylimääräinen sisältö jota ei tulla koskaan toteuttamaan, siivota pois kehitysjonosta. Tuotteen kehitysjono ei ole koskaan valmis, vaan se kehittyy tuotteen ja ympäristön kehittymisen kehittyessä. Kehitysjonossa olevien vaatimusten muuttuminen ei pääty koskaan, sillä

muutokset liiketoiminnan vaatimuksissa, markkinoinnissa ja teknologiassa aiheuttavat myös muutoksia tuotteen kehitysjonoon. Tuotteen kehitysjonossa korkeammalla ovat yleensä selkeämpiä ja yksityiskohtaisempia tietoa sisältäviä vaatimuksia. Järjestystä ei tulisi tehdä prioriteetin mukaan, vaan arvon mukaan, jota määrittävät myös riskit, työmääräarvio, liiketoiminta-arvo, sekä riippuvuudet. Kehitysjonon hallinta tulee olla jatkuvaa ja usein toistuvaa. (Cobb 2015, 68; Schwaber & Sutherland 2017, 15)

Käyttäjätarinat (user story) ovat vakiintunut tapa etenkin ketterässä kehityksessä kuvata kehitysjonossa oleva ominaisuus tai toiminnallisuus lyhyesti ja selkeästi. Käyttäjätarinan lyhyessä selosteessa käy ilmi, kuka on ominaisuuden loppukäyttäjä, mitä ominaisuus sisältää ja miksi. Käyttäjätarinoiden ei ole tarkoitus olla täydellinen määrittely, vaan pikemminkin muistutus määrittelystä. Tarina kirjoitetaan aina muotoon ”Käyttäjänä X haluan Y jotta Z”. Käyttäjätarina voisi siis olla esim. ”Lääkärinä voin nähdä kuvaajan potilaan hoidon kehittymisestä, jotta pystyn määräämään potilaalle oikeanlaisen jatkohoidon”. Käyttäjätarina ei saa olla liian tekninen, ei liian ylimalkainen, mutta ei liian yksityiskohtainenkaan. Käyttäjätarinoita käytetään usein sellaisenaan kuvaamaan myös testitapauksia. Hyvän käyttäjätarinan piirteitä ovat:

- Riippumattomuus, jotta se on helposti priorisoitavissa ja liikuteltavissa.
- Neuvoteltavuus, jotta siinä on joustamisen varaa.
- Arvokkuus, jotta se tuo lisäarvoa käyttäjälle, eikä kehittäjälle.
- Arvioitavuus, jotta suunnittelu olisi mahdollista.
- Pienuus, jotta tarina ei ole liian monimutkainen ymmärrettäväksi ja tehtäväksi.
- Testattavuus, jotta laatu saadaan pidettyä yllä ja varmistutaan toteutuksen toimivuudesta ja sen määrittelyn alaisuudesta. (Davis 2012, 16, 62; Vanderjack 2015, 16, 22, 26-27)

Kehitystiimi vastaa kaikista työmääräarvioista. Tuoteomistajan tehtävänä on auttaa tuotekehitystiimiä ymmärtämään vaatimuksia ja tekemään suunnitelmaan kompromisseja tuotekehitystiimin ehdotuksien pohjalta, mutta työmääräarvion tulisi antaa ne henkilöt, jotka työn tekevät. (Cobb 2015, 68; Schwaber & Sutherland 2017, 15)

3.4 Laadun varmistus

Laatu on käsite, jonka arviointikriteerit riippuvat tilanteesta ja asetetuista tavoitteista. Tuotteen laatu arvioidaan sen perusteella, miten hyvin tulos vastaa oikeaa laatua. Kai Ruuska toteaa kirjassaan Projekti hallintaan (1999), että ”Laatu on todettua yhdenmukaisuutta vaatimusten kanssa”.

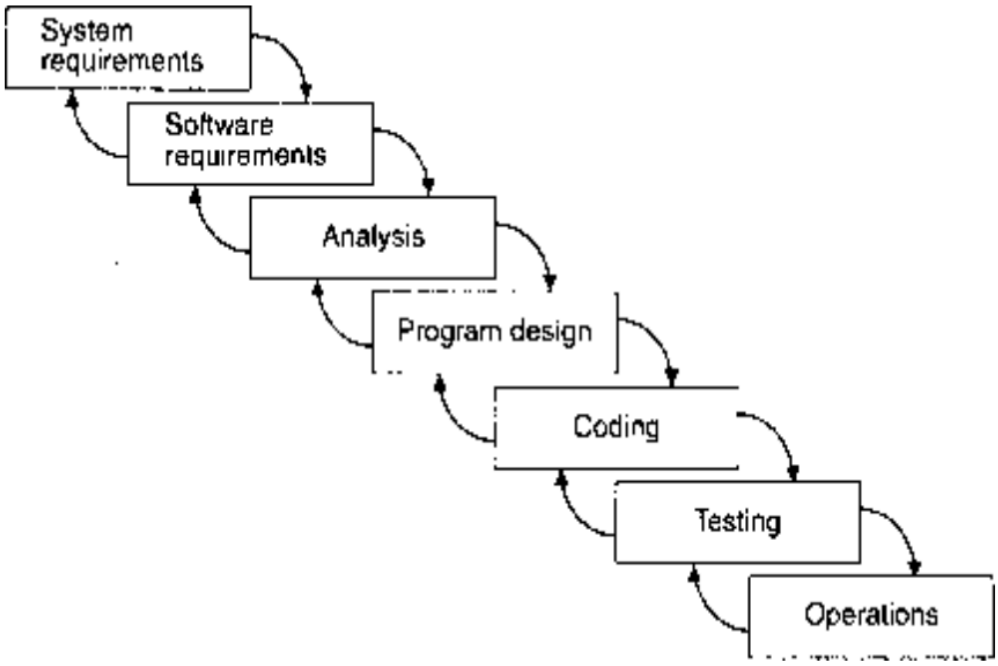
Tuotteen valmistumisen ajankohta ei ole varmaa tietoa ennen laadun varmistusta. Perinteisessä ohjelmistokehityksessä laadunvarmistus on jätetty yleisesti pitkälti projektin loppuvaiheille, mikä aiheuttaa epävarmuutta ja vaikeuttaa siten myös tuotteen myyntiä ja markkinointia. Suurimmaksi ongelmaksi perinteisessä tavassa kuitenkin muodostuu palautesyklin pituus, sillä projektin loppuvaiheessa huomattujen virheiden syitä on vaikeampi paikallistaa ja korjata. Virheitä korjatessa tullaan samalla tehneeksi myös uusia virheitä, jotka myös vaativat oman laadun varmistuksensa. Nykyaikaiset kehitysmenetelmät kuitenkin mahdollistavat laadunvarmistuksen jatkuvasti projektin edistymisen ohella, ilman että kehityksen hidastumista. Aputyökaluina voidaan käyttää esimerkiksi automatisoituja yksikkö- ja käyttöliittymätestejä, jatkuvaa integrointia ja koodin analysointia. Edellä mainittuja työkaluja hyödyntäen virheiden havaitseminen tapahtuu viikkojen tai kuukausien sijasta jopa minuuteissa, mikä tekee virheiden korjaamisesta huomattavasti vaivattomampaa, eikä kehityssykli pääse kestämään odottamattoman kauaa. Kehitystä voidaan johtaa tehokkaammin myös liiketoiminnan näkökulmasta, sillä tuote voidaan julkaista haluttuna ajankohtana, kun laatu on varmistettu jatkuvasti osana tuotteen kehitystä. (Järvenpää & Kovanen 2018)

Testien ja julkaisun automatisointi vie kuitenkin myös paljon projektin työaikaa, joten tuotteen ominaisuuksien määrästä voidaan joutua hieman tinkimään. Automatisointiin käytetty vaiva maksaa itsensä nopeasti takaisin parantaessaan tuotteen laatua, mahdollistaen paremmat jatkokehitysmahdollisuudet, sekä tehden ongelmatilanteiden tutkimisesta ja korjaamisesta suoraviivaisempaa ja helpompaa. Ohjelmoitua toteutusta ei myöskään tulisi koskaan sanoa valmiiksi, mikäli sitä ei ole testattu. Automatisoitu testi on nopeata ajaa ja se ns. siirtää testausosaamisen kaikille. (Forsgren ym. 2018; Järvenpää & Kovanen 2018;)

3.5 Perinteinen vesiputousmalli

Ohjelmistotuotannossa yksi tunnetuimmista perinteisen projektinhallintaprosessin mukaisista menetelmistä on vesiputousmalli, jonka katsotaan usein saanen alkunsa Winston Roycen vuonna 1970 julkaisemastaan artikkelista. Vesiputousmallissa suunnitteluja toteutusprosessi etenee vaihe vaiheelta ns. alaspäin kuten vesiputouksessa (kuva 2). Vesiputousmallia seurattaessa edetään järjestyksessä yhdestä vaiheesta toiseen. Kun vaatimusten määrittely on valmistunut, siirrytään suunnitteluvaiheeseen, eikä vaatimuksia saisi enää muokata. Vesiputousprosessin vaiheet ohjelmistokehityksessä kuvataan usein viitenä eri vaiheena:

1. määrittely
2. suunnittelu
3. ohjelmointi
4. testaus
5. käyttöönotto. (Cobb 2015, 4)



Kuva 2. Vesiputousmalli esitettynä Winston Roycen 1970 kuvaaman kaltaisena. (Cobb 2015, 18)

Ongelmana perinteisessä menetelmässä on sen jäykkyys uusille muutoksille. Tuote määritellään ennen kuin sen tuotanto aloitetaan, mutta määrityksiä päästään muuttamaan vasta kun tuote on julkaistu ja projekti viety päätökseensä. Huolellinen suunnittelu ohjelmiston elinkaaren alussa voidaan laskea vesiputousmallin eduiksi, mutta lähes aina ongelma ja vaatimukset tarkentuvat projektin edetessä ja tiedon määrän kasvaessa. Asiakas ja tuotantotiimi ymmärtävät tehdyt virheet usein liian myöhään, jolloin syntyy paljon tyytymättömyyttä. Vahvuudeksi vesiputousmallissa voidaan todeta mm. se, että asiakkaan ei tarvitse osallistua kehitykseen määrittelyprosessin jälkeen. Vesiputousmalli tekee myös testauksesta suoraviivaisempaa, sillä testisuunnitelma voidaan tehdä jo hyvissä ajoin kehityksen aikana määrittelyä vaste. (Brooks 1995, 265; Davis 2012, Braude & Bernstein 2016, 37)

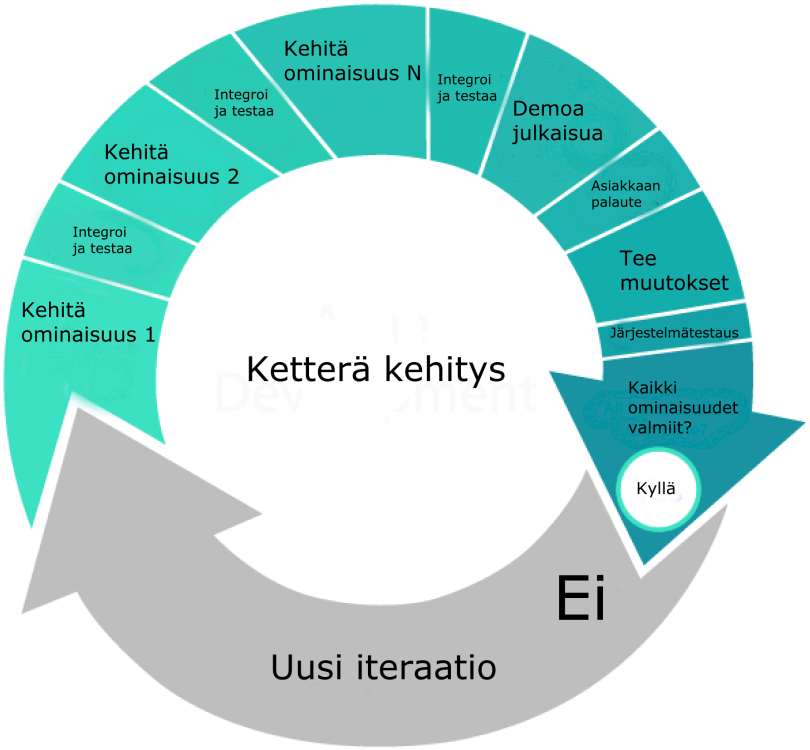
3.6 Ketterät kehitysmenetelmät

Ketterä ohjelmistokehitys on joukko ohjelmistotuotantoprojekteissa käytettäviä menetelmiä. 17 ketterän kehityksen pioneeria kokoontuivat vuonna 2001 keskustelemaan yhteisen kevyen menetelmän kehittämisestä, tarkoituksenaan luoda yhteinen pohja ketterille menetelmille, sekä edistää ketterän ajattelun leviämistä. Tuloksena kokoontumiselle oli ketterän kehityksen perusmääritelmänä pidetty julkaisu nimeltä Agile Manifesto (Agile Alliance 2001). Manifestissa määritellään 4 arvoa, sekä 12 periaatetta, joita ketterän kehityksen menetelmät noudattavat. Manifestin määrittelemät arvot ovat:

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

Manifestin määrittelemät 12 periaatetta julistavat ketteriä ohjelmistokehitysmenetelmiä käyttävän organisaation vannoutuvan asiakkaiden tarpeiden täyttämiseen, muuttuvien vaatimusten vastaanottamiseen, jatkuvaan säännölliseen toimitukseen, yhteistyöhön liiketoiminnan edustajien ja ohjelmistokehittäjien kanssa, motivoitumiseen, tehokkaaseen kommunikointiin kasvokkain, toimivaan ohjelmistoon ensisijaisena mittarina, kestävään toimintatapaan, yksinkertaisuuteen ja tehokkuuden maksimointiin, itseorganisoituvaan tiimiin, sekä säännölliseen jatkuvaan tarkasteluun. (Agile Alliance 2001)

Ketterää ohjelmistokehitystä sanotaan usein myös agileksi kehitykseksi, mikä johtuu sen englannin kielisestä termistä ”agile software development”. Kehitettäessä ohjelmistoa ketterin menetelmin saadaan tuotteesta nopeasti aikaiseksi priorisoitu ja riisuttu perusversio, jota voidaan muokata osa-alueilta, joilta se nähdään kannattavaksi (kuva 3). Tuotteen kehitystila pysyy hyvin hallussa ja työ on helppoa siirtää tehtäväksi tai jatkokehitettäväksi toiselle tiimille. Barbee Davis kirjoittaa kirjassaan Agile Practises for Waterfall Projects (2012, 13-14) ketterän kehityksen 3 tärkeimmäksi avaintekijäksi iteraatioissa kehittämisen, lisäävästi kehittämisen, sekä omistautuneen kehitystiimin. Ketterä ohjelmistokehitys on siis kehitystyötä jossa kehitystiimi työstää projektin tuotetta toistuvasti ja lisäävästi, kunnes tuotteen kaikki ominaisuudet ovat valmiit, eikä lisätyö enää tuota tarpeeksi arvoa asiakkaalle. Ohjelmistoa ei tehdä palasina, vaan jatkuvasti parantaen ja jatkuvasti testaten. Ketteryys ei myöskään tarkoita suoraan nopeaa, vaikka nopeus usein moninkertaistuukin ketterässä kehityksessä (Chin 2003, 9). Kolmantena Davisin mainitseman avaintekijänä oli omistautunut tiimi, mikä tarkoittaa että tiimi keskittyy projektiin täysipäiväisesti, sen sijaan että he yrittäisivät tehdä montaa projektia saman aikaisesti.



Kuva 3. Ketterässä kehityksessä työskennellään toistavasti ja lisäävästi (iteraatioissa), kunnes tuotekehitysprojekti valmistuu. (Davis 2012, 13)

Ketterät menetelmät eivät oikein käytettynä hylkää suunnitelmallisuutta, mutta painopiste siirtyy dokumentaatiosta prosesseihin. Päämääränä on, että kaikki projektiin liittyvät henkilöt ja sidosryhmät ovat yhteisymmärryksessä siitä, mihin kehityksessä pyritään. Ketterässä kehityksessä tärkeää on suunnittelu, eikä niinkään suunnitelma. Suunnitelma on tilapäinen työväline, joka on hyödyllinen suunnittelun aikana. Ketterässä kehityksessä korkea laatu tarkoittaa, että prosessi on järkevä ja että sitä käytetään. Ketterässä kehityksessä tarkkojen suunnitelmien sijalle tulee enemmän prosessien kuvauksia, asioiden priorisointia, työn etenemisen seurantaa, sekä yhteistyötä. (Toikkanen ym. 2013)

3.6.1 Ketterän kehityksen vahvuudet

Projektien tarpeet muuttuvat lähes aina kesken projektin. Kun perinteinen vesiputousmalli on hyvin haavoittuvainen asioille, joita ei pystytä projektin alussa tunnistamaan, ketterässä kehityksessä tarpeita arvioidaan jatkuvasti kokemuksien ja tiedon karttuessa. Kentältä ja loppuasiakkailta saatu palaute on äärimmäisen tärkeää. Ketterä kehitys ei tuota arvoa pelkästään siksi että kehitystyö on nopeaa, vaan yksi tärkeimmistä arvoa tuovista asioista on, että kehitystyötä tehdään jatkuvasti yhdessä asiakkaan kanssa. Tällöin saadaan jatkuvasti palautetta sekä päästään tarkentamaan projektin päämäärää ja ongelmakuvausta, sekä pohtimaan sitä, millaiset ominaisuudet ja muutokset tuottavat eniten arvoa asiakkaalle. (Davis 2012, 76; Cobb 2015, 12)

Järjestämällä työt pieniin vain muutaman viikon mittaisiin kehitysjaksoihin, kokonaisuuden hallinnasta tulee huomattavasti helpompaa, kuin pitkissä kehityssykleissä, joita perinteisessä mallissa käytetään. Ihmisillä on myös usein tapana tehdä asiat viime tingassa ja vasta lähellä aikarajan päättymistä. Ketterässä kehityksessä hyödynnetään tätä ilmiötä pienentämällä sykliä ja tuomalla projektiin useampia aikarajoja ja päätöspisteitä. Päivittäisillä palavereilla, sekä lyhyin väliajoin pidettävien esittely- ja arviointipalaverien myötä tiimi saa jaettua tietoaan tehokkaasti niin tiimin sisällä kuin sen ulkopuolellekin. Julkaisukelpoinen tuote tarkoittaa myös dokumentoitua tuotetta, jolloin lyhyet sykäykset pakottavat tiimiä kirjoittamaan ja jakamaan kehitystietoa tuoreeltaan. (Davis 2012 16; Cobb 2015, 12-13)

Käytettäessä ketteriä menetelmiä, koko kehitystiimi on aina itse mukana suunnittelussa ja työmäärän arvioinnissa koko projektin ajan. Tämä helpottaa lähtökohtaisesti erittäin vaikeaa arviointia liittyen projektin aikatauluihin. Työskentelemällä lyhyissä kehitysjak-

soissa, tiimin kehitysnopeus tulee nopeasti esiin kaikille tiimin jäsenille jo muutaman jakson jälkeen. Tiimin onnistumista työmäärän arvioinnissa arvioidaan jokaisen kehitysjakson jälkeen, joten aikataulujen arviointi helpottuu jokaisen kehitysjakson päätteeksi. (Cobb 2015, 12-13; 41)

Ketterissä projekteissa tulisi aina olla tuoteomistaja, jolla on vastuu tuotteen kehitysjonon sisällöstä ja kehitystiimin tuottaman työn arvon maksimoinnista. Hän toimii toimialan asiantuntijana ja välikätenä loppukäyttäjien, ohjausryhmän ja muiden projektin sidosryhmien välillä. Näin voidaan taata, että kehitystiimillä on aina yksi henkilö johon olla yhteydessä projektiin liittyvissä ongelmissa ja että projektin vaatimukset pysyvät selkeinä ja yhdenmukaisina. (Cobb 2015, 41)

3.6.2 Ketterän kehityksen heikkoudet

Ketterässä kehityksessä on selkeät rakenteelliset säännöt ja viitekehykset, mutta todellisen ketterä kehityksen mallia kuitenkin toteutetaan harvoin, mikä on yksi menetelmän suurimpia ongelmia. Ketterän kehityksen säännöt eroavat hyvin paljon perinteisestä mallista, mikä tekee mallista erittäin mukautuvan, mutta se ei silti tarkoita että siltä puuttuisi prosessit ja organisoituneisuus. Kaikkien organisaatiossa tulee sisäistää ketterän kehityksen kulttuuri ja noudattaa sen asettamia sääntöjä tai siitä tulee ketterän sijasta sekavaa ja muodostuu suuri riski epäonnistumiselle (luku 3; VersionOne 2015; Cuningham 2015).

Tiedon ja tietämyksen kasvaessa uudelleen tekemisen tarpeet usein kasvavat, sillä huomataan että aiemmin tehty ei palvelekaan asiaansa, vaan ominaisuus tulisi tehdä toisella tavalla. Uudelleen tekeminen on normaalia, mutta se vaatii kehitystiimin työn synkronointia, jottei tiimi törmää lopulta tilanteeseen, jossa iso osa tuotteen koodista ei toimikaan toisen osan kanssa yhteen. Uudelleen tekemisen eli refaktoroinnin tarve kasvaa myös tuotteen teknisen velan kasvaessa. Teknisellä velalla tarkoitetaan sitä, että kehityksestä tulee jatkuvasti kalliimpaa teknisten syiden vuoksi, esim. löytämättä tai korjaamatta jääneiden virheiden tai viivästyneen päivityksen vuoksi. Kun ohjelmistoa kehitetään nopeasti ominaisuus kerrallaan, teknistä velkaa syntyy aina jo lähtökohtaisesti, kun perusteelliselle suunnittelulle ei annetta aikaa. (Auer 2013, 77)

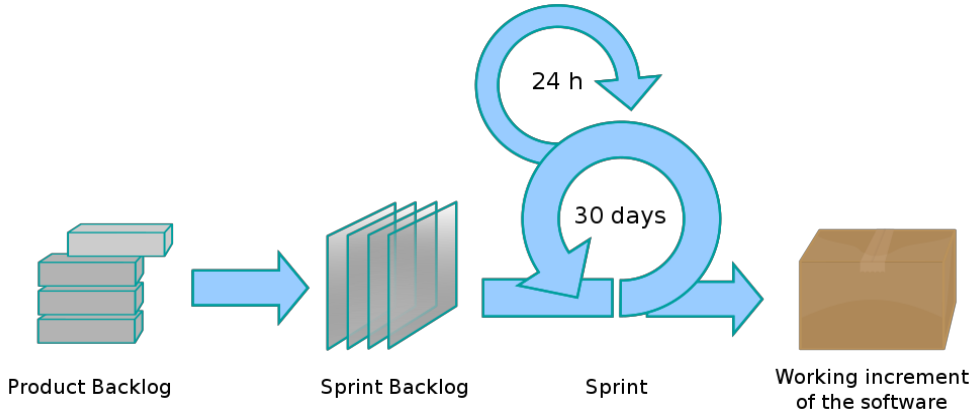
Yksi ketterän kehityksen heikkouksista on asiakasriippuvaisuus. Vaikka asiakastyöskentely ja tiivis yhteistyö ovat ketterän kehityksen suurimpia etuja, mikäli asiakas ei

sitoudu projektiin riittävästi, tulkitaan asiakasriippuvaisuus heikkoudeksi. Asiakkaalta tai tuoteomistajalta on löydyttävä tarvittava aika tiimin työn edistämiseen tai kehitystahti hidastuu ja arvon tuottaminen vähenee, sillä ymmärrys asiakkaista ja heidän tarpeistaan tarvitaan. Yhteistyötä tuoteomistajan kanssa tarvitaan mm. kehitysjonon muodostamiseen, riskien tunnistamiseen ja analysointiin, palautteen antoon tehdystä työstä, sekä . (Auer 2013, 27; Canty 2015)

3.6.3 Scrum

Scrum on projektinhallintamenetelmä ja viitekehys, jota käytetään yleisesti ketterässä ohjelmistokehityksessä. Scrumia voidaan soveltaa ohjelmistoprojektien hallinnan lisäksi myös yleisesti projektinhallinnassa. Scrum projektinhallintamenetelmälle, kuten muillekin ketterän ohjelmistokehityksen menetelmille, on yhteistä toimivan ohjelmiston ensisijaisuus, suora viestintä ja nopea muutoksiin reagointi. Scrum kehitettiin alun perin tuotehallintaan ja tuotteiden kehittämiseen. Sitä on 1990-luvun alkupuolelta lähtien käytetty ympäri maailman laajalti mm. markkinoiden, tuotteiden ja teknologioiden tutkimiseen, tuotteiden kehittämiseen ja jalostamiseen, sekä tuotteiden uudistamiseen ja ylläpitoon. (Vanderjack 2015, 1; Wysocki 2013, 334-335; Schwaber & Sutherland 2017)

Scrumissa, kuten useimmissa ketterissä menetelmissä, työskennellään iteratiivisesti ja inkrementaalisesti (toistavasti ja lisäävästi) ennustettavuuden optimoimiseksi ja riskien kontrolloimiseksi (kuva 4). Tavoitteena oleva tuote kehittyy pikkuhiljaa täydellisemmäksi ja valmiimmaksi useiden kehitysjaksojen aikana. Kehitysjakson kesto on ketterissä menetelmissä tyypillisesti n. 2-4 viikkoa tai yhden kuukauden. Kukin kehitysjakso on kuin pieni projekti ja sisältää kaikki uusien toimintojen julkaisemiseen tarvittavat tehtävät: projektisuunnittelun, vaatimusanalyysin, suunnittelun, toteutuksen, testauksen ja dokumentoinnin. Kehitysjakson lopuksi projektin prioriteetit arvioidaan uudelleen ja päätetään seuraavan jakson sisällöstä. Scrumissa kehitysjaksoa kutsutaan tuotekehityssprintiksi (tai sprintiksi). Valmiin määritelmä on käsite, jonka kaikkien tulee ymmärtää samanlaiseksi. Toisin sanoen, kun jokin osa tuotteesta on valmis, kaikkien on ymmärrettävä mitä se tarkoittaa. Valmiin määritelmä on siis suunniteltava yhdessä, jotta läpinäkyvyys olisi turvattu. Valmiin määritelmä auttaa kehitystiimiä hahmottamaan kehityskohtien haastavuutta ja työmäärää. (Davis 2012, Vanderjack 2015; Schwaber & Sutherland 2017)



Kuva 4. Scrumissa työskennellään iteratiivisesti ja inkrementaalisesti esim. 30 päivän mittaisissa iteraatioissa, sprinteissä. (Wikimedia Commons 2009)

Scrum-kehityksessä käytetään yhtä tai useampaa tiimiä, jotka koostuvat tuotteen omistajasta, scrummasterista ja kehitystiimistä. Kehitystiimi päättää kunkin sprintin tavoitteet ja tehtävät sekä vastaa yhdessä siitä että asetettuihin tavoitteisiin päästään. Kehitystiimi on itseohjautuva ja monitaitoinen, ja sillä on valta päättää omista työmenetelmistään. Se koostuu ammattilaisista, jotka muuttavat tuotteen kehitysjonon tuotteeksi. Tuotteen omistajan tehtävä on määritellä tuotteen vaatimukset sekä priorisoida tuotteen kehitysjono scrumtiimin kanssa. Tuoteomistaja vastaa tuotteen arvon ja kehitystiimin työn maksimoimisesta. Scrummasterin rooli on tapaamisten järjestäminen, kehitystiimin esteiden poistaminen ja muiden tiimien kanssa koordinoiminen. Scrummaster vastaa siitä, että kaikki ymmärtävät ja käyttävät Scrumia. Hänen tehtävänsä on myös suojella kehitystiimiä ulkopuoliselta hälyltä ja taata sille työrauha. Hän tarkkailee työn etenemistä ja sprintin tavoitteiden saavuttamista, sekä kommunikoi tuotteen omistajan kanssa mahdollisista ongelmista ja viivästymisistä. (Vanderjack 2015, 1-2; Davis 2012, 143; Schwaber & Sutherland 2017)

Scrumissa käytetään ennalta sovittuja tapahtumia luomaan säännöllisyyttä ja pitämään palaverit tehokkaina ja niiden tarve mahdollisimman pienenä. Scrumin tapahtumat ovat aikarajattuja jotta varmistuttaisiin siitä, että suunnittelulle ja toteutukselle varataan riittävästi aikaa. Scrumissa on 4 muodollista palaverityyppiä tarkasteluun ja sopeuttamiseen: sprintin suunnittelupalaveri, päiväpalaveri, sprinttikatselmointi, sekä sprintin retrospektiivi.

Suunnittelupalaveri (ns. aloituspalaveri) on seuraavan tuotekehitysiteraation suunnittelua. Tapahtuma suoritetaan jokaisen tuotekehityssprintin alkajaisiksi. Aloituspalaverin

päämääränä on lukita seuraavassa sprintissä tehtävät asiat, sekä varmistaa että kaikki osapuolet ymmärtävät mistä valituissa tehtävissä on kyse. Tuotekehitystiimi sitoutuu ominaisuuksien toteutukseen ja tuotepäällikkö sitoutuu opastamaan näiden teossa. (Layton & Ostermiller 2017, 76; Vanderjack 2015, 18; Schwaber & Sutherland 2017)

Sprinttikatselmointi (ns. päätöspalaveri) on tapahtuma joka suoritetaan jokaisen tuotekehityssprintin päätteeksi. Tässä tapahtumassa kehitystiimi esittelee projektin sidosryhmille tai vaikka koko organisaatiolle sprintin aikaiset julkaistut tuotoksensa. Tapahtuman päämääränä on saada sidosryhmiltä välitöntä palautetta tehdystä ja sen onnistumisesta. Sidosryhmän tehtävä on informoida tuotepäällikköä toivomistaan muutoksista, jotta tämä voi päivittää tuotteen kehitysjonoa ja asettaa seuraavan sprintin tavoitteet. Tarkoituksena ei ole esitellä ainoastaan uusia ominaisuuksia joita tiimi on pyöritellyt sprintin aikana, vaan kokonaisuutta, eli osoittaa myös miten uudet ominaisuudet toimivat vanhan toiminnallisuuden kanssa yhteen. (Layton & Ostermiller 2017, 76; Vanderjack 2015, 19; Schwaber & Sutherland 2017)

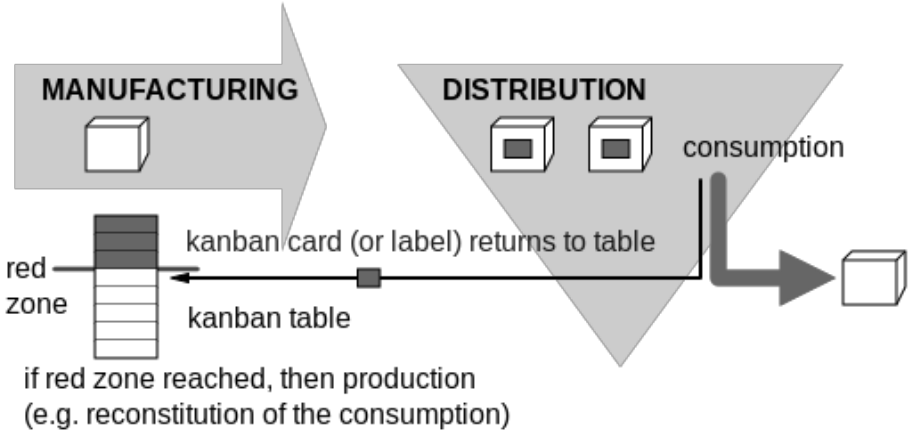
Sprintin retrospektiivissä koko projektitiimi keskustelee yhdessä, mikä sprintissä meni hyvin ja missä olisi vielä parantamisen varaa. Tapahtuman ajankohta on jokaisen sprintin lopussa. Koska ketterässä kehityksessä kannustetaan ns. nopeaan virheen kuittaamiseen, ihmisten ei tulisi olla ujoja tuomaan esille epäonnistumisia. Ennemminkin tulisi huomata, että virheet ovat seurausta pieleen menneestä prosessista. Kun hyvin ja huonosti menneet asiat on kerätty, tiimi päättää yhdessä, mitkä niistä ovat mahdollisia korjata, jonka jälkeen korjaustoimenpiteet tulisi lisätä myös kehitysjonon tehtävälistalle. Virheistä tulee oppia, jottei niitä toisteta. (Layton & Ostermiller 2017, 76; Vanderjack 2015, 20-21; Schwaber & Sutherland 2017)

Päiväpalaveri on sprintin päivittäinen tapahtuma, jolle tiimi sopii jokapäiväisen ajankohdan. Palaverin ei ole tarkoitus kestää 15 minuuttia kauempaa. Päiväpalaverin tarkoituksena on synkronoida tiimin tekemistä. Jokainen kehittäjä kertoo vuorollaan, mitä on edellisen päivän aikana tehnyt, mitä tulee seuraavan päiväjakson aikana tekemään, sekä onko hänen työllään joitakin esteitä. Palaverissa ei ole tarkoitus alkaa ratkoa esiin tulleita ongelmia, vaan informoida tiimin kollegoita projektin tilanteesta. Päiväpalaverissa myös tarkistetaan että kehitysjonon tehtävien tila on oikea. (Layton & Ostermiller 2017, 76; Vanderjack 2015, 21; Schwaber & Sutherland 2017)

3.6.4 Kanban

Kanban on Lean-periaatteen mukainen tuotannon ajoitusjärjestelmä, joka auttaa määrittämään, mitä pitää tuottaa, milloin ja millaisissa määrissä. Se on myös työväline, jolla visualisoidaan ja hallitaan työjonoja. Kanbanissa on kyse jatkuvasta kehityksestä ja toimituksesta. Yksi Kanbanin tärkeimmistä säännöistä on asettaa jokaiseen työnkulun vaiheeseen rajoitus samanaikaisen työn määrästä. Kanban havainnollistaa mitä tehtäviä on työn alla, mikä estää tehtävien etenemistä ja paljonko aloittamattomia tehtäviä on jonossa. Kanban on ohjelmistokehityksessä yleinen vaihtoehto Scrumille ja soveltuu hyvin erityisesti ylläpito- ja tukityöhön, jossa ennalta suunnitteleminen on hankalaa, eikä kehitysjaksojen tehtäviä voida lyödä lukkoon. Kanban perustuu vahvasti keskeneräisen tuotannon rajoittamiseen. (Cobb 2015)

Kanban-prosessin ymmärtämiseksi tulee ymmärtää ero työntö- ja vetotuotannossa (pull- ja push-prosessit). Työntötuotannossa pystytään etukäteen kertomaan valmistumisajat, mutta muutostilanteissa koko tuotanto joudutaan suunnittelemaan uudelleen. Vetotuotanto seuraa työntötuotantoa paremmin toimittajan ja asiakkaan tarpeita. Vetotuotanto perustuu aina tarpeeseen ja mukautuu nopeammin ja helpommin erilaisiin tarpeisiin. Vetotuotanto perustuu seuraavalta työvaiheelta saatuun lupaan. Kanbanissa tuotanto pohjautuu aina tarpeeseen. Kuvassa 5 on esitettynä Kanban-prosessi. Kun ohjelmointitehtävä saadaan valmiiksi, tehtäväjonosta otetaan aina seuraava priorisoitu tehtävä työn alle. Mikäli tehtävässä ei ole tarpeeksi tietoa sen suorittamiseen tai sen suorittaminen on jostain muusta syystä estynyt, valitaan uusi tehtävä. Saman aikaisten tehtävien lukumäärä on rajoitettu, jotta tekemistä voitaisiin paremmin seurata. (Cobb 2015)



Kuva 5. Kanban-prosessi. (Wikimedia Commons 2013)

Kanbanissa ja scrumissa molemmissa on periaatteena käsitellä isot ja monimutkaiset tehtävät pienempinä palasina. Molemmat pyrkivät jatkuvaan kehittämiseen ja työn optimointiin. Molemmat myös tarjoavat helpon näkyvyyden työn alla oleviin töihin. Taulukossa 1 on Charless Cobbin teoksessa “The project Manager’s Guide to Mastering Agile” (2015) kuvaamat eroja Scrumin ja Kanbanin välillä.

Taulukko 1. Kanbanin ja Scrumin erot.

Tyypillinen Kanban-prosessi	Tyypillinen Scrum-prosessi
Perustuu jatkuvan tuotannnon malliin, eikä työtä jaeta aikarajattuihin kehitysjaksoihin.	Työtä tehdään aikarajatuissa ja toistuvissa kehitysjaksoissa eli sprinteissä.
Tehtäviä voidaan lisätä tuotantoon työn alla olevaksi (WIP, Work In Process) aina kun tuotannossa on tilaa.	Tehtäviä ei voida lisätä tehtävälistalle (WIP) kesken sprintin, vaan sprintissä työstetään aina ennalta sovittua tehtävälistaa.
Työmäärän arviointi on valinnaista	Työmäärä on arvioitava ennen jokaisen sprintin aloittamista, sillä sprintin (lukittu) sisältö perustuu tiimin oletettuun työvauhtiin.
Työn alla olevien tehtävien lista voidaan jakaa useisiin rajoitettuihin tasoihin.	Kanbanin kaltaista työn alla olevien tehtävien rajausta voidaan käyttää, mutta Scrumissa se ei ole oleellista.
Kanban-taulu voidaan jakaa monen tiimin kesken.	Sprintin tehtävälista on aina vain yhden tiimin omistuksessa.
Ei roolitusta, vaan Kanban-prosessi voidaan jalkauttaa kaikenlaisille eri rakenteen omaaville tiimeille.	Roolitus (scrummaster, tuoteomistaja ja kehitystiimi) on Scrumissa tärkeää

3.6.5 Ketterien kehitysmenetelmien yleiset ongelmat

Ketteryydestä ei ole hyötyä, jos projektia ei ymmärretä lopettaa, kun kehityksestä ei enää synny uutta arvoa. Asiakkaan tulisi pystyä päättämään projekti jokaisen sprintin päätteeksi havahtuessaan ettei arvoa enää synny tarpeeksi. Säännölliset tapaamiset asiakkaan ja toimittajan välillä kertovat arvon tuottamisesta ja projektin päättymisestä usein hyvissä ajoin. (Järvenpää & Kovanen 2018). Mikäli säännöllisiin toistuviin tapaa-

misiin ei pystytä, rikotaan ketterän kehityksen sääntöjä. Monessa organisaatiossa yhdistetään ketterät kehitystavat osaksi perinteistä vesiputousmallia. Kansainvälisen konsulttiyhtiön Actuation Consultingin teettämän tutkimuksen ”Product Development Methods: Is Agile The Most Widely Used?” mukaan jopa 45,41% kaikista tutkimuksessa olleista ohjelmistokehitysmenetelmistä on vuonna 2014 ollut ns. hybridimallia.

Lee Cunninghamin (2015) kirjoittaa Agile Alliancen verkkosivuilla julkaisemassaan artikkelissa ”8 Reasons Why Agile Projects Fail” syistä, joiden vuoksi ketterät kehitysmenetelmät epäonnistuvat. Hän pohjaa artikkelinsa VersionOne organisaation teettämään vuosittaiseen tutkimukseen ”9th annual state of agile report” vuodelta 2015. Hän esittää kulttuurin sisäistämisen epäonnistumisen yhdeksi yleisimmistä syistä ketterän kehittämisen epäonnistumiseen. Tutkimuksen mukaan näin oli vastannut 42% tutkimukseen osallistuneista henkilöistä, jotka kokivat ketterän kehittämisen kulttuurin epäonnistuneen organisaatiossaan. Tutkimus osoitti että 44% epäonnistuneissa tapauksissa organisaatiolla on ollut puutteita ketterän kehityksen taidoissa ja kokemuksissa. VersionOnen sama tutkimus vuodelta 2018 ”12th annual state of agile report” osoittaa että kulttuurin puuttumisen koetaan olevan ongelma yhä useammin, sillä vuonna 2018 53% vastanneista koki kulttuurin olevan ketterän kehityksen vastainen. 41% koki, että organisaatiolla olisi puutteita ketterän kehityksen taidoissa vuonna 2018. Epäpätevyys ja sääntöjen noudattamattomuus ovat siis varsin yleiset ongelmat.

Taulukossa 2 on esitetty 12 yleistä syytä ketterän kehityksen epäonnistumiseen. Taulukon asiat on koostettu amerikkalaisen ohjelmistoyhtiö R-style Labin digitaalisen markkinoijan Maria Shestakovan (2014) ja Lee Cunninghamin (2015) artikkelien pohjalta. Myös VersionOnen vuosittaiset tutkielmat (2015; 2018) vahvistavat taulukon sisällön.

Taulukko 2. 12 kohtalokasta virhettä, jotka voivat tuhota ketterien tapojen kehitysprosessin (Shestakova 2014; Cunningham 2015; VersionOne 2018)

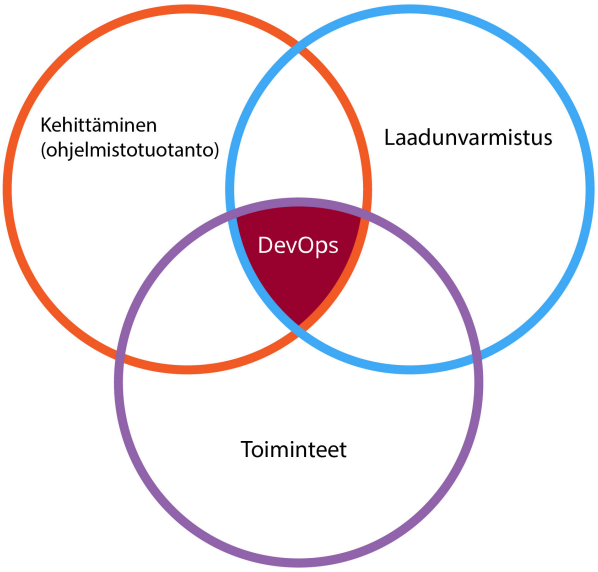
	Ongelma	Kuvaus
1.	Omakutoinen tuotekehitysprosessi, jossa ketterää kehitystä käytetään vesiputousmallin rinnalla	Agile-lähestymistapa liitetään usein kehitys- ja julkaisuvaiheeseen. Tuotteen kehitykseen osallistuvat muut tahot pysyvät kuitenkin usein perinteisessä mallissa, jonka johdosta arkkitehtien, suunnittelijoiden ja määrittelijöiden on hankalaa reagoida asiakkaan toivomiin yllättäviin muutoksiin.

2.	Ketterien kehitystapojen parhaiden käytäntöjen laiminlyöminen	Ketterän kehityksen prosessilla on välttämättömiä toiminteita: pienet itseohjautuvat ryhmät, jotka työstävät käyttötapausvaatimukset aina valmiiksi saakka, päivittäiset nopeat ytimekkäät kokoukset, selkeät kehitysprioriteetit ja säännölliset iteraatiot. Näiden käytäntöjen laiminlyöminen saa minkä tahansa ketteränprojektin epäonnistumaan.
3.	Kommunikaation ja itseorganisoitumisen puuttuminen	Ketterän kehityksen ryhmänjäsenten tulee kommunikoida keskenään päivittäin. Lyhyet päivittäiset tapaamiset ovat välttämätön osa kehitysprosessia. Jokaisen jäsenen tulee ymmärtää, mitä on tehty eilen ja mitä on seuraavan päivän ohjelmassa. Jokainen voi osallistua näkemyksellään seuraavaksi tehtävän suunnitteluun.
4.	Liiallista ohjaamista	Jatkuva ohjaaminen on pahasta. Johtajan jatkuva projektin kaitseminen tuhoaa ketterän kehityksen. Itseohjautuminen motivoi työntekijää parhaiten.
5.	Väärä valinta avainpersoonaksi (scrummaster)	Avainpersoonan tulee ymmärtää miten ketterässä kehityksessä toimitaan, miten sitä ohjataan ja miten ongelmat ratkaistaan. Hänen tehtävänsä ei ole tehdä päätöksiä koko tiiminsä puolesta. Hän ei saa samanaikaisesti työskennellä tuoteomistajana tai osallistua kehitystiimin toimintaan.
6.	Tuoteomistajan osallistumisen puute	Tuoteomistajan antama palaute on elintärkeää. Tuoteomistaja ohjaa kehitysprojektin suunnan. Hänen tehtävänsä on jakaa näkemyksensä kehitystiimille, kommunikoida kehityksestä jokaisessa tuotekehitysjaksossa ja vahvistaa tehdyn työn laatu. Hänen tehtävänsä on varmistaa että kaikki ovat hänen mielipiteestään tietoisia, eivätkä unohda sitä. Tuoteomistajan tulee olla hyvä johtaja.
7.	Liian isot tuotekokonaisuudet ominaisuuksien työstämiseen	Jokainen projekti tulee paloitella pieniin käsiteltäviin osiin. Mitä pienempiä paloitellut tehtävät ovat, sitä helpommin ne on tehtävissä. Kokonaiset toiminnot eivät usein synny yhden kehitysjakson aikana, vaan tarvitsevat useamman toiston valmistuakseen. Näin tuoteomistajalla on enemmän aikaa reagoida asiakkaan jatkuvasti vaihtuviin mielipiteisiin.

8.	Epäonnistuneet aikatauluarviot ja suunnitelmat	Ensimmäisen sprintin sisältö on uudelle tiimille aina vaikeaa arvioida. Arviointi parantuu toistoin, joten seuraavassa iteraatiossa arviointi on aina helpompaa. Tuotteen tehtävälistan groomausta ei tule unohtaa tai vähätellä. Groomauksessa tehdään selväksi, että kaikki ymmärtävät mitä tehtävän suorittamiseen vaaditaan ja työmääräarvio tehdään yhdessä tiimin kanssa, joten itse tehtävän tekijällä ei ole merkitystä.
9.	Julkaisun viivästymiset	Agile-tiimi on sitoutunut aikatauluihin ja kehittää tuotetta vähän kerrallaan. Yksi tuotekehitysjakso (iteraatio tai sprintti) on yleensä 2-4 viikkoa pitkä. Päivän tai kahden myöhästyminen voi pahimmillaan johtaa tällöin jopa kuukauden myöhästymiseen.
10.	Laatu on julkaisua tärkeämpää	Aikainen julkaisu ja tuotteen näkyvyys asiakkaalle ovat isoja etuja asiakkaalle ja loppukäyttäjälle. Julkaisu ei kuitenkaan saa luoda suurta määrää poikkeumia tuotteen tehtävälistaan asiakkaan tyytymättömyyden vuoksi.
11.	Huono testausprosessi	Tiimin laadunvalvonnasta vastaavien tulee testata jokaista tuotteen ominaisuutta pienissä paloissa vähän kerrallaan jo kehityksen aikana, eikä vasta sitten kun se on valmis kokonaisuudessaan. Näin ollen testauksesta syntyy jatkuva prosessi, joka estää bugien syntymistä, sen sijaan että niitä etsittäisiin vasta jälkikäteen. Hyvä laadunvalvonta on järjestelmävirheiden syntymisen estämistä, ei niiden löytämistä.
12.	Epäselvät määritelmät valmiista	Valmiin määritelmä on se mitä tuote on aivan lopuksi. Kun valmis-tila on saavutettu, projekti päättyy. Määritelmä valmiista tulee aina olla olemassa ja määritelmien tulee olla selkeitä sekä vailla moniselitteisyyttä. Yleensä ihmiset työskentelevät paremmin, kun he tietävät, mitä heidän pitäisi saavuttaa.

3.7 DevOps

DevOps on sähköisten palvelujen tuotantoon tehty toimintamalli, jolla pyritään automatisoimaan ohjelmistokehitykseen, testaamiseen ja ylläpitoon liittyvät IT-palvelutoiminnot, sekä siten lyhentämään ja tehostamaan kehittämiseen käytettävää aikaa. DevOps termi on muodostettu sanoista ”development” eli kehitys sekä ”operations” eli tuotanto. Jotta DevOps toimisi tehokkaasti, on ohjelmistojen oltava käyttöönotettavia, muokattavia, testattavia sekä valvottavia. DevOps yhdistää ketterän kehityksen menetelmät, jatkuvan integraation (continuous integration), sekä jatkuvan toimituksen (continuous delivery) menetelmiä. Kun ketterässä kehityksessä pidetään keskiössä lähinnä kehitysprosessia, DevOpsissa keskiössä on asiakas. DevOps pyrkii mahdollistamaan järjestelmän vaatimusten ja automaattisen laadunvarmistuksen yhdistämisen, jolloin yrityksen koko henkilökunta voi seurata tuotantoprosessin etenemistä reaaliaikaisesti. Jatkuvassa integraatiossa kehitetyt toiminnallisuudet ja korjaukset viedään heti kehityksen yhteydessä tuotantoon osaksi toimivaa ohjelmistoa ja jatkuva toimitus puolestaan sisältää toimenpiteet joiden avulla uusi versio ohjelmistosta viedään tuotantokäyttöön. Teknisten ratkaisujen lisäksi DevOpsissa on kyse myös kulttuurista ja jatkuvasta parantamisesta. Se haastaakin kysymään miksi ja miten asioita tehdään, ei niinkään millä työkaluilla. Kaiken kaikkiaan DevOps on siis ohjelmiston kehittämistä, laadun varmistusta sekä toimenpiteiden automatisointia (kuva 6). (Swartout 2012; Atlassian 2018)



Kuva 6. DevOps koostuu ohjelmistokehityksestä, laadunvarmistuksesta ja toiminteista.

Jatkuvassa parantamisessa hyödynnetään retrospektiivejä eli tapaamisia, joissa tarkastellaan yhteistyön ja ohjelmistokehityksen sujuvuutta. Toimintatapoja voidaan tällöin kehittää tehtyjen havaintojen pohjalta. Ongelmia ja haasteita tulee lähestyä systemaattisesti, jolloin voidaan varmistaa perehtyminen asiaan ja sen mahdollisiin aiheuttajiin. Systemaattisuus auttaa myös suunnittelemaan sekä toteuttamaan niin korjaavat kuin ennaltaehkäisevät toimenpiteet ongelmatilanteeseen. Oleellista on myös seuranta ja tulosten analysointi ennen käyttöönottoa. (Järvenpää & Kovanen 2018)

3.7.1 Jatkuva integraatio

Nykyaikaiset ohjelmistot koostuvat useista komponenteista joita kehitetään toisistaan erillään. Jotta ohjelmien ja komponenttien päivittäminen sekä yhteen liittäminen olisi mahdollisimman helppoa ja suoraviivaista, tulee muutokset julkaista mahdollisimman useasti, jopa monta kertaa päivässä. Näin taataan uusille toiminnoille jatkuvasti toimiva ja käyttöönotettava pääohjelmisto. Myös tuotteen tekninen velka saadaan pidettyä kurissa.

Jatkuvan integraation (Continuous Integration, CI) keskeisimpiä käytäntöjä ovat versionhallinta, päivittäinen muutosten päivittäminen, ohjelmiston koostaminen, testaus, käyttöönotto ja tiedon jakaminen. Se painottaa tehtävien automatisointia, jatkuvaa testausta sekä ohjelmiston tilanteen koosteen luomista jokaisen ohjelmistomuutoksen yhteydessä. Automatisoinnin avulla julkaisuprosessi saadaan sulavaksi ja inhimillisten virheiden esiintyminen saadaan parhaimmillaan jopa kokonaan mitätöityä. Esim. testaus voidaan suorittaa läpinäkyvästi taustalla heti, kun ohjelmistokoodin muutokset julkaistaan versionhallintaan. Näin saadaan myös välitön palaute uuden ohjelmakoodin toimivuudesta ja varmistutaan yhteen koodin sopivuudesta muun ohjelmakoodin kanssa. (Swartout 2012, 77-78; Vadapalli 2018, 25; Atlassian 2018)

3.7.2 Jatkuva toimittaminen

Jatkuva toimittaminen (Continuous Delivery, CD) täydentää jatkuvan integraation toimintoja mm. automatisoiduin julkaisuskriptein, pyrkien tekemään tuotteen toimittamisesta mahdollisimman sulavan prosessin, joka parhaimmillaan ei tarvitse kuin yhden napin painalluksen. Jotta tämä olisi mahdollista, tarvitaan myös automaattisesti suoritettavia testejä (esim. yksikkö- ja integraatiotestit, sekä funktionaaliset testit), jotka

varmistavat ohjelmistokoodin toimivuuden ja yhteensopivuuden. (Swartout 2012; Vadapalli 2018, 26; Atlassian 2018)

Jatkuva toimittaminen on monimutkainen prosessi, jonka mm. kansainvälinen IT-alan konsulttiyhtiö Praqma jakaa kypsyystaulukossaan (taulukko 3) 5 eri tasoon:

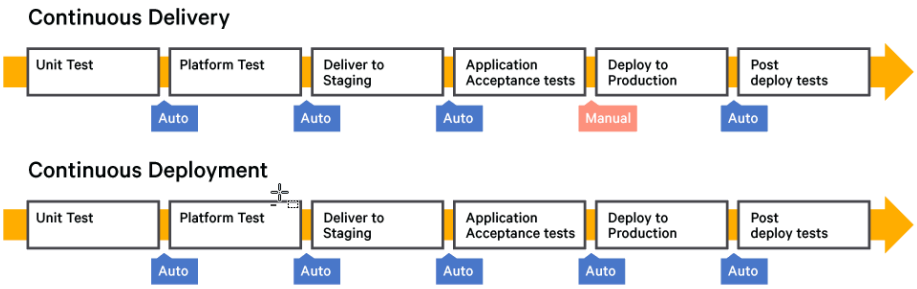
1. 1-tasolla toiminteet ovat suurimmaksi osaksi manuaalisia.
2. Aloittelevaa jatkuvaa käyttöönottoa, jossa on toistettavia ja hallittavia prosesseja.
3. Keskitasolla on johdonmukaisia ja määriteltyjä käyttöönottoja
4. Automatisoitu kokonaisuus
5. DevOps eksperttien optimoima ja mittaroitu jatkuva käyttöönotto

Taulukko 3. Jatkuvan toimittamisen kypsyystaulukko (Pragma 2017).

	Novice	Beginner	Intermediate	Advanced	Expert
Build	Automated builds	Artifacts are managed	Automated release notes	Full traceability	Delivery pipeline
Test	Unit testing, mocks, stubs and proxies	Automated functional tests	Maintain test data	Adaptive test suites	Test in production
Version Control	Commits are tied to tasks	Release train branching strategy	Version numbers matter	Use distributed VCS	Pristine integration branch
DevOps	One Team	Automated deployment	Access to production-like environments	Infrastructure as code	Live monitoring and feedback
Architecture & Design	Code metrics	Testable code	Dependencies are managed	Individually releasable components	Full audit trail in production
Organization & Culture	Agile process	Buy-in from management	Tasks are groomed	Designated roles	Explicit knowledge transfer

Jatkuvaa toimitusta ei pidä sekoittaa jatkuvaan käyttöönottoon (kuva 7), jolla tarkoitetaan hyvin samankaltaista prosessia jossa kuitenkin viimeinen vaihe eli julkaisu, on hyvin paljon erilainen käsin laukaistavan toimituksen kanssa. Toimitettaessa tuote testipalvelimelle tai tuotantopalvelimelle testaukseen, sovelluksen siirto ja testaus on käsin tehtävää työtä, ainakin siinä määrin, että automatisoitu prosessi on laukaistava itse.

Automaattisessa käyttöönotossa tuotteen julkaisun koko prosessi tapahtuu itsekseen esim. yöllisinä ajoina. Tällöin inhimillisten virheiden mahdollisuus saadaan mahdollisimman pieneksi ja julkaisu mahdollisimman sulavaksi, sillä ainoastaan automaattisten testien löytämät virheet voivat keskeyttää automaattisen julkaisuprosessin. (Pragma 2017; Atlassian 2018)



Kuva 7. Jatkuva toimittaminen ja jatkuva käyttöönotto.

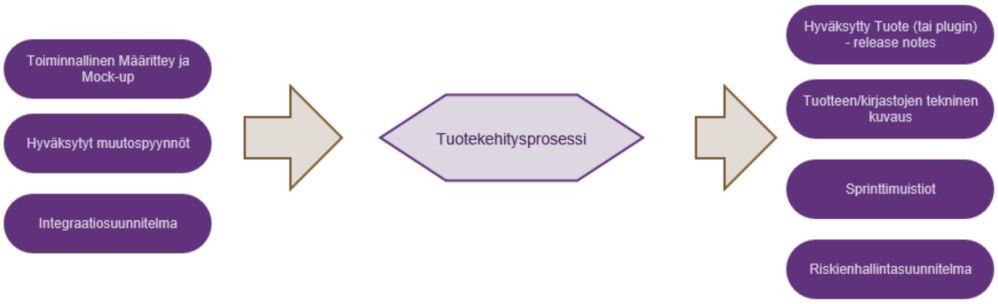
3.7.3 Jatkuva käyttöönotto

Jatkuva käyttöönotto menee vielä askeleen jatkuvaa toimittamista pidemmälle. Jatkuvassa käyttöönotossa jokainen testit läpäissyt ohjelmistokooste viedään tuotanto- tai testiympäristöön täysin automatisoidusti, ilman inhimillistä osallistumista. Jatkuva käyttöönotto on erinomainen työkalu esim. palautteenannon jaksottamiseen ja nopeuttamiseen. Lisäksi se vie ohjelmointitiimiltä vastuuta ja painetta pois, sillä tiimi voi julkaisupäivän sijasta keskittyä pelkkään tuotteen kehittämiseen. (Swartout 2012; Atlassian 2018)

Jatkuva käyttöönotto voidaan sopia tehtäväksi esim. päivittäin, öittäin, viikoittain tai kuukausittain. Jotta jatkuvasta käyttöönotosta ja automatisoinnista saataisiin mahdollisimman paljon hyötyä, julkaisuväli tulisi kuitenkin pitää mahdollisimman lyhyenä. Pienten ohjelmistokokonaisuuksien julkaiseminen myös parantaa esiintyvien ohjelmistovirheiden jäljitettävyyttä, sekä tekee ongelmatilanteesta elpymisen helpommaksi. (Swartout 2012; Vadapalli 2018, 26; Atlassian 2018)

4 KEHITETTÄVÄ TUOTEKEHITYSPROSESSI

Tämän kehittämishankkeen kohteena on toimeksiantajaorganisaation tuotekehitysprosessi. Tuotekehitysprosessi perustuu Scrum -projektinhallinnan viitekehykseen. Prosessissa voidaan valmistaa uusi tuote tai uusia ominaisuuksia olemassa olevaan tuotteeseen. Aloitusehtoina tuotekehitysprosessille ovat yleiset tuotteiden perusvaatimukset (arkkitehtuuri ja toimintokirjastot, käyttäjähallinta, yms), hyväksytty toiminnallinen määrittely, hyväksytty hahmotelma toteutettavasta tuotteesta, sekä tuotekohtainen integraatioiden suodatuskuva. Kuvassa 8 on kuvattuna toimeksiantajan tuotekehitysprosessin vaatimat ja tuottamat asiat.



Kuva 8. Tuotekehitysprosessi. (Toimeksiantaja 2018)

Tässä luvussa käsitellään myös määrittely-, testaus- ja toimitusprosessit. Määrittelyprosessin päättyminen on ehto tuotekehitysprosessin alkamiselle. Tuotekehitysprosessissa käsiteltävä tuotekehitysprojekti pohjautuu siis määrittelyprosessin tuotokseen. Toimeksiantajan prosessikuvausten ja laatukäsikirjan mukaan tuotekehitysprojektin tuotepäällikkönä toimii määrittelyprosessia johtanut määrittelijä, mikä on ongelmallista tilanteissa, joissa tuotepäällikkö joutuu määrittelemään saman aikaisesti tuotekehitykseen tulevia tuotteita, sekä ohjaamaan tuotekehitysprojektia. Aikataulutukseen liittyvää ongelmaa pyritään selventämään kuvaamalla toimeksiantajan tuotestrategiaa, joka ohjaa organisaation eri prosessien työstämien projektien prioriteetteja ja aikatauluja. Testausprosessi ja toimitusprosessi ovat myös oleellisia tuotekehitysprosessia käsiteltäessä, sillä toimeksiantajan prosessikuvausten ja laatukäsikirjan mukaan niillä on ns. takaisinveto-oikeus tuoda tuotekehitysprojekti takaisin tuotekehitysprosessin kehitettäväksi tai korjattavaksi.

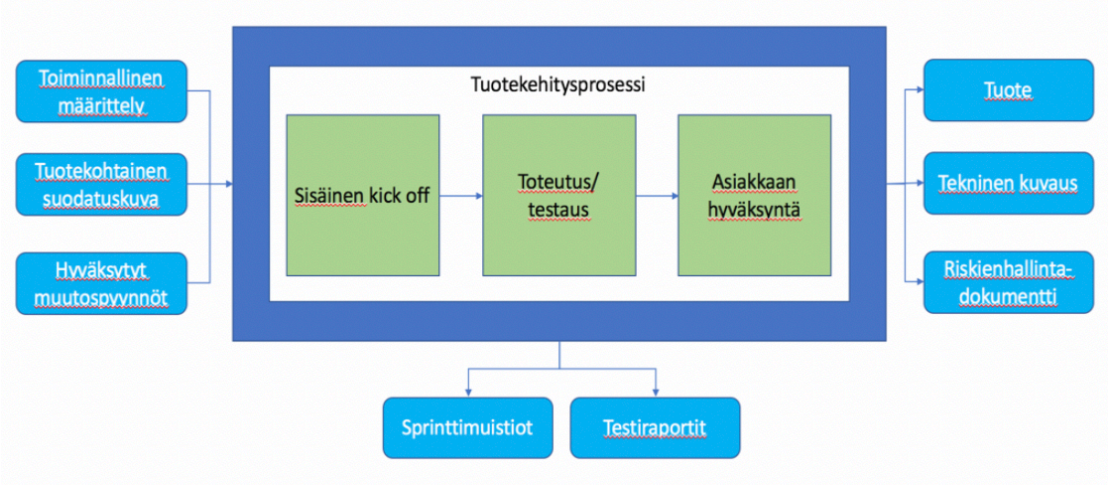
4.1 Tuotestrategia ja aikataulutusongelma

Roadmap on visuaalinen työkalu tuotestrategian kommunikointiin. Sen tehtävänä on kuvata vaiheita, joiden kautta lopulliseen määränpäähän on suunniteltu päästävän. Tuotekehitysprosessiin roadmap vaikuttaa siten, että se määrää yhden tilikauden aikana tuotekehitysprosessissa suoritettavien projektien määrän ja järjestyksen. Siten se vaikuttaa myös toiminnan ketteryyteen, sillä projektien määrä ja järjestys tietyssä aikataulussa lukitsee projektiin käytettävän ajan, jossa tuotteeseen suunnitellut toiminnot ja ominaisuudet on tehtävä.

Roadmapin projektien aikatauluvat perustuvat määrittelyprosessin yhteydessä tehtyihin arvioihin projektin koosta. Ennalta asetetuissa aikatauluissa ja ennalta määritellyissä tuotteen ominaisuuksissa ongelmaksi muodostuu tilanne, jossa arviot työmäärästä eivät kohtaakaan todellista toteumaa. Mikäli tuotteen ominaisuudet muuttuvat ketterästi tehtävän tuotekehitysprojektin aikana, syntyy valtava paine aikataulujen toteutumisesta. Organisaation kaikki eri prosessit pyritään priorisoimaan roadmapin asettaman ohjelman mukaiseksi ja mikäli aikataulut muuttuvat, jää työpöydälle useampi saman aikainen tuote. Tuotteen määrittelyssä pyritään aina ottamaan huomioon tuotteen koko ja toiminnot siten, että tuote olisi roadmapin mukaan tehtävissä.

4.1.1 Tuotekehitysprosessi

Tuotekehitysprosessi alkaa sisäisellä projektin aloituspalaverilla eli ns. tuotekehitys kick-offilla, jota seuraa tuotteen ohjelmointi ja tuotekehityksen aikainen testaus. Kun kaikki tuotteen ominaisuudet on toteutettu tai ne ovat lähellä valmistumista, tuotteelle haetaan asiakas hyväksyntää asiakashyväksyntäpalaverissa määrittelijän toimesta. Tuotekehitysprojekti päättyy, kun tuotteen kaikki vaaditut toiminnot ja ominaisuudet on toteutettu, ja ne on testattu (hyväksyttynä ja tulos raportoituna järjestelmäntestausraportilla), verifioitu ja validoitu, ja tuote tai tuotteen uusi ominaisuus on hyväksytty myös loppuasiakkaan toimesta (lopputuotteen asiakaskatselmointi). Tuotekehitysprosessin tuloksena on hyväksytty valmis tuote, joka toimii aloitusehtona toimitusprojektille eri asiakkaille. Toimeksiantajan tuotekehitysprosessin syöttö- ja tuotostiedot ovat kuvattuna kuvassa 9. (Toimeksiantaja 2018)



Kuva 9. Tuotekehitysprosessin syöttö- ja tuotostiedot. (Toimeksiantaja 2018)

4.1.2 Tuotekehitysprosessin sidosryhmä

Tuotekehitysprosessi alkaa määrittelyprosessin päätyttyä ja päättyy vasta kun hyväksytty testitulos on saatu testausprosessista ja hyväksyntä asiakkaalta. Tuotekehitysprosessin sidosryhmiin kuuluvat siis seuraavanlaiset roolit ja vastuut, jotka ovat kuvattuna seuraavassa taulukossa:

Taulukko 4. Tuotekehitysprosessin sidosryhmät toimeksiantajan tuotekehitysprosessissa (Toimeksiantaja 2018)

Rooli	Vastuut
Tuoteomistaja	- Toimii asiakkaan edustajana tuotekehitykselle. - Toimii yhteistyössä muiden osapuolien kanssa päättäessään tuotteen ominaisuuksista, vastaa siitä, että tuote ja sen toiminnot ja ominaisuudet vastaavat asiakkaan tarvetta - Valitsee tiimin tehtävät sprinttiin yhteistyössä tiimin kanssa. - Raportoi tuotekehitysjohtajalle
Tiimipäällikkö	- Toimii oman tiimin edustajana, tiimi toimii omatoimisena scrum-tiiminä. - Toimii scrummasterina - Raportoi Tuotekehitysjohtajalle

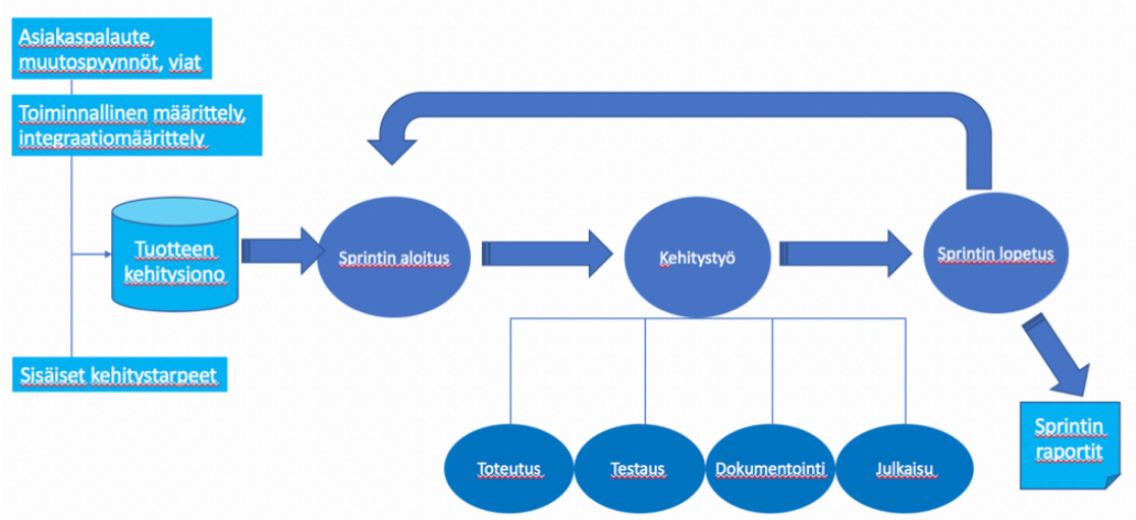
Tuotekehitysjohtaja	- Vastaa resursoinneista - Vastaa tuotekehityksen prioriteeteista - Vastaa tiimien työnjaosta - Hyväksyy tehtävät uudet ominaisuudet
Tiimi	- Toimii itseohjautuvana yksikkönä - Suorittaa sprinttiin valitut tehtävät - Valitsee sprintin tehtävät yhdessä tuoteomistajan kanssa
Arkkitehti	- Vastaa toimintojen ja ominaisuuksien teknisestä arkkitehtuurista - Vastaa tuotteiden teknisestä toteutustavasta ja teknologioista
Testaaja	- Validointi - Verifiointi - Riskianalyysi - Vastaa tuotteen laadun testauksesta - Arvioi toiminnallisen määrittelyn puitteissa valitun riskiluokan oikeellisuuden

4.1.3 Tuotekehitysprosessin vaiheet

Tuotekehitysprosessissa kehitystyö tapahtuu 2 viikon mittaisissa kehitysjaksoissa, Scrum sprinteissä. Jokaisen sprintin päätteeksi tuote julkaistaan tuotteen asiakaskohtaiselle laadun tarkkailuun osoitetulle palvelimelle, eli ns. QA-palvelimelle (Quality Assurance). Tuotetta ei kuitenkaan julkaista prosessin toimesta koskaan tuotantopalvelimelle, vaan ohjelmaversion tuotantoonsiirto tehdään omassa prosessissaan, joka suoritetaan omalla aikataulullaan tuotekehitysprosessin ja testausprosessin päätteeksi testaustiimin ja asiakasvastaavien hyväksymälle ohjelmaversiolle. (Toimeksiantaja 2018)

Yksi tuotekehitysjakso eli sprintti koostuu tuotteen kehitysjonon muodostamisesta, sprintin aloituspalaverista, varsinaisesta kehitystyöstä eli ohjelmoinnista (ja testauksesta), sekä sprintin lopetuspalaverista. Tuotteen kehitysjonoa tarkennetaan sprintin aikana tehtävissä ns. backlog grooming -tapahtumissa, jossa tiimi ja tuotepäällikkö käyvät

läpi tuotteen kehitysjonossa olevia epäselviä tehtäviä ja priorisoivat kehitysjonon tehtäviä. Tuotepäällikön tehtäviin tuoteomistajana kuuluu auttaa tiimiä ymmärtämään tuotteen vaatimuksia. Backlog grooming -tapahtumassa arvioidaan kunkin käsiteltävän tehtävän kohdalla siihen liittyvät riskit. Havaitut riskit arvioidaan niiden vakavuuden, esiintymisen ja havaittavuuden mukaisesti. Sprintin aloituspalaverissa valitaan tiimin seuraavan sprintin tehtävät tuotteen kehitysjonosta. Valintaan osallistuvat tuotepäällikkö ja tuotekehityksen scrumtiimi. Aloituspalaverissa varmistutaan siitä, että tehtävät on riittävällä tasolla määritelty ja että ne on mahdollista suorittaa seuraavan sprintin aikana. Kehitystiimi pisteyttää kehitysjonon työmäärät yleensä backlog grooming -tapahtumissa, mutta aloituspalaverissa pisteet on viimeistään lyötävä lukkoon. Pisteytyksellä pyritään kuvaamaan nimenomaan työn valmistumiseen tarvittavaa työmäärää, eikä siihen kuluvaa aikaa, sillä eri ihmiset tekevät saman työn yleensä eri ajassa, johtuen heidän yksilöllisestä osaamisestaan. Scrumtiimi tyypillisesti pystyy tekemään yhdessä sprintissä tietyn kokonaismäärän pisteitä. Sprintin päätteeksi pidetään lopetuspalaveri, jossa katselmoidaan sprintin tulos. Lopetuspalaverin yhteydessä on katselmoinnin päätteeksi sovittu pidettäväksi myös retrospektiivi, jossa käydään läpi sprintin onnistumiset ja parannusta tarvitsevat asiat. Kuvassa 10 on kuvattuna tuotekehitysprosessin vaiheet laatukäsikirjan mukaisesti. (Toimeksiantaja 2018)



Kuva 10. Tuotekehitysprosessin vaiheet. (Toimeksiantaja 2018)

Sprintin tuotekehitystyö koostuu suunnittelusta, ohjelmoimisesta, testauksesta, dokumentoinnista, julkaisusta, sekä tuotteen käyttöönotosta QA-palvelimella. Työn suunnitteluun osallistuu koko kehitystiimi ja työ suunnitellaan siten, että kaikki ymmärtävät tehtävän työn ja sen tarvitseman panoksen. Työtä tiimin jäsentenvälillä synkronoidaan

myös päivittäin pidettävässä lyhyessä päiväpalaverissa, jossa on tarkoituksena kertoa edellisen vuorokauden aikana tehdyt asiat, seuraavan vuorokauden aikana tehtävät asiat, sekä mahdolliset työtä estävät ongelmat. Ohjelmoinnin lomassa on pyrkimys suorittaa myös mahdollisimman paljon testausta, jotta tuotteen laatu olisi varmistettu sprintin päätteeksi julkaistavassa versiossa. Tehty työ on tarkoitus dokumentoida tekniseen dokumentaation sekä tuotekehitysjonon tehtäviin. (Toimeksiantaja 2018)

4.1.4 Kehityksen aikainen muutosten hallinta

Kehityksen aikaiset muutokset tunnistetaan ja tallennetaan tuotteen tehtävienhallintajärjestelmään. Muutokset katselmoidaan, aikataulutetaan, verifioidaan ja validoidaan. Huomioitavaa on, että tuotepäälliköt pitävät tuotekehityssprinttien välillä asiakaskatselmointeja väliversiolle, joita tuotekehitys on tuottanut sprinteissään. Asiakaskatselmoinneista saattaa koitua muutospyyntöjä, jotka vaikuttavat kehityksessä olevan tuotteen työmääriin ja aikatauluihin. Muutospyynnöt käsitellään backlog grooming -tapahtumissa. (Toimeksiantaja 2018)

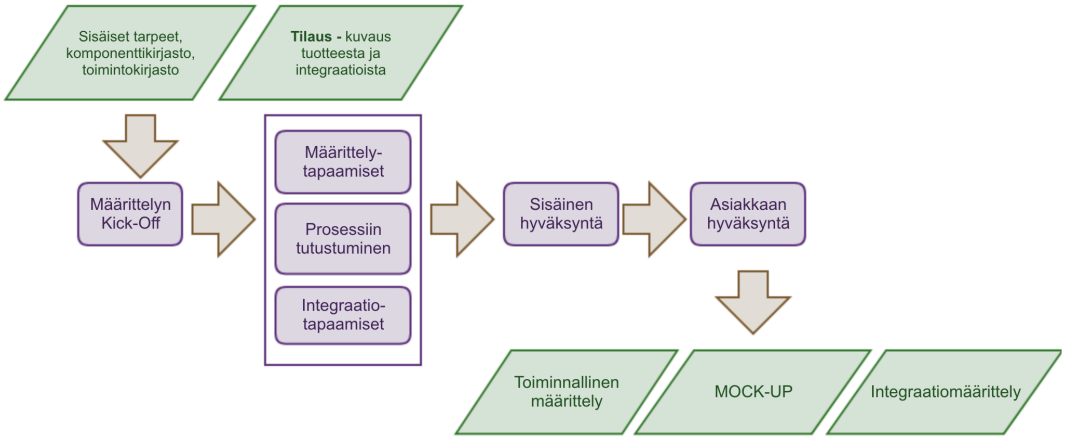
Jos muutos tehdään verifiointiaktiviteettien jo alettua, muutoksen vaikutus jo verifioituihin tuotteen osiin pitää varmistaa ja mahdollisesti uudelleen verifioida. Sprintin aikaiset hyväksytyt muutokset liitetään tuotteen kehitysjonoon, groomataan ja aikataulutetaan. Riskienhallintasuunnitelma päivitetään muutostenhallinnan yhteydessä. (Toimeksiantaja 2018)

4.1.5 Sisäinen tuotekehitysprosessi

Sisäisessä tuotekehityksessä kehitetään tai otetaan käyttöön tuotekehitykseen, tuotteiden toimittamiseen tai tuotteiden ylläpitoon liittyvä tekninen ratkaisu, mitä ulkoinen asiakas ei ole suoranaisesti tilannut. Tällaisia ratkaisuja ovat esimerkiksi teknisesti parempi näkymämahdollisuus tuotteisiin, uuden version käyttöönotto tuotteiden käyttämistä tietokannoista tai teknisesti paremman tavan kehittäminen tuotteiden ylläpitoon. Sisäinen tuotekehitysprosessi noudattaa normaalia tuotekehitysprosessia soveltuvin osin.

4.2 Määrittelyprosessi

Määrittelyprosessi on tuotekehitysprosessia edeltävä prosessi. Määrittelyprosessia johtaa tuotekehitysprojektin tuotepäällikkö. Tuotekehitysprojektin tuotepäällikkö esittelee tuotekehitysprosessin tuotekehitystiimille määrittelyprosessissa määrittelemänsä tuotteen ja osallistuu siten 2 eri prosessiin. Määrittelyprosessissa voidaan määritellä kokonaan uusi tuote tai uusia ominaisuuksia olemassa olevaan tuotteeseen. Määrittelyprosessin aloitusehtoina ovat yleiset organisaation asettamat tuotteiden perusvaatimukset liittyen arkkitehtuuriin, toimintokirjastoihin, käyttäjähallintaan, yms. ja hyväksytty asiakkaan tekemä tilaus tai muutoshallinnassa hyväksytty toiminto. Prosessiin sisältyvät määrittelyprojektin aloituspalaveri, määrittelytapaamiset, sisäinen hyväksyntäkatselmointi, sekä asiakashyväksyntäkatselmointi. Kuvassa 11 on esiteltynä toimeksiantajan määrittelyprosessin eri vaiheet.



Kuva 11. Määrittelyprosessin vaiheet. (Toimeksiantaja 2018)

Määrittelyprosessin määrittelyprojekti alkaa aloituspalaverista eli ns. kick-off -tapaamisesta. Aloituspalaverissa sovitaan asiakkaan kanssa projektin sisältö, määrittelyryhmä ja toimintatavat. Sisällön rajaukseen vaikuttavat pääasiassa sisäiset tarpeet, sekä käytettävissä olevat komponentti- ja tuotekirjastot. Pohjana määrittelyprojektin alkamiselle on projektin tilaus, jossa on karkea kuvaus tuotteesta ja integraatioista. (Toimeksiantaja 2018)

Määrittelytapaamisissa tuotetaan projektiryhmässä tuotteeseen sisältöä, toimintoja ja käydään läpi hoitoketjun prosessi ja sen liittyminen toteutettavaan tuotteeseen. Määrittelytapaamisten perusteella määrittelijä rakentaa toiminnallista hahmotelmaa tuotteesta, jolla varmistetaan tuotteen toimintatapa ja tietosisältö. Määrittelytapaamisten perusteella ja tilauksen tiedoilla määritellään toteutettavat integraatiot. Integraatioiden osalta määrittelijöiden tehtävänä on tuottaa tuotekohtainen suodatuskuva. (Toimeksiantaja 2018)

Sisäisessä hyväksyntäkatselmoinnissa esitellään määritelty tuote valitulle arkkitehdeille ja tarvittaessa tuotekehitysjohtajalle. Katselmoinnin tarkoituksena on varmistua siitä, että määritellyt toiminnot ja integraatiot ovat toteuttamiskelpoisia ja vastaavat organisaation sisäistä ohjeistusta ja aikatauluvaatimuksia eli ovat tuotestrategian (roadmap) mukaan tehtävissä. (Toimeksiantaja 2018)

Määrittelyprojekti päättyy tuotetun dokumentaation asiakashyväksyntään. Asiakashyväksynnässä projektiryhmälle ensin esitellään tehty hahmotelma tuotteesta ja tämän jälkeen toimitetaan katselmoitavaksi määrittelydokumentti. (Toimeksiantaja 2018)

4.3 Testausprosessi

Testausprosessin aloitusehtoina on, että toteutettu järjestelmä on asennettu testausympäristöön (laadun varmennukseen tarkoitettu QA-palvelin), järjestelmän asiakaskohtaisuudet on asetettu tuotteen asetustiedostoihin ja testaustiimiä on informoitu mahdollisuudesta testata. Testausprosessin tuloksena testaaja laatii testausraportin, josta selviää, että onko tuote tuotantokelpoinen vai ei. Testausraportissa listataan testissä löydetyt virheet virhekuvauksineen. Virheet kirjataan myös tuotteen kehitysjonoon järjestelmävirhe tehtävinä. Virheiden luokittelu tehdään testaussuunnitelman mukaisesti, ja niiden vaikutus raporttiin arvioidaan testaajan ja mahdollisesti myös testauspäällikön toimesta. Päätöksen tuotteen tuotantokelpoisuudesta tekee vastuullinen testaaja tai testauspäällikkö. (Toimeksiantaja 2018)

Tuotekehitystiimi on velvollinen seuraamaan testausprosessissa kirjattuja virheitä. Mikäli virheitä esiintyy, tuote palaa tuotekehitystiimin työpöydälle ja käy jälleen läpi tuotekehitysprosessin. Jottei projektille tulisi tarpeetonta viivästystä, tuotekehitystiimi varautuu mahdollisiin virhekorjauksiin sprinttiohjelmissaan.

4.4 Toimitusprosessi

Toimitusprosessi määrittelee toimenpiteet jotka toimitusprojektissa on tehtävä tuotteen toimittamiseksi asiakkaalle. Toimitusprosessin mukainen toimitusprojekti alkaa, kun tuotehallintakokous on hyväksynyt toimituksen aikataulutuksen. Toimitusprojekti loppuu kun tuotehallintakokous hyväksyy toimituksen päättyneeksi ja toimitusprojektin vetäjä merkitsee toimitusprojektin päättyneeksi. Toimitusprojektin kulku on kuvattuna kuvassa 12. Toimitusprojekti tehdään 3 vaiheessa. Ensimmäisessä vaiheessa vastuuhenkilöt täyttävät projektin tuotantoonsiirron esiehdot. Toisessa vaiheessa suoritetaan tuotteen siirto tuotantoympäristöön ja mahdollinen vanhan ohjelmaversion ylikirjoitus. Kolmannessa vaiheessa asiakas ottaa tuotteen käyttöön. (Toimeksiantaja 2018)

Kuva 12. Toimitusprojektin kulku. (Toimeksiantaja 2018)

Jos tuotantoonsiirrossa tulee ongelmia, asiakasvastaava tekee päätöksen siirrosta ja sen aikataulusta tai toimituksen takaisinvedosta. Tuotantoonsiirron aikaisen poikkeaman sattuessa takaisinvetopäätöksen voi tehdä myös siirtoa suorittava henkilö. Tällöin tapahtuma kirjataan tuotantoonsiirtoraporttiin ja mahdollinen aikaisempi versio palautetaan käyttöön. Tuotekehitystiimi on velvollinen tukemaan toimitusprojektissa tehtävän tuotteen käyttöönottoa. Mikäli toimitusvaiheessa törmätään vielä järjestelmävirheisiin tai puutteisiin, tuotekehitystiimi on velvollinen ottamaan tuotteen takaisiin omaan sprinttiohjelmaansa.

5 LÄHTÖTILANTEEN KARTOITTAMINEN

Tuotekehitysprosessin lähtötilannetta on kartoitettu sekä kvantitatiivisisin ja kvalitatiivisten tutkimusten kautta. Tuotekehitysprosessin ongelmia on ensin lähdetty tunnistamaan erilaisten mittareista kerätyn tiedon pohjalta. Kun ongelmista on saatu jokin tuntuma, saaduille olettamuksille on pyritty saamaan vahvistus digitaalisen haastattelulomakkeen avulla. Tutkimuksen rajaamiseksi, haastattelututkimus on suoritettu tiettyjä olettamia vasten ja haastattelun avulla on pyritty vahvistamaan oletettujen ongelmien olemassaolo ja niiden laajuus. Koska kehityshankkeen päämääräksi on asetettu prosessin tehostaminen ja nopeuttaminen jopa 30%, myös tutkimus ja olettamukset keskittyvät läpivientiaikojen ja läpivientiin liittyvien ongelmien tutkimiseen. Jotta tiedettäisiin mitä prosessissa voidaan läpimenoajan osalta tehostaa, on selvitettävä missä vaiheessa prosessia ongelmia esiintyy.

Ensimmäinen tutkimukselle asetettu olettama on, että tuotekehityksen ketterät menetelmät koetaan kyllä tarpeellisiksi, mutta niitä ei tehdä akateemisesti oikein. Tämä tarkoittaa sitä, että vaikka tapahtumat järjestetään ja vaikka asianomaiset ihmiset niihin osallistuisivatkin, tapahtumissa ei käsitellä oikeita asioita, niissä ei hyödynnetä jo tarjolla olevaa tietoa, niissä ei opita virheistä ja ne voisivat olla huomattavasti tehokkaampia tapahtumia. Vaikka Scrum-viitekehyksen mukaiset tapahtumat järjestetään, niiden tärkeyttä ei ymmärretä, eikä niitä siten käytetä oikein. Ketterälle kehitykselle asetetut säännöt rikotaan, mikä ilmenee mm. palavereihin valmistautumisen puutteena tai osallistumatta jättämisenä, sekä tuotekehityksen tukemattomuutena. Ongelma on huomioitu mm. tuotekehityssprinttien retrospektiiveissä, joiden pöytäkirjat ovat toimeksiantajan sisäisessä verkossa yleisesti nähtävillä, mutta myös kehityskeskustelujen yhteydessä, ja ovat olleet jatkuvasti esillä myös ns. kahvipöytäkeskusteluissa.

Toinen tutkimukselle asetettu olettama on, että kaikki eivät ymmärrä tuotekehityksen menetelmiä, niiden käyttötarkoitusta tai niiden tärkeyttä. Olettama pohjautuu tuotekehityssprintin palaverien muistioihin (ml. aloituspalaverimuistiot, päätöspalaverimuistiot, sekä retrospektiivit) pitkän ajanjakson aikana kirjattuihin havaintoihin siitä, että tuotekehityssprintin tapahtumia ei priorisoida tärkeäksi tapahtumaksi muiden tehtävien töiden joukossa. Ongelma ilmenee palavereihin valmistautumisen puutteena tai osallistumatta jättämisenä.

Kolmantena tutkimusta rajaavana olettamana tutkimukselle on asetettu, että Lean-ajatusmaailman mukaisista hävikkityypeistä odotus on tuotekehityksen kannalta suurin hävikkiä tuottava ongelma. Kolmas olettama pohjautuu kvantitatiivisella tutkimuksella tehtyyn tutkimustulokseen, jossa tuotekehitysprojektien odotusajat ovat ajoittain jopa pidempiä kuin aktiiviseen työstämiseen käytetty aika, mikä ilmenee mm. liitteen 1 läpimenoaikoja kuvanneesta taulukosta ja kuvasta.

Lisäksi tutkimuksessa on oletettu, että suurin odotusta aiheuttava ongelma on, että asiakkaan tahtotilaa ei saada tuotua tuotekehitykselle asti niin, että tuotekehitys ymmärtäisi sen. Olettama toimii tutkimuksen neljäntenä olettamana. Neljäs olettama pohjautuu tuotekehitysprinttien retrospektiiveissä tehtyihin havaintoihin määrittelyiden muuttumisesta tai puutteellisuudesta, sekä kirjattuihin muutosten määrään. Muutokset tulevat kehitystiimin tietoon liian myöhäisessä vaiheessa, jotta voitaisiin vielä pysyä ennalta asetetussa aikataulussa.

5.1 Mittarit

Tuotekehitysprosessin ongelmia ja tutkimuksen lähtötilannetta tutkittiin kvantitatiivisin menetelmin, asettamalla tutkimukseen 3 seurattavaa mittaria. Mittareina käytettiin projektien läpivientiaikaa, tuotteiden toiminnan laatua, sekä hyväksyntäkatselmoinnin jälkeen tehtävien muutosten määrää. Mittareista yksi seuraa siis aikataulua ja kaksi laatua. Muutokset tulevat useimmiten asiakaskatselmoinnista, joten läpivientiaikaa on mitattu aloituksesta katselmointiin ja katselmoinnista valmistumiseen.

Läpivientiaika on yksi yleisimmistä projektinhallinnan mittareita. Läpivientiaika on tärkeä, sillä mitä nopeammin päästään markkinoille tai tuote otetaan käyttöön, sitä paremmassa asemassa tuotteen tai palvelun tarjoaja on. Kilpailun ollessa markkinoilla kovaa, projektien läpivientiajat ja elinkaaret lyhenevät jatkuvasti. Ensimmäisenä markkinoilla olevalla on aina etulyöntiasema, sillä hänellä on valmis tuote. Tuotekehitysprosessin läpivientiaikaa analysoitaessa merkityksellistä on, että kuinka iso tiimi tuotekehitykseen on osallistunut ja kuinka monta päivää tuotekehitysprosessiin käytettiin aikaa per resurssi. Merkityksellistä on toki myös se, että onko projekti onnistunut vai ei. Lisäksi eri projektit voivat olla eri kokoisia. Kehittämishankkeen toimeksiantajalla projektit ovat usein varsin samanlaisia, joten voidaan olettaa niiden läpivientiaikojen olevan vertailtavia. Projektit jaetaan kolmeen eri kategoriaan: helppo (ja nopea), keskivaikea (ja keskipitkä), sekä vaikea (ja pitkä). Läpivientiaikaa tarkasteltaessa lasketaan projektiin

käytettyjä henkilötyöpäiviä. Määrittelyprosessin aikana projektin suuruudesta tehdään arvio henkilötyöpäivinä. Arvion avulla projekti aikataulutetaan tuotekehitystiimien tehtäväksi. Lopullinen projektiin käytetty henkilötyöpäivämäärän suuruus saattaa kuitenkin olla paljon pienempi tai paljon suurempi, mikä aiheuttaa molemmissa tapauksissa sekaannusta aikatauluihin ja painetta prosessien ja työntekijöiden keskinäiseen synkronointiin. (Chin 2003, 123 -126; Toimeksiantaja 2018)

Testausprosessin lopputuotoksena on testausraportti, jossa on kuvattuna tuotekehityksessä syntyneiden ongelmien määrä ja sisäinen laatu. Testausraportin avulla voidaan siis seurata tuotteen sisäisen laadun kehittymistä. Sisäisellä laadulla tarkoitetaan tuotteen ominaisuuksien toimivuutta, sekä yhdenmukaisuutta määrittelyn kanssa. Huono sisäinen laatu aiheuttaa paljon lisätyötä ja kustannuksia. Sen vaikutuksia asiakkaaseen ovat muun muassa mahdollinen toimitusaikojen venyminen, sekä huonon laadun kulkeutuminen loppuasiakkaalle asti. Ulkoisia laatukustannuksia ovat ulkoisilla asiakkailla ilmenneiden virheellisten tuotteiden aiheuttamat hyvitykset ja korjaukset. (Haverila 2009, 376; Värpiö 2017, 6)

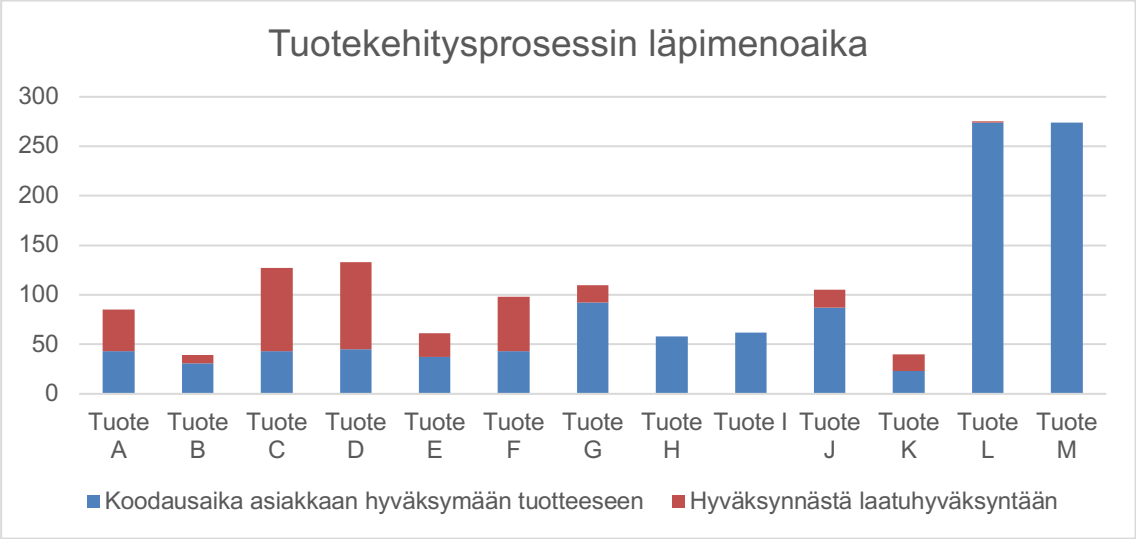
Testausraportti tuotetaan toimeksiantajan prosesseissa ensimmäisen kerran vasta tuotekehitysprosessin hyväksyntäkatselmoinnin jälkeen. Testausraportti on tehty aina julkaistua tuoteversiota vasten. Sisäistä laatua seurataan ja testataan myös ennen ensimmäisen testausraportin julkaisua, mutta tällöin testaustulokset jaetaan suoraan kehittäjälle, usein vain sanallisesti, jotta kehittäjä voi korjata tekemänsä virheet välittömästi jo ennen version julkaisua. Koska toimeksiantajan tässä kehittämishankkeessa tutkittavat ohjelmistoprojektit rakennetaan kaikki yhteisen alustan päälle, edellisen projektin testauskierros lähtökohtaisesti kasvattaa seuraavan projektin laatua. Alustaa kehitetään ja laajennetaan lähes jokaisen projektin yhteydessä, joten projekti on myös aina lähtökohtaisesti edellistä isompi, sillä siinä voidaan olettaa olevan kaikki edellisen projektin ominaisuudet, mutta myös hieman omaa väriä lisänä. Bugien laatu ja niiden määrät eivät siis ole suoraan verrattavia, mutta kuitenkin vahvasti suuntaa antavia.

Tuotteen ominaisuuksien määrittely -dokumentti ja tehdyn määrittelytyön laatu vaikuttavat tuotekehitysprosessiin ja siinä toteutettavaan projektin viivästymiseen oleellisesti. Tuotekehityksen aikana tehtävistä määrittelymuutoksista on tuotettava muutostenhallintadokumentti. Muutostenhallinnan avulla voidaan seurata muutosten määrää ja siten määrittelyiden elävyyttä ja laatua.

5.1.1 Läpivientiaika

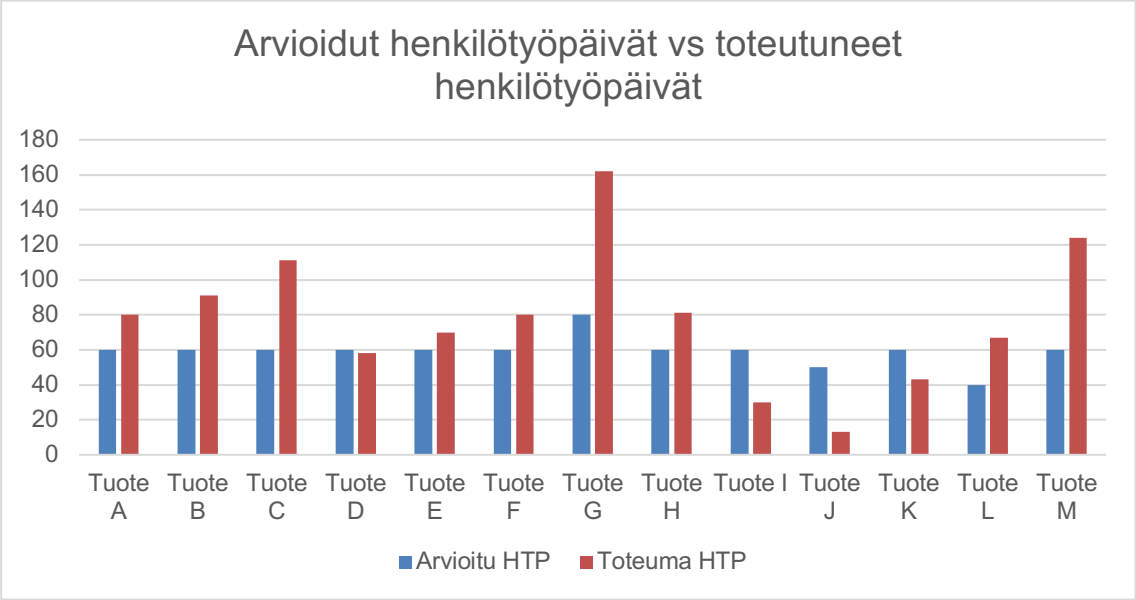
Projektien läpivientiaikaa on tutkittu 13 satunnaisesti valitulla tuotekehitysprojektilla. Projektit ovat aika järjestyksessä tuoreimmasta vanhimpaan, mutta eivät ole välttämättä peräkkäisiä projekteja. Projekteille yhtenäistä on, että ne on tehty samaa alustaa käyttäen. Yhtenäiselle alustalle tuotteita rakentamalla on pyritty yhtenäistämään tuotteita, sekä tekemään kehityksestä selkeämpää ja helpompaa. Alustan tehtävänä on tarjota valmiita ratkaisuja usein esiintyviin ongelmiin ja tilanteisiin. Alusta kehittyy lähes jokaisen projektin myötä, joten sen tulisi tarjota seuraavalle tuotteelle aina aiempaa enemmän. Huomioitavaa kuitenkin on, että ominaisuuksien ja työkalujen määrän kasvaessa, myös projektin koko kasvaa, vaikka enemmän ja enemmän ratkaisuja olisikin valmiiksi toteutettuna tai ainakin helpommin tehtävissä.

Kuvaajassa 1 on mitattu toimeksiantajan tuotekehitysprosessin läpimenoaikoja 13 satunnaisesti valitulla tuotekehitysprojektilla. Kuvaajan data pohjautuu liitteen 1 taulukoihin. Koska kaikki taulukon (ja kuvaajan) projektit jakavat saman ohjelmistoalustan, läpimenoaikojen tulisi lähtökohtaisesti nopeutua, mikäli projektien ominaisuuksien laajuus on sama. Vaikutus on luettavissa myös kuvaajasta 1, sillä sininen palkki kertoo koodausajasta asiakkaan kanssa tehtävään hyväksyntäkatselmointiin. Hyväksyntäkatselmoinnissa tulee usein vielä muutoksia tuotteen ominaisuuksien määrittelyyn, joten ominaisuuksien kehittäminen jatkuu myös hyväksyntäkatselmoinnin jälkeen. Hyväksyntäkatselmoinnin aikoihin testaustiimi alkaa myös testata tuotteen sisäistä laatua. Kuvaajassa 1 punaisella värillä nähdään aika, joka projektiin on käytetty hyväksyntäkatselmoinnin jälkeen. Se siis sisältää määrittelyiden muutoksien vaatiman ajan, sekä ohjelmistovirheiden korjaukseen käytetyn ajan. Punainen palkki on kasvussa, mikä oletuksena kielii siitä, että muutoksien ja korjattavien asioiden määrä olisi lisääntynyt ajan saatossa. Tuotekehitysprosessi on kuitenkin valmis vasta kun tuote on siirretty tuotantopalvelimelle ja käyttöönotettu. Hyväksyntäkatselmoinnin jälkeinen aika siis kasvaa, mikäli tuotetta ei saada julkaistua. Julkaisulle puolestaan oma toimitusprosessinsa, jonka tehtäviä priorisoidaan tuotehallinnan kautta. Huomioitavaa kuitenkin on, että kaikki hyväksyntäkatselmoinnin jälkeinen aika tuotekehitysprosessin Lean-periaatteiden mukaisesti hukkaa. Projekteista 3:ssa (tuote C, D ja F) on mennyt enemmän aikaa hyväksyntäkatselmoinnin jälkeen, kuin ennen sitä. Tuotteen A kohdalla päivän vajaa yhtä paljon. Kuvaajasta on myös todettavissa että viivästyminen on ollut uudemmissa tuotteissa tavanomaista ja jopa kasvavaa.



Kuvaaja 1. Tuotekehitysprosessin läpimenoaikoja. Kuvaajassa sinisellä koodausaika kalenteriajassa asiakkaan hyväksymään versioon, sekä punaisella aika hyväksynnän jälkeen käytetystä kalenteriajasta.

Kuvaajassa 2 on verrattu arvioituja työmääriä toteutuneisiin työmääriin. Myös kuvaajan 2 data pohjautuu liitteen 1 taulukoihin. Epäonnistunut arvio on aina epäonnistunut, olisi arvio sitten liian suuri tai liian pieni. Organisaation eri osastojen tekemistä ja työntekijöiden töitä ohjaava roadmap elää, mikäli arviot eivät osu kodalleen. Ongelmana roadmapin elämisessä on, että suunnitellut työt eivät eri osastojen välillä eivät pysy synkronissa. Työmääräarvio projektille annetaan määrittelyprosessin yhteydessä yleensä tuotekehitysarkkitehtien toimesta. Arvioitaessa työaikaa tuotteen ominaisuuksien määrittely ei kuitenkaan aina ole täysin valmista ja muutoksia määrityksiin tulee poikkeuksetta myös kesken tuotekehitysprosessin. Arvion antajat ovat saattaneet vaihdella, mutta kaikki arviot kuitenkin esitellään ja hyväksytään tuotekehityksen tuotehallintapalavereissa joka toinen viikko. Tuotehallintapalaverissa päätetään mm. projektien prioriteeteista, aikatauluista ja tekijöistä. Kuvaajassa 2 on huomattavissa, että työmäärä arviot ovat harvoin osuneet yläkanttiin. Tarkastelluista projekteista 4:ssä (13:sta) arvio on ollut ylimitoitettu. Projekteista 8:ssa (tuotteissa A, B, C, F, G, H, L ja M) arvio on ylitetty yli 20%:lla. Projekteista 3:ssa (tuotteet I, J ja K) arvio on alitettu yli 25%:lla. Arvioiden heitot ovat siis usein todella isoja verrattuna toteutuneeseen työtön.



Kuvaaja 2. Tuotekehitysprojektien ennalta arvioidut työmäärät henkilötyöpäivinä verrattuna toteutuneisiin työmääriin henkilötyöpäivinä.

Tavoite siitä, että tuotekehitysprosessia nopeutettaisiin 30%:lla, olisi lähellä täyttymistänsä jo kun hyväksyntäkatselmoinnin jälkeinen viivästely saataisiin poistettua. Muussa tapauksessa arvioita työmäärän koosta on myös selkeästi nostettava tai määrittelyä kavennettava. Mitatut toteutuneet henkilötyöpäivät sisältävät myös työn hyväksyntäkatselmoinnin jälkeen. Arviot työn määrästä osuisivat paremmin kohdallensa, mikäli hyväksyntäkatselmoinnin jälkeistä työtä saataisiin vähennettyä tai jopa kokonaan poistettua.

5.1.2 Laatu

Eräs perinteinen laatumääritelmä niin ohjelmistotuotannossa kuin muussakin teollisuudessa on Philip Crosbyn (1979) esittämä kuvaus: "Laatu tarkoittaa yhteensopivuutta vaatimusten kanssa". Ohjelmiston laatu mittaa ohjelmistotuotannossa sitä, kuinka hyvin tietokoneohjelma tai ohjelmisto on suunniteltu ja kuinka hyvin toteutettu ohjelmisto noudattaa suunnitelmaa. Siinä missä suunnitelmanmukaisuus keskittyy siihen vastaako toteutus ohjelmiston suunnitelmaa, suunnittelun laatu mittaa kuinka kelvollisia suunnitelma ja määrittelydokumentin vaatimukset ovat onnistuneen ohjelmistotuotteen laatimiseen.

Bugi on ohjelmointivirhe ohjelman lähdekoodissa. Bugien vakavuusaste merkitään toimeksiantajaorganisaatiossa 5 tasoisella asteikolla seuraavanlaisesti:

- **Blokkaava** bugi on ohjelmointivirhe, joka tulee korjata heti, sillä se on ohjelman käyttöä estävä. Korkein luokitus korostaa asian kriittisyyttä ja nopeaa reagointitarvetta. Blokkaavat bugit pitäisi saada korjattua viimeistään seuraavaan versioon (tai konfiguraatioon).
- **Kriittinen** bugi on ehdottomasti korjattava nopealla aikataululla ennen käyttöönottoa, mutta korjausta ei tarvitse välttämättä tehdä välittömästi, koska tuotetta voidaan käyttää ilmankin. Yhtään kriittisentason bugia ei saa olla avoinna kun tuoteversio käyttöönotetaan.
- **Merkittävä** bugi on kun jokin asia ei toimi ja haittaa merkittävästi toimintaa. Vaikutukseltaan merkittävät bugit olisi hyvä korjata ennen käyttöönottoa, mutta asiakasvastaavan päätöksellä se voidaan siirtää korjattavaksi seuraavaan tuoteversioon.
- **Vähäpätöinen** bugi merkitään kun jokin toiminto ei toimi, mutta sen välttämiseksi on olemassa helppo kiertotie tai asia ei haittaa käyttäjää merkittävästi.
- **Triviaali** bugi on lähinnä kosmeettinen virhe, kuten kirjoitusvirhe tai hieman väärin sijoitettu komponentti.

Taulukossa 5 on esitettynä tuotekehitysprosessin läpikäyneisiin tuotteisiin kirjattujen eri tasoisten bugien määrä vuosikohtaisesti. Bugien määrä on vuonna 2018 laskenut merkittävästi, vaikka organisaation koko ja tuotantokyky on vuonna 2018 ollut suurempi kuin vuonna 2017. Bugien määrä on laskenut kaikilla vakavuusasteilla. Muutos parempaan laatuun voidaan todeta johtuvan kokemuksen lisääntymisestä, sekä ohjelmistoalustan kypsymisestä tai sitten kirjauskäytäntöjen muuttumisesta. Vuosien 2016 ja 2017 käännekohdassa on tehty isoja uudistuksia ohjelmistoalustaan, jotka selittävät ohjelmavirheiden määrän kasvun vuoteen 2015 nähden.

Taulukko 5. Kirjattujen bugien määrä vuosittain.

Vuosi	Blokkaavat	Kriittiset	Merkittävät	Yhteensä	Yhä avoimet
2018	50	180	1107	1337	369
2017	56	230	1418	1704	394
2016	19	140	1496	1655	543
2015	29	162	1034	1225	253
2014	52	135	848	1035	129
2013	80	141	689	910	164

Bugien määrän kehittymistä voidaan seurata hieman tarkemmin, kun tilastoidaan tuotekohtaisten bugien määrän esiintymistä. Taulukossa 6 on esitettynä bugien määrä tuotekohtaisesti. Taulukkoon on valittu satunnainen joukko tuotteita, jotka ovat keskenään suunnilleen saman kokoisia. Taulukon rivit on järjestetty tuotteiden toteutusjärjestyksessä vanhimmasta tuoreimpaan. Joukko on satunnainen, eivätkä tuotteet välttämättä ole peräkkäisiä projekteja. Taulukon 6 esittämistä bugien määrästä voitaisiin todeta taulukon 5 tapaan, että bugien määrä tuotekohtaisesti on vähentynyt alustan kehittyessä ja kokemuksen karttuessa, vaikka yrityksen ja tuotannon koko ovat kasvaneet vuosittain. Toisaalta, tulos ei ole yksinään luotettava, sillä bugien kirjauskäytännöt ovat saattaneet myös muuttua vuosien varrella. Ketterää kehitystä suoritettaessa, vähintään 1 testaaja on jatkuvasti mukana tuotekehityksessä. Bugeja saatetaan korjata heti ilman erillistä kirjausta, sillä myös turha kirjaaminen on hävikkiä, mikäli niiden korjaus saadaan hallittua ilman erityistä kirjallista virhekuvausta. Mikäli bugien määrän lasku perustuu kirjauskäytäntöjen uudistumiseen, ei sekään ole tuotekehitysprosessin kehittämisen kannalta huono asia, sillä kaikenlainen hävikin poistaminen tehostaa ja yksinkertaistaa prosessia. Vähempään kirjaamiseen tulisi jopa kannustaa, mikäli kirjaaminen ei ole bugien hallinnan kannalta tarpeellista.

Taulukko 6. Kirjattujen bugien määrä tuotekohtaisesti aikajärjestyksessä.

Tuote	Bugimäärä
Tuote 1 (2013)	323
Tuote 2 (2013)	227
Tuote 3 (2013)	250
Tuote 4 (2014)	215
Tuote 5 (2014)	314
Tuote 6 (2014)	288
Tuote 7 (2014)	394
Tuote 8 (2014)	309
Tuote 9 (2014)	213
Tuote 10 (2015)	324
Tuote 11 (2015)	210
Tuote 12 (2016)	195
Tuote 13 (2016)	114

Tuote 14 (2017)	170
Tuote 15 (2017)	44
Tuote 16 (2017)	57
Tuote 17 (2018)	78
Tuote 18 (2018)	53
Tuote 19 (2018)	86
Tuote 20 (2018)	44

5.1.3 Muutostyön määrä

Tuotteeseen tehtävät harkitut muutokset nostavat aina lähtökohtaisesti tuotteen arvoa, mutta arvon kasvu tapahtuu tällöin aina työmäärän ja aikataulujen kustannuksella. Mikäli tuotekehitysprosessia halutaan ajaa määrittely- ja aikataulupainotteisesti, kaikki muutokset ovat hävikkiä ja niitä tulisi välttää.

Osittain johtuen siitä, että tuoteomistajat eivät ehdi seurata tuotteen kehitystä ja asiakaskatselmointeja pyritään pitämään niiden järjestämisen hankaluuden vuoksi mahdollisimman vähän, suurin osa tuotteen muutoksista kirjataan vasta tuotekehityksen loppuvaiheissa pidettävästä asiakaskatselmoinnista (tai useammasta -katselmoinnista). Mitä varhaisemmassa vaiheessa muutostarpeet saataisiin kuvattua, sitä helpompaa niihin on tuotekehitysprosessissa reagoida. Katselmointimuutospyynnöt kirjataan tuotteen kehitysjonoon, joten niiden työmäärää voidaan verrata alkuperäiseen suunnitelmaan työmäärästä. Näin voidaan tilastoida, kuinka iso osa ajasta tuotekehitysprosessissa menee katselmointimuutosten tekemiseen. Liitteen 1 taulukossa on kuvattu asiakashyväksynnän jälkeistä kalenteriaikaa, johon sisältyvät mm. katselmointimuutosten toteuttaminen, mutta ko. taulukon hävikkiajat saattavat pohjautua mm. kommunikaatioongelmiin, eikä suureen muutostyömäärään.

Taulukossa 7 on esitettynä 19 satunnaisen tuotteen katselmointimuutosten työmäärät tarinapisteinä. Tarinapisteitä käytetään arvioimaan työmäärää yksittäisille tehtävälistan käyttäjätarinoille. Tarinapiste on suhteellinen mittayksikkö, jota käytetään suhteellisten työmääräarvioiden antamiseen. Ideana tarinapisteissä on sopia yhteinen tapa arvioida työmäärän kokoa, riippumatta siitä, kuka tiimistä tehtävän tekee. Eri taitoiset tiimin jäsenet suorittavat määrittelyn asettamat velvoitteet ja toiminnot eri ajassa, mutta työ-

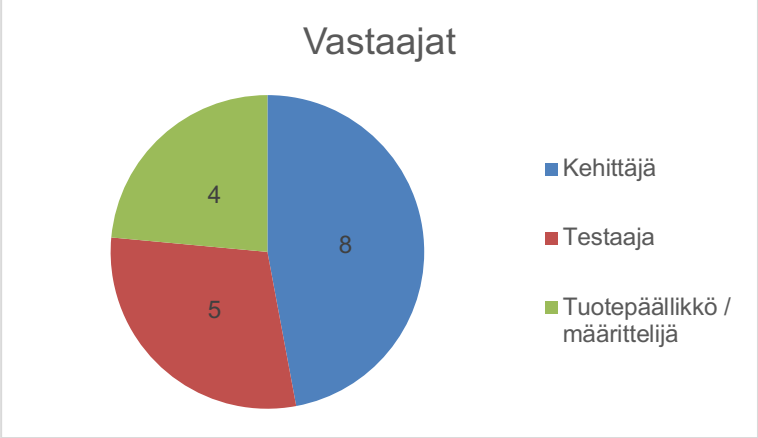
määrä on silti kaikille sama. Taulukon 7 lukujen tehtävänä on esittää hajontaa työmäärästä katselmoinnin jälkeen. Poistamalla joukosta 2 selkeästi muista poikkeavaa arvoa, keskiarvoksi työmäärälle muodostuu n. 11 pistettä (ennen poistoa 15). Mediaani poistojen jälkeen on 9 pistettä (11 ennen poistoa) ja keskihajonta n. 6 pistettä. Keskihajonta katselmointimuutosten työmäärästä voidaan todeta olevan melko maltillinen ja kohtuullisen hyvin ennustettavissa. Koska katselmointimuutoksia tuntuu tämän otoksen mukaan kuitenkin poikkeuksetta tulevan, kannattaisi katselmointeja pitää useammin, jotta palautetta saataisiin aikaisemmassa vaiheessa.

Taulukko 7. Tuotteiden katselmointimuutosten työmäärät tarinapisteinä.

Tuote	Pisteet
Tuote 1	5
Tuote 2	6
Tuote 3	6
Tuote 4	7
Tuote 5	7
Tuote 6	7
Tuote 7	8
Tuote 8	9
Tuote 9	9
Tuote 10	11
Tuote 11	12
Tuote 12	12
Tuote 13	15
Tuote 14	16
Tuote 15	16
Tuote 16	23
Tuote 17	26
~~Tuote 18~~	~~43~~
~~Tuote 19~~	~~54~~

5.2 Haastattelu

Haastattelu suoritettiin 24 kysymyksen digitaalisena lomakkeena (Liite 2). Haastatteluun vastattiin anonyyminä, mutta tuotekehitysprosessin rooli ensimmäisenä vastauksena antaen. Haastattelulomake jaettiin tuotekehitysprosessissa työskentelevälle 12 koodaajalle, 6 tuotepäällikölle, sekä 7 testaajalle eli yhteensä 25 henkilölle. Henkilöt valittiin siten että he ovat työskennelleet tutkimuksessa mukana olevien projektien parissa. Vastaukset saatiin 8 kehittäjältä, 5 testaajalta, sekä 4 tuotepäälliköltä eli yhteensä 17 henkilöltä (kuvaaja 3). Puuttumaan jäi siis yhteensä 8 henkilöä tasaisesti jokaisesta rooliryhmästä.



Kuvaaja 3. Haastatteluun vastanneiden henkilöiden lukumäärä rooleittain.

Haastattelun tarkoituksena oli vahvistaa tutkimukselle asetetut olettamat, sekä saada syventävää tietoa oletetuista ongelmakohdista. Haastattelulomakkeelle valituilla kysymyksillä pyrittiin siis vahvistamaan olettamat siitä, että tuotekehityksen menetelmiä ei ymmärretä eikä suoriteta akateemisesti oikein, odotus on Lean-periaatteiden mukainen tuotekehityksen suurin hävikkiä aiheuttava ongelma ja että tuotekehitysprosessin suurin odotusta aiheuttava ongelma on, että asiakkaan tahtotilaa ei saada tuotua tuotekehitykselle asti. Haastattelulomakkeen 1. kysymyksellä tunnistetaan vastaajan rooli tuotekehitysprosessissa (koodaaja, tuotepäällikkö tai testaaja). Näin voidaan tutkia tuloksia eri roolien näkökulmista. 3 seuraavalla kysymyksellä pyritään keräämään tietoa ketterän projektikolmion jokaisesta sivuista: aikataulurajoitteesta, laadusta ja arvosta. Seuraavat 4 kysymystä eli kysymykset 5 - 8 ovat odotusta aiheuttavaan ongelmaan syventymistä. Kysymys 6 on vastattavissa vain, mikäli kysymykseen 5 ”Onko asiakkaan tahtotilan tuotekehitykselle asti saaminen suurin odotusta aiheuttava ongelma?”,

vastattiin ”Ei”. Kysymykset 9 - 11 puolestaan käsittelevät eri rooleilta tuen saamista tuotekehitysprosessin aikana. Kysymykset 12 -16 kysyvät tuotteen ja määrittelyn laadusta ja kysymykset 17 – 24 tuotekehityssprintin eri tapahtumien hyödyllisyydestä. Jokainen osa-alue (ensimmäistä lukuun ottamatta) pitää sisällään monivalintakysymyksen, sekä valintaa täsmentävän vapaan sanan lisätietokentän, esim. perusteluille. Näin saadaan sekä tilastoitavaa dataa, että syventäviä asioita esille tuovia ja ajatusta vaativia vastauksia. Perusteluiden pohjalta voidaan nähdä myös, miten kukin vastaaja ongelman ymmärtää ja onko kysymys ymmärretty oikein. Perustelun myötä myöskään summamutikassa vastaaminen satunnaisin arvoin ei onnistu.

5.2.1 Tuotepäälliköiden mielipide

Tuotepäälliköiden päällimmäiset tehtävät tuotekehitysprosessissa tulisi olla mm. tehtävälistasta, määrittelyistä ja riskeistä huolen pitäminen, katselmointihyväksynnän hakeminen asiakkaalta tuotteelle, sekä tuotekehitystiimin tukeminen.

Liitteen 3 esittelemien haastattelussa annettujen vastausten perusteella, tuotepäälliköt kokevat, että heidän on hankalaa saada selvitetty asiakkaan tahtotilaa. He kokevat, että asiakas ei itsekään tiedä, mitä hän kehitettävältä tuotteelta pohjimmiltaan haluaa. Eri sprinttipalaverien hyödyllisyydestä tuotepäälliköiden mielipide jakautui kovasti. Yhtenäiseksi mielipiteeksi vastauksista voitaisiin päätellä, että sprintin aloitus- ja lopetuspalaverissa ei koeta tarpeelliseksi käydä läpi tehtäviä kovin tarkalla tasolla, sillä kaikki määritellyt ominaisuudet on joka tapauksessa toteutettava, eikä heille nykyprosessissa mahdollisteta kuin yksi asiakaskatselmointi. Backlog Groomingin eli tuotteen kehitysjonon työstämisen hyödyllisyydestä he olivat yhtä mieltä ja kokivat sen hyväksi tapahtumaksi ja sitä toivottaisiin käytettävän enemmän. Ongelmana kaikessa on kuitenkin kiireelliset aikataulut ja yliresursointi, sillä kaikki haastatteluun osallistuneet tuotepäälliköt mainitsevat aikataulukiireiden vaikuttavan tehtävän työn laatuun niin määrittelyn kuin ominaisuuksienkin osalta.

5.2.2 Kehittäjien mielipide

Kehittäjien tehtävänä tuotekehitystiimissä on ohjelmoida tuote annettujen määrittelyiden pohjalta valmiiksi. Haastattelussa annettujen vastausten perusteella (liite 3) kehittäjät ovat yksimielisiä siitä, että suurin tuotekehitysprosessissa odotusta aiheuttava

tekijä on, että tuotekehitys ei saa riittävän tarkasti tietoonsa määrittelyjä tai asiakkaan tahtotilaa. Odotteluun kuluu liikaa aikaa ja ilman tarkempia määrittelyjä tehdyt päätökset joudutaan usein tekemään uudelleen. Ketterän kehityksen perusperiaatteisiin kuuluu, että tuotetta voidaan määritellä uudelleen, mutta kehittäjät kokevat saamansa linjaukset liian myöhäisessä vaiheessa, jolloin ollaan jo tilanteessa, jolloin tuotteen pitäisi olla valmis ja tuotekehitysprosessin tulisi päättyä. Kehittäjät kokevat että tuotteen määrittelijät eli tuotepäälliköt eivät ole riittävän hyvin tavoitettavissa. Tuote jää jatkuvaan epävarmaan tilaan ja ominaisuuksissa joudutaan tekemään kompromisseja. Sprinttipalavereissa ei saada asioita päätökseen, sillä kehittäjät kokevat, että aloitus- ja lopetuspalaverien, sekä retrospektiivien tärkeyttä ei ymmärretä organisaatiossa. Vaikka vastauksissa eri sprinttipalaverien tärkeydestä olikin hajontaa, vastauksiin liittyneet perustelut osoittivat, että hajonta johtui suurimmaksi osaksi siitä, että kysymys ymmärrettiin 2 eri tavalla:

- Sprinttipalaveri koettiin vähemmän tärkeäksi, koska tuoteomistajia ei saada osallistumaan palavereihin
- Sprinttipalaveri koettiin tärkeäksi, sillä se on niitä harvinaisia tilaisuuksia, jossa tuoteomistajaan ylipäänsä saadaan säännöllisesti yhteyttä.

Kehittäjät kaipaisivat enemmän tukea tuotekehityssprinttien aikana eli tuotepäälliköiden tulisi osallistua enemmän sprintin toimintaan.

5.2.3 Testaajien mielipide

Testaajat ovat tuotekehitysprosessin ongelmista suurimmaksi osaksi samaa mieltä kuin kehittäjät. Myös testaajat kokevat sprinttipalaverit tehottomiksi koska niissä ei saada tehtyä tuotetta koskevia päätöksiä riittävän tehokkaasti eikä niissä saada varmistusta tehdyille asioille riittävän usein. Suurimmaksi odotusta aiheuttavaksi ongelmaksi testaajat esittivät lähes yksimielisesti sen, että tuoteomistajat keskittyvät sprinttien aikana muihin projekteihin. Myös testaajat kaipaisivat enemmän tuoteomistajien panosta sprinttien aikana, jotta määrittelyt pysyisivät ajantasaisina ja asiakkaan tahtotila saataisiin selkeästi esille. (Liite 3)

6 ONGELMAKOHTIEN TARKASTELU

Haastattelututkimuksen vastauksista on tunnistettavissa useita selkeitä ongelmakohtia hankkeen toimeksiantajan tuotekehitysprosessissa. Toimeksiantajan tuotekehitysprosessia on onnistuttu kehittämään vuosien aikana jo paljon, mutta haastattelun tuloksista tehdyt havainnot antavat ymmärtää, että muutamia selkeitä kehittämiskohteita löytyy yhä. Tämän luvun tarkoituksena on tarkastella haastattelutuloksia, nostaa esille muutamia selkeimpiä ja tuotekehitysprosessin tehokkuuteen vaikuttavimpia ongelmakohtia.

Antti Auer (2013) kirjoittaa suurimmista läpimenoaikaan negatiivisesti vaikuttavista ongelmista kirjassaan Ketterää kehitystä (2013, 26 - 27). Pelkästään ketteröimällä määrittelyä, toteutusta ja testausta, ei saada tuotekehityksen läpimenoaikaa lyhennettyä. Auer nostaa esille 3 läpimenoaikaan eniten vaikuttavaa tekijää, joista jokainen koetaan ongelmaksi myös haastattelututkimuksen pohjalta (Liite 3):

1. Päätösmenettelyn raskaus
2. Monta samanaikaista projektia
3. Tuotteen tarpeeton monimutkaisuus

Auer kuvaa raskaat päätösmenettelyt yhdeksi projektin ongelmista. Samasta ongelmasta kirjoittaa Kari Mäkelä (2006). Päätöksen tekoon osallistuu usein iso joukko ihmisiä, jotka joutuvat usein hakemaan hyväksyntää vielä kukin omilta tahoiltaan. Päätöksen tekoon osallistuva joukko voi koostua asiakkaan tai useiden asiakkaiden edustajista ja muista yhteistyökumppaneista, mutta myös yrityksen omasta henkilökunnasta kuten tuotepäälliköistä tai kehittäjistä. Läpimenoajan sanotaan lyhenevän merkittävästi, mikäli päätökset tulevat yhdeltä ihmiseltä (tai pieneltä porukalta). Myös ketterän kehityksen perusperiaatteiden mukaisesti, tuoteomistaja on tuotekehitystiimille elintärkeä resurssi jonka tulee olla aina tavoitettavissa.

Tavoitettavuuteen hankaluuteen voi vaikuttaa useakin tekijä, mutta yhdeksi yleisimmistä Auer esittelee, että resursseja varataan useaan saman aikaiseen projektiin. Usean yhtäaikaisen projektin ongelma nousi esille myös haastattelututkimuksen kautta (Liite 3).

Kolmantena asiana Auer esittää läpimenoajan lyhentämiseen tarpeettoman monimutkaisuuden karsimisen, millä on usein vaikutusta myös asiakastyytyväisyyteen. Monimutkaisuudella voidaan tarkoittaa niin tuotteen ominaisuuksien monimutkaisuutta, kuin

projektin läpiviennin monimutkaisuutta. Monimutkaisuus vaatii aina enemmän suunnittelua, sekä enemmän aikaa toteutukselle ja laadun varmistukselle. Tarpeeton monimutkaisuus ei myöskään tuota asiakkaalle mitään arvoa, vaan päinvastoin, hankaloittaa projektin etenemistä ja tuotteen käyttämistä.

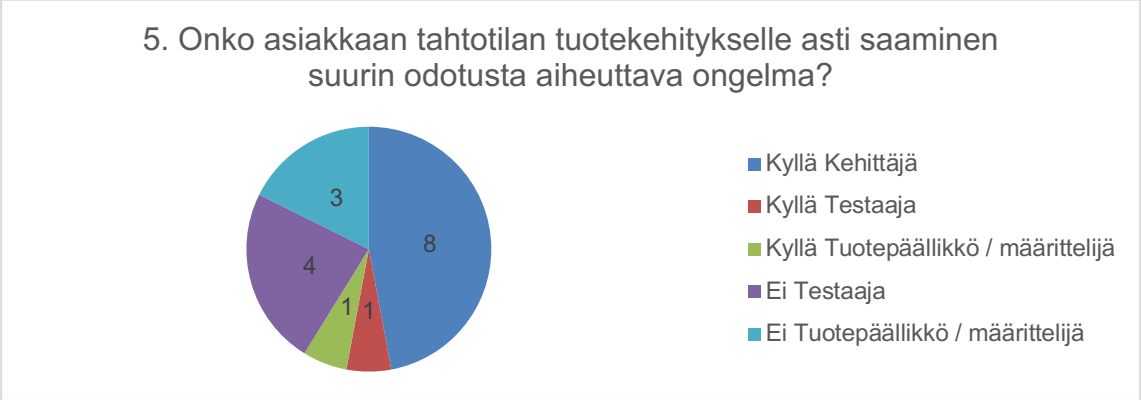
6.1 Odottelu hävikkinä

Suurin tuen tarve tuotekehitysprosessissa kohdistuu tuotekehitysprojektin viimeistelyvaiheeseen, jossa voidaan todeta liitteiden 1 ja 3 tarjoamien perusteluiden, taulukoiden ja kuvaajien perusteella olevan paljon päätöksiin ja määrittelyiden selvennykseen liittyvää odottelua. Tutkimuksen mukaan (liite 3), jotta viimeistelyvaihe saataisiin nopeammaksi, määrittelyiden laadun tulisi olla paremmalla tasolla: Muutokset ja ominaisuudet tulee dokumentoida siten, että niin määrittelijät, koodaajat, testaajat kuin julkaisijatkin ymmärtävät tuotteen asetukset ja ominaisuuden, myös tulevaisuudessa. Vesiputousmallia suorittavassa tuotekehitysprosessissa määrittelyt eivät saisi enää toteutus vaiheessa muuttua. Mikäli tuotekehitysprosessiin halutaan lisätä ketteryyttä, määrittelyt saavat olla kevyemmät ja elävämmät, mutta tuotepäälliköiden tulisi tukea enemmän tuotekehitysprosessia. Riippumatta kehitysmallista, jos määrittelyt eivät ole riittävän selkeitä että ne ymmärrettäisiin, tuotepäällikön tulisi olla saatavilla ohjaamaan ja määrittelemään ominaisuutta uudelleen. Kaikki odotus ja epävarmuus ovat hävikkiä.

Liitteen 1 taulukot tuovat esille tuotekehitysprosessin odotusajat. Tuotekehityssprintti on aikarajoitettu jakso tekemistä, jonka päättyessä tehtäviä voi vielä jäädä tekemättömäksi. Nämä ns. roikkuvat hännät voivat johtua siitä, ettei tuotekehitystiimi koe työtehtäviä tärkeiksi tai siitä, että tiimissä ei ole tarpeeksi ymmärrystä tai erityisosaamista tehtävien suorittamiseen. Tehdyn haastattelun perusteella tuotekehitykseen liittyvät hännät koettiin johtuvan useimmiten epäselvästä tai puutteellisesta tehtäväkuvauksesta tai siitä, että tehtävä odottaa toisen ryhmän toimijaa (Liite 3). Sprintin aikana tekemättä jääneet tehtävät tulee siirtää uudelle sprintille mikäli ne koetaan lisäarvoa tuottaviksi, mutta muussa tapauksessa ne tulisi poistaa, jottei ne ole tiellä tehtävälistassa ja aiheuta lisää hävikkiä. Joka tapauksessa, tekemättä jääneet tehtävät tulee käsitellä uudelleen tuotteen tuotekehitysjonoa läpikäytäessä. Päätös tehtävien tärkeydestä tulisi tehdä tuotekehitystiimin ja tuotepäällikön yhdessä, mikä edellyttää paljon yhteistyötä. (Auer 2013, 79)

Haastattelulomakkeen (liite 2; liite 3) kysymyksellä 5 ”Onko asiakkaan tahtotilan tuotekehitykselle asti saaminen suurin odotusta aiheuttava ongelma” pyrittiin hakemaan vahvistusta olettamalle suurimmasta odottelua aiheuttavasta tekijästä. Kaikki kehittäjät vastasivat kysymykseen ”kyllä” (kuvaaja 4), mikä osoittaa selkeästi kehittäjien turhautumisen määrittelyiden ja tehtävien kuvauksen laatuun ja luotettavuuteen. Samaan kysymykseen kuitenkin vain 1 tuotepäällikkö (4 vastanneesta) oli kehittäjien kanssa monivalintakysymyksen pohjalta samaa mieltä. Kysymys 6 oli vapaan sanan perustelu kysymyksen 5 vastaukselle ”Ei”. Yhden tuotepäällikön perustelusta ”Asiakas ei itsekään tiedä mitä haluaa, ennen kuin saa valmiin, katselmoidun tuotteen käyttöönsä. Asiakkaiden silmät aukeavat ns. liian myöhään tuotekehitysprojektin prosessiin nähden.”, voidaan kuitenkin vetää myös sama johtopäätös, että asiakkaan tahtotilaa ei saada vietyä tuotekehitykselle asti ymmärrettävässä muodossa. On normaalia, että asiakas ei ymmärrä mitä haluaa. Kun näkee valmiin toteutuksen, on paljon helpompaa ilmaista kehityskohteet. Tästä syystä, määrittelyssä kannattaisi keskittyä enemmän tarpeiden ja vaatimusten kuvaamiseen, sekä pyrkiä hakemaan loppukäyttäjältä palautetta useammalla tuotekatselmoinnilla. Toinen tuotepäällikkö yhtyi 3 testaajan kanssa samaan mielipiteeseen siitä, että suurin odotusta aiheuttava tekijä on, että tuotekehitysprosessissa eri rooleissa toimivat toimijat ohjataan toisten projektien pariin. Tuotepäälliköt ja testaajat priorisoidaan muihin tehtäviin, vaikka heitä tarvittaisiin tukena tuotekehitysprosessissa. Kyselyn pohjalta voidaan päätellä, että kaksi suurinta syytä odotukselle tuotekehitysprosessin aikana ovat:

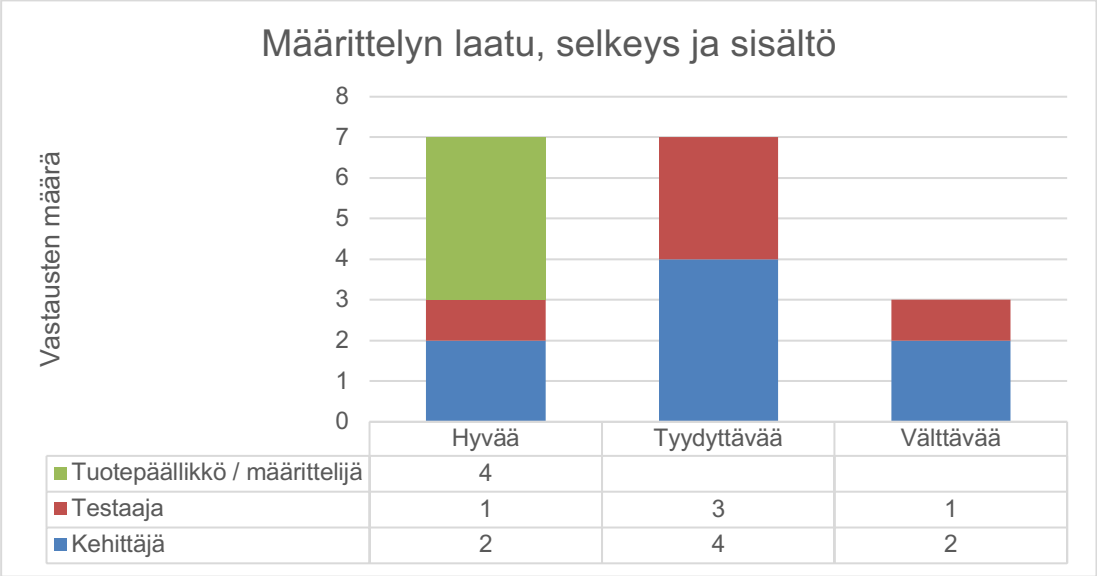
1. Asiakkaan tahtotilan tuotekehitykselle asti saaminen
2. Tuotekehitystiimiä auttavien eri roolien prioriteetit eivät kohtaa tuotekehityksen prioriteettien kanssa.



Kuvaaja 4. Haastattelulomakkeen kysymyksen 5 ”Onko asiakkaan tahtotilan tuotekehitykselle asti saaminen suurin odotusta aiheuttava ongelma?” vastaukset rooleittain.

6.2 Epäselvät määrittelyt ja tehtäväkuvaukset

Tuotteen määrittelyllä tarkoitetaan määrittelyprosessissa tehtävän toiminnallisen määrittelyn sisältöä, tuotteen kehitysjonossa olevien tehtävien kuvauksia, sekä katselmointimuutoksia kuvaavia dokumentteja. Haastattelun tuloksista koostettu kuvaaja 5 osoittaa tuotteen määrittelyn (tai määrittelyiden) olevan useimmiten hyvällä tai ainakin tyydyttävällä tasolla. Haastattelulomakkeen kysymykseen 14 (Liite 3) annettujen perustelevien vastausten perusteella voidaan todeta, että tuotekehitysprosessissa toimivat kehittäjät kokevat kehitettävän tuotteen määrittelyn olevan keskimäärin hyvällä tasolla, mutta paikoittaiset aukot tai epäselväksi jäävät asiat määrittelyssä, aiheuttavat turhautumista tekemisessä sekä viivästymistä prosessin kulussa.



Kuvaaja 5. Haastattelulomakkeen kysymyksen 13 määrittelyn laadusta annetut vastaukset rooleittain lajiteltuina.

Tuotteen määrittelydokumentin koetaan myös usein jäävän jälkeen eri palavereissa (sprintin tapahtumat, asiakaskatselmoinnit, jne.) sovituista muutoksista. Määrittelyiden raahatessa perässä, on epäselvää, mitä dokumenttia tai määrittelyä kehitettäessä tulisi noudattaa. Mikäli asioita ei dokumentoida heti, ne unohtuvat ja jäävät usein kokonaan dokumentoimatta. Tällöin, vaikka esim. kehittäjä olisikin ominaisuuden toteuttanut annettujen määrittelyiden mukaisesti, testaajalle ei kuitenkaan selviä, mitä määrittelyä vasten ominaisuus tulisi testata. Epäselvyydet luovat tarvetta kysymyksille, joihin koetaan kuitenkin olevan hankalaa saada vastausta. Tällöin muodostuu jälleen odottelua.

6.3 Tuotekehitysprosessin epäketteryys

Toimeksiantajaorganisaatio pyrkii tuotekehitysprosessissaan olemaan ketterä, sillä määrittelyille halutaan jättää liikkumavaraa. Haastattelututkimuksen mukaan (liite 3), tuotekehitysprosessi kuitenkin kärsii merkittävästä kankeudesta. Tuotekehitysprosessia edeltävä määrittelyprosessi ja sitä seuraavat testaus- ja käyttöönottoprosessi ovat perinteisen vesiputousmallin mukaisia, irrallaan tuotekehitysprosessista. Määrittelyt ovat kuitenkin riittämättömät valmiin julkaistavan tuoteversion tekemiseen ilman tuoteomistajan jatkuvaa tukea. Tuoteomistaja puolestaan on hankalasti tavoitettavissa. Kun tuote on tuotekehityksen osalta valmis, sitä ei saada julkaistua, vaan jäädään odottamaan hyväksyttyä testiraporttia, tuotantoonsiirtopalaveria ja varsinaista käyttöönottoa. Prosessit eivät toimi ketterästi yhteen, mikä saattaa johtua suurimmaksi osaksi siitä, että organisaatio työskentelee liian monen projektin kanssa saman aikaisesti. Muita ongelmia tuotekehitysprosessissa voidaan todeta olevan mm. resurssipula, kommunikaatioongelmat, sekä ehkä tärkeimpänä, ketterän kehityksen kulttuurin puute. Jotta prosessi olisi oikeasti ketterä, sen tulisi olla kokonaisuudessaan iteratiivinen ja koko organisaation tavoitteena pitäisi olla jatkuva parantaminen, kuten luvussa 3 ja taulukossa 2 esitettiin. Organisaation nykyinen tuotekehitysprosessi voidaan sanoa olevan ns. hybridimalli eli sekoitus perinteistä vesiputousmallia ja Scrumia. Hybridimalliutta voidaan puolustaa mm. tuotestrategian (roadmap) asettamilla aikatauluilla, asiakkaan hankalalla tavoitettavuudella, sekä ketterän kehityksen kulttuurin puutteella. Taulukon 2 esittelemissä 12 ketterän kehityksen kohtalokkaassa virheessä, ns. omakutoiset prosessit listataan yhdeksi yleisimmistä syistä, jonka vuoksi ketterät kehitys usein epäonnistuu. VersionOnen tutkimuksen (2018) mukaan jopa 34% tutkimukseen vastannaista esitti epäjohdonmukaiset prosessit haasteeksi ketterässä kehittämisessä. 53% vastannaista vastasi syyksi ketterän kehityksen epäonnistumiselle organisaation laajuisen ketterän kehittämisen kulttuurin puutteen ja 46% vastasi yhdeksi epäonnistumisen syyksi muutoshalukkuuden puuttumisen.

Taulukon 2 esittämistä ketterän kehityksen 12 kohtalokkaasta virheestä, toimeksiantajan tuotekehitysprosessissa voidaan todeta esiintyvän ainakin seuraavat 8 ongelmaa:

- Omakutoinen tuotekehitysprosessi, sillä prosessissa sovelletaan omaa hybridimallia.

- Ketterien kehitystapojen parhaiden käytäntöjen laiminlyönti, mm. siksi, että asiakas ei osallistu tuotekehitysprosessiin, eikä Scrumia päästä suorittamaan oikein.
- Kommunikaation ja itseorganisoitumisen puuttuminen, sillä scrumtiimi ei ole täysin itseorganisoituva ja kommunikaatio on todettu olevan hankalaa.
- Väärä valinta avainpersoonaksi, koska tuotekehitystiimin vetäjä joutuu toimimaan scrummasterina ja tekemään myös paljon tuoteomistajan työtä.
- Tuoteomistajan osallistumisen puute.
- Epäonnistuneet aikatauluarviot, kuten mm. liite 1 osoittaa.
- Julkaisun viivästymiset, kuten mm. liite 1 osoittaa.
- Epäselvät määritelmät valmiista, sillä organisaatiolle jää usein epäselväksi, mitä on toteutettu ja onko se toteutettu oikein.

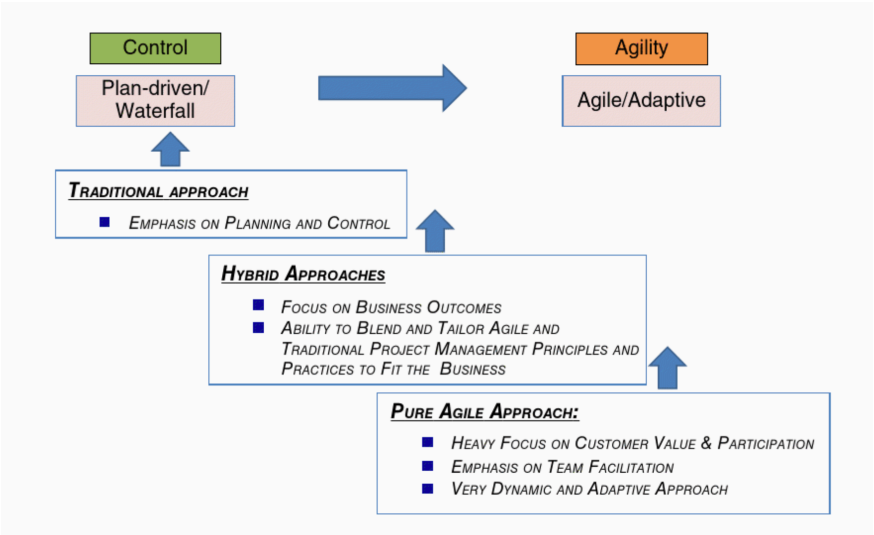
6.3.1 Hybridimalli

Vaikka tuotekehitys tehtäisiinkin yrityksessä Scrumin mukaisesti, yrityksen muut osastot saattavat karttaa Scrumia ja tuntevat työskentelyn kotoisammaksi vesiputousmallilla. Yhdistettäessä nämä kaksi eri mallia, voidaan yrittää käyttöön ottaa parhaat käytännöt molemmista malleista. Yhdistelmää kutsutaan usein esim. nimillä agilefall, wagile, water-agile-fall. Perinteiseen kehitykseen verrattuna, käyttöönotettaessa Scrumia osaksi vesiputousmallia, ei käytännössä menetetä mitään. Scrumissa raportoidaan edistymistä yleisesti vesiputousmallia useammin, joten projektin seurattavuus paranee. Tuotteen laadussakaan ei hävitä, sillä Scrumin päivittäiset palaverit ja jatkuva testaus pitävät laatua yllä perinteisiä vesiputousmallin tapoja aggressiivisemmin. Myös budjetti pysyy paremmin hallinnassa päivittäisen seurannan vuoksi. Verratessa Scrumin lähestymistapaa perinteiseen vesiputousmalliin, Scrum on oikeastaan vain erittäin lyhyt ja toistuva vesiputousprosessi. Lyhyessä kehitysjaksossa on järkeenkäyvät edut riskien-, aikataulujen-, sekä budjetinhallintaan liittyen. (Pries & Quigley 2011; Davis 2012)

Vaikka hybridimalli perinteisen ja ketterän kehityksen välillä onkin yleisimpiä syitä ketterän kehityksen epäonnistumiseen (luku 3), jossain ympäristöissä se on kuitenkin välttämätöntä (Cobb 2015, 123; Davis 2012). Vesiputousmalli ei salli muutoksia kesken kehityksen ja ketterä kehitys kärsii tarkkaan ennalta asetetusta aikataulusta ja lukituista ominaisuuksien määrittelystä. Kuva 13 on Charles Cobbin kuvaus hybridimallista perinteisen vesiputousmallin ja ketterän kehityksen välissä, teoksesta The Project Mana-

ger's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach. Perinteisessä kehityksessä painotetaan suunnitelmallisuutta ja kontrollia, kun ketterässä kehityksessä keskitytään tuottamaan asiakkaalle arvoa ja sitouttamaan asiakas mukaan projektiin. Yhdistelmämallissa painopiste on liiketoiminnan tuloksessa ja sitä käytettäessä tarvitaan kykyä sekoittaa ja räätälöidä ketterää kehitystä perinteisen projektinhallinnan periaatteisiin ja käytäntöihin, jotta ne sopivat liiketoiminnan kanssa yhteen. Päämääränä hybridimallissa on lisätä toistuvuutta ja mukautuvuutta perinteiseen malliin, ottamatta käyttöön kaikkia ketterän kehityksen lakeja. Tuloksena pyritään saavuttamaan perinteistä pienempi riskiprofiili, mutta ketterää kehitystä tavanomaisempaa suunnitelmallisuutta. Cobbin mukaan hybridimalli voi olla perinteistä mallia parempi ratkaisu mm. kun

1. Sopimusskenaariot edellyttävät sitoutumista jonkinlaiseen kustannus- ja toimitusaikatauluun, mutta vaativat vielä jonkin verran joustavuutta vaatimusten määrittämisessä.
2. Työskennellään aloilla kuten lääketeollisuus, jossa saatetaan joutua osoittamaan vaatimustenmukaisuuden jäljitettävyyttä ja riittävää valvontaa testaus- ja vapautusprosesseissa, jotta varmistutaan siltä, että prosessi noudattaa kaikkia sääntelyä koskevia vaatimuksia.
3. Työskennellään hankkeissa jotka on skaalattu suurempiin ja monimutkaisempiin yritystasoisiin hankkeisiin, jolloin tarvitaan myös huomattava määrä projektinhallintaa.



Kuva 13. Charles Cobbin kuvaus hybridistä agile-kehityksestä.

Haastattelututkimuksen (liite 3) mukaan hybridimalli ei ole nykyisillä säännöillä ja periaatteilla toimiva ratkaisu toimeksiantajan tuotekehitysprosessiin. Tuotekehitysryhmän mukaan nykyisen hybridimallin menetelmät eivät ole tarpeeksi byrokraattisia, jolloin liikaa suunnittelemattomia muutoksia sallitaan tuotekehitysprosessin tehtäväksi, tilanteessa jossa kokonaisaikataulu kehitykselle on tarkasti rajattu. Syntyy suunnittelematonta kiirettä jo alun alkaenkin tiukkoihin aikatauluihin. Kehittäjät kokevat että tuoteomistajilla ei ole tarpeeksi aikaa tai arvostusta tuotekehitysprosessin aikaisille palavereille, eikä heillä ole myöskään aikaa reagoida kehitystiimin esittämiin huomioihin tai kysymyksiin, jolloin muutospyynnöt ja tarkennukset määrittelyihin tulevat liian myöhään. Määrittelyprosessin tuottaman määrittelydokumentin laatu ennen tuotekehitysprosessia ei ole riittävä tuotekehitysprosessin loppuunsaattamiseksi myöskään tuotepäälliköiden mielestä.

Hybridiratkaisu tuo haasteita projektipäällikölle, sillä hänen tulee saman aikaisesti ymmärtää sekä perinteistä vesiputousmallia, että ketterää kehitysmallia. Lisäksi hänen tulee osata yhdistää molempien mallien arvoja juuri sopivasti yhteen. Toimeksiantajan prosesseista projektipäällikkö puuttuu kokonaan ja tehtävää hoidetaan hajautetusti, mikä vaikeuttaa asiaa merkittävästi. Hybridiratkaisulla ei myöskään voida odottaa saavutettavan ketterän kehityksen kaikkia etuja. Mikäli toiminta ei ole täysin ketterää, ei voida myöskään saavuttaa kaikkia ketterän kehityksen hyötyjä. Mikäli määrittelyt ovat muuttuvia, hybridimalli todennäköisesti parantaa tuloksia perinteiseen kehitykseen nähden, lieventäen riskejä ja nostaen projektin tempoa. Lisäksi se auttaa organisaation kulttuurin muutoksessa lähemmäksi ketterää kehitystä. Hybridimalli vaatii toimiakseen jatkuvaa hyvää kommunikaatiota, säännöllisiä tapaamisia ja jatkuvaa tietoista projektinhallintaa. Vahvaa projektinhallintaa vaaditaan, jotta projekti pysyisi raiteillaan. Vesiputousmallin mukaan työskentelevät ryhmät olisi hyvä saada mukaan säännöllisiin palavereihin joita ketterätiimi harjoittaa, jotta informaatio muuttuvista määritelmistä ja toteutetuista ratkaisuista saadaan jaettua tehokkaasti. Hybridimallissa suositellaan pidettäväksi myös ketterälle kehitykselle ylimääräisiä isompia tapaamisia perinteisten tiimien kanssa n. puolivälissä projektia, jotta päämäärät ja suunnitelmat saadaan synkronoitua tiimien välillä. (Cobb 2015; Davis 2012; Pries & Quigley 2011)

6.3.2 Tuoteomistajan puuttuminen

Tuoteomistajan ensisijaisiin tehtäviin tuotekehitysprosessissa ketterän kehityksen periaatteiden mukaan kuuluu mm. tehtävälistasta, määrittelyistä ja riskien analysoinnista vastaaminen, sekä tuotekehitystiimin tukeminen. Asiakas eli tuotteen loppukäyttäjä, on paras kertomaan, mitä tuotteelta halutaan. Tutkimuksen (Liite 3) mukaan, asiakkaalla ei kuitenkaan ole tarpeeksi aikaa omistautua tuotekehitykselle, joten asiakasta edustaa tuotteen määrittelijä (eli tuotepäällikkö) ja yrityksen oma projektijohto, sekä muut asiakasrajapinnassa työskentelevät. Asiakasrajapinnan näkökulmasta kehitysmenetelmät eivät ole tarpeeksi joustavia. Muutoksien tulisi tapahtua nopeammin, julkaisun tulisi olla useammin tehtävää ja määrittelyt ovat jatkuvassa muutospaineessa. Kaikki viittaavat siis siihen, että tuotekehitysprosessin tulisi olla ketterän kehityksen mallin mukainen. Kuten aiemmin on todettu, ketterässä tuotekehityksessä tuoteomistajan omistautuminen tuotekehitysprosessille on välttämätöntä.

Tuotekehitysprosessiin osallistuvat kehittäjät (ja kehitystiimin vetäjä ns. team lead) tuotekehitystiimistä, testaaja testaustiimistä, sekä määrittelijä määrittelytiimistä. Tuotekehitysprosessin alkaessa määrittelyprosessin päätyttyä, asiakasta ja hänen tarpeitaan ei ole päässyt kuulemaan kuin yrityksen myyjät, määrittelijät, asiakasvastaavat ja yrityksen johto. Määrittelijät ovat yrityksen ainoa taho, jotka ovat oikeasti paneutunut asiakkaan kanssa tuotteen ominaisuuksien määrittelyyn ja ongelman ratkaisuun tuotekehitysprosessin alkaessa. Näin ollen, mikäli yrityksen asiakasta eli tuotteen loppukäyttäjää ei pystytä sitouttamaan tuotekehitysprosessiin mukaan, ainoa mahdollinen tuoteomistajarooliin kelpaava taho on tuotteen määrittelijä, joka toimii tuotteelle yrityksen sisäisenä asiakkaana. Kuvaajalla 6 osoitetaan, että tuotteen tuotepäällikkö ei ole riittävästi tavoitettavissa tuotekehityksen aikana. 8 kehittäjästä 5 on sitä mieltä, että tuotepäälliköltä ei saada tarpeeksi tukea ja tietoa tuotekehityksessä tehtävään työhön.

Kuvaaja 6. Tulokset rooleittain kysymyksestä 9 ”Saatko tuotepäälliköiltä riittävästi tukea ja tietoa työhösi”.

Puuttuvaa tuoteomistajan roolia on tuotekehitysprosessissa pyrkinyt paikkaamaan tuotekehitystiimin vetäjä, joka toimii tuotekehitysprosessissa myös koodaukseen ja kehitykseen osallistuvana scrummasterina. Tuotekehitystiimin vetäjä on auttanut tuoteomistajaa tuotteen kehitysjonon hoitamisessa aina sen luonnista tehtävien sulkemiseen asti. Tiimin vetäjän odotetaan siis tekevän kolmoisroolin, jossa edellytetään tiimin johtamisen lisäksi tietoa ja taitoa ohjelmointiin sekä myös tuotteen ominaisuuksien kuvaamiseen, sekä päätöksiä kehitysjonon työstämiseen liittyen. Tiimin vetäjä ei kuitenkaan kehitystiimin jäsenien tavoin ole koskaan nähnyt tuotteen loppukäyttäjää, vaan on täysin määrittelijän kirjoittamien määrittelyiden ja projektista pidetyn kick-offin armoilla. Tiimin vetäjä ei tiedä, mikä tuotteessa on oikeasti tärkeää ja mikä ei tai mikä asia tuotteessa tuottaa eniten arvio asiakkaalle. Parhain tieto yrityksen sisällä asiasta on tuotteen määrittelijällä. Scrummaster ei myöskään ole lähtökohtaisesti persoonallisuudeltaan projektipäällikön kaltainen ihminen, sillä projektipäällikkö tulee kulttuurista jossa tekemistä kontrolloidaan tarkasti ja noudatetaan ennalta tehtyjä määrittelyitä, mikä sotii Scrumin ja ketterän kehityksen kulttuuria vastaan. Scrummaster on enemmänkin ohjaaja, kuin johtaja (Hughes 2013, 46). Mike Cohn kirjoittaa että perinteisesti ajattelevat tuoteomistajat saattavat olla sitä mieltä, että ei ole merkitystä sillä, missä järjestyksessä ominaisuudet valmistuvat, sillä kaikki ominaisuudet ovat tärkeitä ja niiden tulee olla tehtynä, jotta tuote olisi valmis. Cohnin esittämää väitettä tukee myös haastattelun (liite 3) perusteluiden joukosta löytyvä yksittäinen perustelu, jossa esitetään samanlainen

tuoteomistajan ilmaisema mielipide. Tällöin tehtävien priorisointi ja paloittelu jää tuotekehitystiimin tehtäväksi, mikä voi johtaa tilanteeseen, jossa aikataulun tullessa täyteen, arvokkaita ominaisuuksia on tiputettu kehityksestä pois ja vähemmän arvokkaita on toteutettu. (Cohn 2016, 16-17; Pries & Quigley 2011, 21)

6.3.3 Sprinttipalaverien tärkeyden ymmärtämättömyys

Koska Scrumissa pyritään priorisoimaan tekemistä perustuen asiakkaan antamaan palautteeseen jäykän, virallisen ja paljon aikaa vievän dokumentaation sijasta, asiakas (tai loppukäyttäjä) kannattaisi kutsua kaikkiin tuotekehityksen tapahtumiin: Projektin, tuotteen ja prosessin katselmointiin, sprinttikatselmointiin, tuotekehitysjonon katselmointiin, sekä päivittäisiin palavereihin. Katselmoinnit ovat yleensä hyviä käytäntöjä, koska ne tarjoavat mahdollisuuden osapuolten väliseen viestintään. Asiakas saattaa kokea usein pidettävät palaverit liiallisiksi tai turhiksi, joten asiakkaan kutsumista katselmointiin kannattaa kuitenkin harkita tarkkaan. Asiakkaan tahdosta hyvin perillä oleva sisäinen tuoteomistaja osaa usein ohjata kehitystä jopa asiakasta tehokkaammin. (Pries & Quigley 2011, 21)

Scrumin tapahtumiin tulisi kaikkien osallistujien valmistautua etukäteen, jotta aikarajattu palaveri olisi mahdollisimman tehokas ja säilyisi kaikkien osallistujien osalta mielenkiintoisena tapahtuman alusta sen loppuun. Esim. sprintin aloituspalaveriin mennessä, tuotteen kehitysjonon käyttäjätarinat ja tehtävät tulisi jo olla työstettynä, karsittuna ja priorisoituna. Tuoteomistajalla tulisi olla jo alustava suunnitelma siitä, mitä sprintin aikana voitaisiin tehdä. Tällöin aloituspalaverissa sovittavaksi jää lähinnä enää sprintin suunnitelma, resurssien läpikäynti, sekä tehtävälistan lukkoon lyöminen.

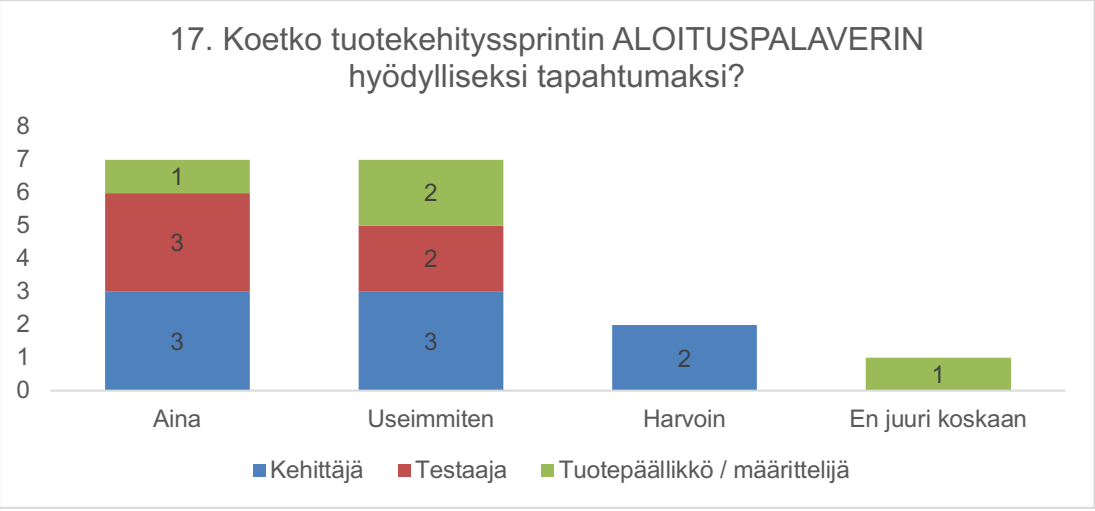
Tuotekehityssprintin aloituspalaveri on erittäin tärkeä osa Scrumia. Aloituspalaveri pidetään jokaisen sprintin alkajaisiksi. Sillä on kaksi tärkeää päämäärää:

1. Tuotekehitystiimi ja tuoteomistaja yhdessä päättävät, mitä käyttäjätarinat tai tehtävät otetaan tuotekehityssprinttiin mukaan.
    a. Tuoteomistaja tarkistaa että tehtävälistan tehtävät ovat prioriteettijärjestyksessä ja priorisoitu oikein. Tuoteomistajan tehtävänä on miettiä alustava sprintin tehtävälistasuunnitelma.
    b. Tuotekehitystiimi arvioivat, kuinka monta tehtävää he ehtivät saada sprintin aikana valmiiksi sprintin aikaisella kokoonpanolla.

2. Tiimi määrittelee riippuvuudet siitä, mitä tehtäviä tulee tehdä, jotta valittujen käyttäjätarinoiden ja suunnitelmien työstäminen olisi mahdollista. Tiimi tekee alustavasuunnitelman siitä, miten tehtävät sprintin aikana jaetaan tiimin jäsenten kesken. Tuoteomistajan ei yleensä tarvitse osallistua tähän osioon. Vaikka tuotteen kehitysjonon tulisi olla tässä kohtaa jo priorisoitu, yleensä joudutaan vielä tekemään viimeiset muutokset tuotekehitystiimin ja tuoteomistajan yhteistyössä. (Cobb 2015)

Haastatteluvastausten (liite 3) valossa, tuotekehityssprintin aloituspalaveri koettiin aina tai useimmiten hyödylliseksi tapahtumaksi. Kuvaaja 7 osoittaa vastausten hajonnan rooleittain. ”Harvoin” vastauksia perusteltiin mm. sillä että sprintin ohjelmaa ei ole mietitty etukäteen ennen palaveria riittävän hyvin, eikä palaverissa välttämättä saada päätöstä täysin lukitusta tehtävälistasta. Tuoteomistajan antamaa yksinäistä ”En juuri koskaan” -vastausta perusteltiin sillä, että tuoteomistajalle ei ole tärkeää tehtävien tekojärjestys, mikä selvästi kielii tehtyjen määritelmien joustamattomuudesta, mutta myös siitä että lukitun tehtävälistan tärkeyttä tai usein saatavan palautteen tärkeyttä ei ole ymmärretty.

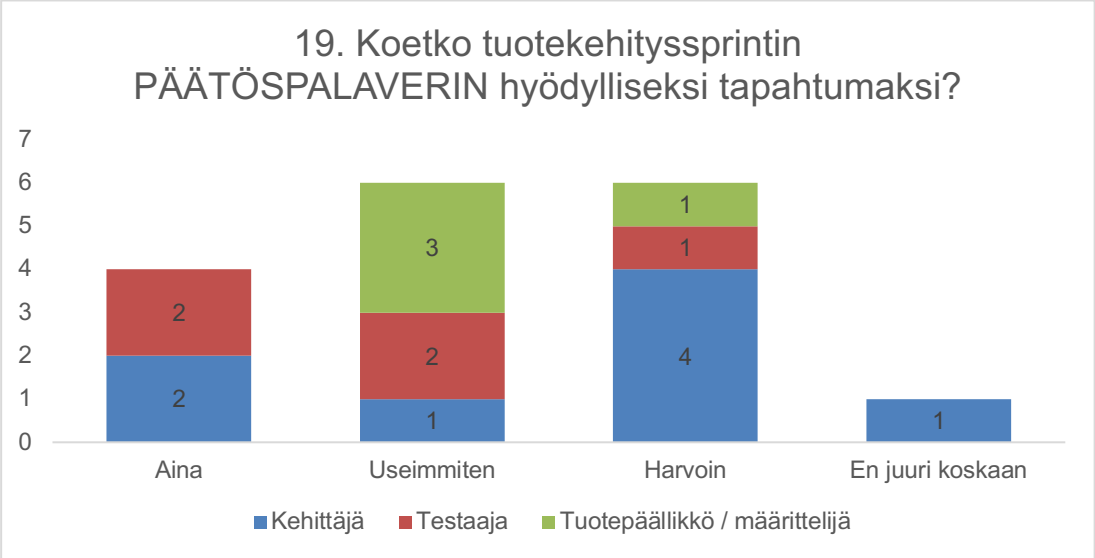
Aloituspalaverin jälkeen tuotekehitystiimi on sitoutunut tekemään sovitun tehtävälistan mukaiset tehtävät, mutta ei muuta. Mikäli tehtävälista muuttuu, ei myöskään lupaus voi enää pitää, mikä on erityisen ongelmallista sprinteissä joissa tiimi työstää useampaa projektia usean tuotepäällikön kanssa ja mahdollisesti vielä usealle asiakkaalle. Muutos sprintin ohjelmaan vaikuttaa lupauksiin kaikkien projektien edistämisestä, sillä tuotekehityssprintti on lyhyt ja tiivis paketti yhdessä sovittuja tehtäviä.



Kuvaaja 7. Kokemukset aloituspalaverin hyödyllisyydestä rooleittain.

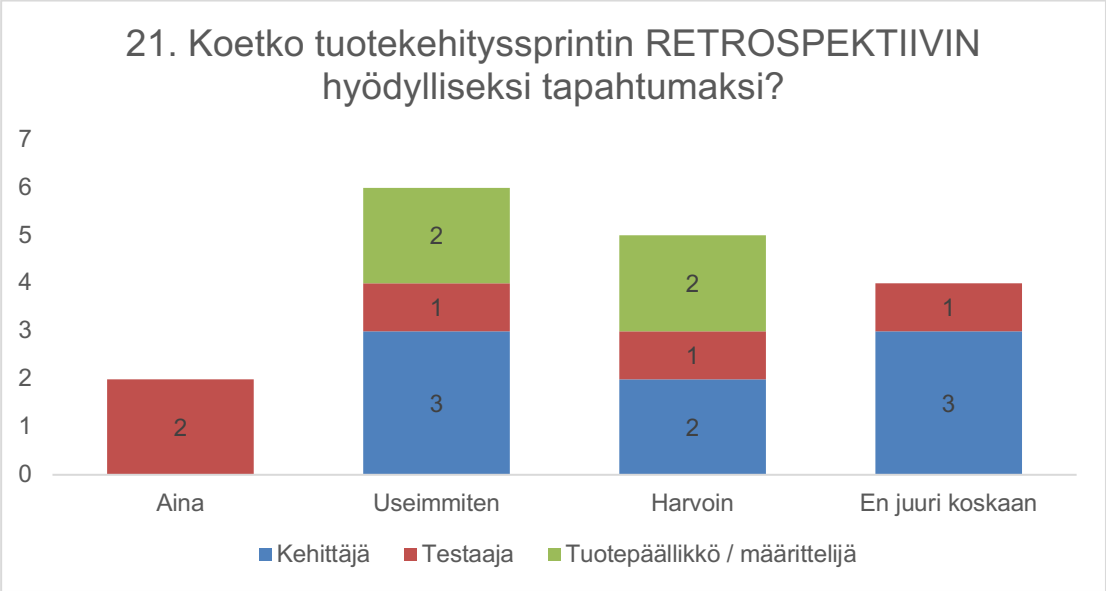
Kokemuksissa lopetuspalaverin hyödyllisyydestä on aloituspalaveria paljon suurempaa hajontaa (kuvaaja 8). Annettujen perusteluiden pohjalta (liite 3) ongelmaksi ei välttämättä muodostu se etteikö palaveria oikeasti koettaisi tärkeäksi, vaan se että paikalle ei saada tarvittavaa väkeä. Tapahtuman katselmoinnit eli ns. demot tehdystä työstä koetaan yleisesti hyviksi.

Päätöspalaverin ideana on esitellä tehtyä tuotetta, saada palautetta tehdystä työstä, sekä tuoda esille kohdatut vastoinkäymiset ja kompromissit. Jotta päätöspalaveri olisi jouhevampi, on hyvä käytäntö pyytää hyväksyntää tuoteomistajalta tehdyille ominaisuuksille jo hyvissä ajoin ennen sprintin päätöstä. (Cobb 2015)



Kuvaaja 8. Kokemukset aloituspalaverin hyödyllisyydestä rooleittain.

Sprintin retrospektiivissä on mahdollista kehittää toimintaa ja keskustella sprintin aikana opituista virheistä, sekä sopia yhdessä, mitä seuraavassa sprintissä kannattaisi toisin. Tapahtuma on tärkeä jatkuvan kehittymisen kannalta. Kuvaajan 9 ja annettujen perusteluiden pohjalta (liite 3) retrospektiivin ideaa ei ole organisaatiossa ymmärretty. Osa vastaajista sanoo ettei osallistu kyseiseen tapahtumaan koskaan ja joku myöntää ettei tiedä mikä kyseinen tapahtuma edes on. Vastaukset ovat huolestuttavia, mutta toisaalta ymmärrettäviä, sillä se tukee väitettä siitä, että edellisten kehitysjaksojen virheistä ei opita, vaan virheet toistetaan kerta toisensa jälkeen. Mikäli kehittämisestä ei keskustella sille sovitussa tapaamisessa, niin missä sitten? Annetuissa vastausten perusteluissa todetaankin, että tapahtuma olisi erittäin hyödyllinen, mikäli siihen saataisiin osallistumaan oikeat ihmiset.



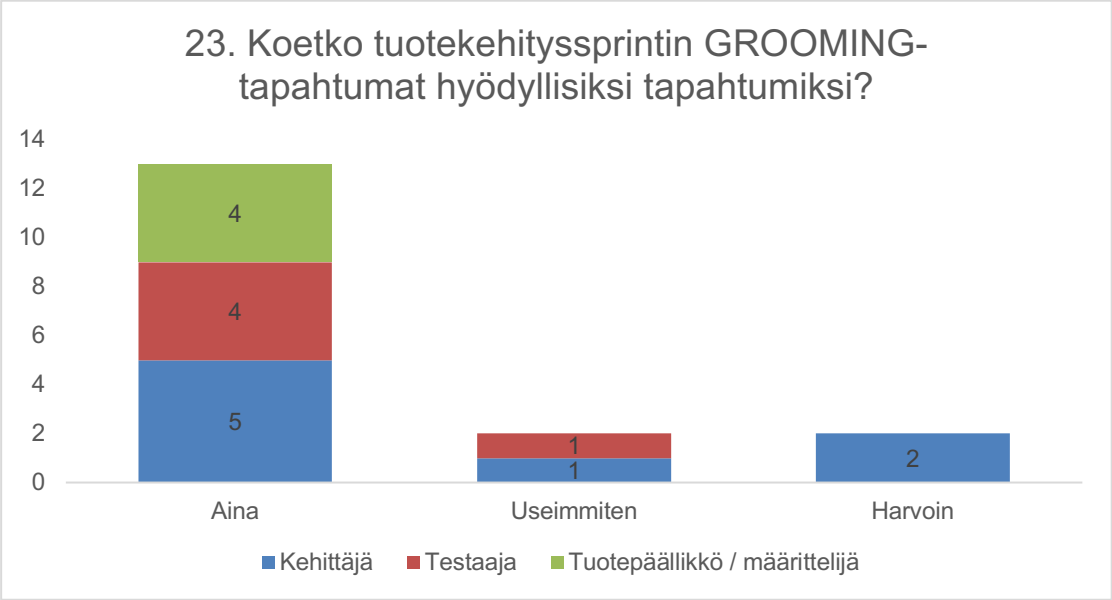
Kuvaaja 9. Kokemukset aloituspalaverin hyödyllisyydestä rooleittain.

Kehitysjonon työstäminen eli groomaus (backlog grooming) on yksi Scrumin tärkeimpiä toiminteita. Tuotteen kehitysjonon työstämisen yhteydessä kehitystiimi ja tuoteomistaja käyvät yhdessä läpi tuotteen kehitysjonoa ja siihen liittyviä epäselvyyksiä. Työstön lopputuloksena on tehty seuraavia toimenpiteitä:

- Ylimääräiset käyttäjätarinat tai tehtäväkuvaukset poistetaan kehitysjonosta.
- Luodaan uusia käyttäjätarinoita tai tehtäväkuvauksia uusista tarpeista.
- Asetetaan käyttäjätarinoille asianmukaiset prioriteetit.
- Asetetaan käyttäjätarinoille työmääräarviot.
- Korjataan aikatauluarvioita uuden tiedon valossa.
- Pilkotaan tehtäviä pienempiin kokonaisuuksiin. (Cobb 2015)

Vastuu kehitysjonon sisällöstä on viime kädessä tuoteomistajalla. Tuotteen ajantasainen kehitysjono mahdollistaa suoraviivaisen kehittämisen vähemmällä määrällä selvitettäviä kysymyksiä. Haastattelun (liite 3) mukaan lähes kaikki kokivatkin palaverin hyödylliseksi ja iso osa vastaajista oli sitä mieltä, että kehitysjono työstämiseen pitäisi saada enemmän tuotekehitysprosessissa enemmän yhteistä aikaa. Kuvaaja 10 esittää haastattelun vastaukset rooleittain. Kehittäjistä 2 vastasivat kokevan tapahtuman harvoin hyödylliseksi. Perusteluissa mainittiin että tapahtuma on usein hankalaa järjestää ja kun se lopulta saadaan järjestymään, saattaa ajankohta olla jo liian myöhäinen ja päätös on jo jouduttu tekemään oman mielen mukaan. Toisessa ”harvoin” -vastauksessa vähäteltiin palaverin merkitystä, mutta ei selkeästi oltu ymmärretty, miksi

palaveria oikeasti pidetään. Tapahtuma on erittäin tärkeä tilaisuus, jossa voidaan käydä kaikki avoimet kysymykset läpi. Sen päättyessä tehtävät tulisi olla käytynä läpi siihen malliin, että kysymyksiä ei myöskään enää jatkossa pitäisi niihin liittyen herätä. (Cobb 2015, 41)



Kuvaaja 10. Kokemukset kehitysjonon työstön hyödyllisyydestä rooleittain.

6.4 Projektin hallinnalliset ongelmat

Projektien aikataulupaineet ja krooninen kiire ovat osoittautuneet myös organisaation aiemmissa tutkimuksissa ja haastatteluissa ongelmallisiksi. Haastattelulomakkeen kysymykseen ”3. Mikä asia vaikuttaa mielestäsi eniten tuotekehitysprojektin lopputuotoksen laatuun” vastaajista puolet mainitsivat haasteellisen aikataulun tai kiireen laatua heikentävänä tekijänä. Yrityksellä koetaan kuitenkin olevan krooninen työvoimapula, mitä on pyritty ratkomaan palkkaamalla yritykseen lisää työntekijöitä yrityksen jokaiseen prosessiin, mikä näkyy mm. yrityksen henkilöstön suuressa kasvussa. Lisäksi koetaan ongelmia projektien ennustettavuudessa ja priorisoinnissa. Näistä syistä johtuen tuotekehitystiimi ja sen sidosryhmät joutuvat tekemään useaa projektia samaan aikaisesti, eikä tuotekehityksessä oleva ensisijainen projekti saa tarpeeksi huomiota ja tukea. Organisaatiossa koetaan olevan ainainen kiire. (liite 3)

Scrumin kehittäjät Schwaber ja Sutherland ovat vuonna 2017 julkaisseet Scrum Guide -oppaan, jossa he kuvaavat myös optimaalisen koon aloittelevalle Scrum-tiimille: ”Kehi-

tystiimin optimaalinen koko on riittävän pieni, jotta se pysyy ketteränä ja riittävän suuri, jotta se saa sprintin akana valmiiksi merkittävän määrän työtä". Oppaassaan he ohjeistavat että tiimin koon ollessa vähemmän kuin kolme henkilöä, vuorovaikutus vähenee, mikä johtaa pienempiin tuottavuus hyötyihin. Liian pieni tiimi saattaa myös törmätä sprintin aikana osaamispulaan jollakin osa-alueella. Mikäli tiimin koko on yli 9 henkeä, Schwaber ja Sutherland kirjoittavat tiimin vaativan liikaa koordinointia. Suuret tiimit aiheuttavat heidän mukaansa liikaa kompleksisuutta. Toimeksiantajan tuotekehitysprosessissa työskentelevien Scrum tiimien koko on n. 4 koodaajaa, testaaja ja tuotepäällikkö. Scrummaster on yksi kehittäjistä. Tuotekehitysprosessin ulkopuolelta kehitykseen osallistuu myös integraatioiden tekijät. Tiimin koko on siis tutkimusten mukaan sopiva, eikä sitä kannata jakaa pienempiin osiin tai kasvattaa enempää, mikäli osaamisen osalta sen kasvattamiseen ei osoiteta selkeää tarvetta. Schwaber ja Sutherland mainitsevat myös että Scrum ei tunnusta kehitystiimin jäsenten titteleitä riippumatta henkilön työn sisällöstä, eikä alitiimejä, jotka vastaisivat erityisistä osa-alueista kuten testaamisesta, arkkitehtuurista, käyttöpalveluista tai liiketoiminta-analyysista. Kehitystiimin jäsenillä voi olla erityistä osaamista tai erilaisia työn painopisteitä, mutta vastuu kehityksestä kuuluu koko kehitystiimille yhdessä.

6.4.1 Useat yhtäaikaiset projektit

Se että projektin jäsenet työskentelevät usean projektin kanssa yhtäaikaisesti, voidaan pitää yhtenä suurimmista syistä projektien epäonnistumiselle. Yleinen argumentti usean projektin kehittämistä vastaan on, että projektihankkeet tuottavat yritykselle tulovirran niiden valmistumisajankohtana. Niiden viimeistely maksimoi tulot, koska useimmiten tulot vähenevät ajan myötä. Se että asiat aloitetaan, ei takaa että ne saadaan myös päätökseen. Toinen yleinen väite on, että siirtyminen tehtävien välillä katsotaan jätteeksi. Ketteränkehityksen toimintatapa ei ole syypää monen yhtäaikaiseen projektin saman aikaiseen työstämiseen, mutta ketteriä menetelmiä käyttämällä ongelma usein korostuu tai nousee esille. Ketterän kehityksen edellyttämä tiivis yhteistyö kärsii, kun henkilökohtaiset prioriteetit eivät kohtaa. (Leach 2014 25, 27; Stettina & Smit 2016; Toikkanen ym. 2013)

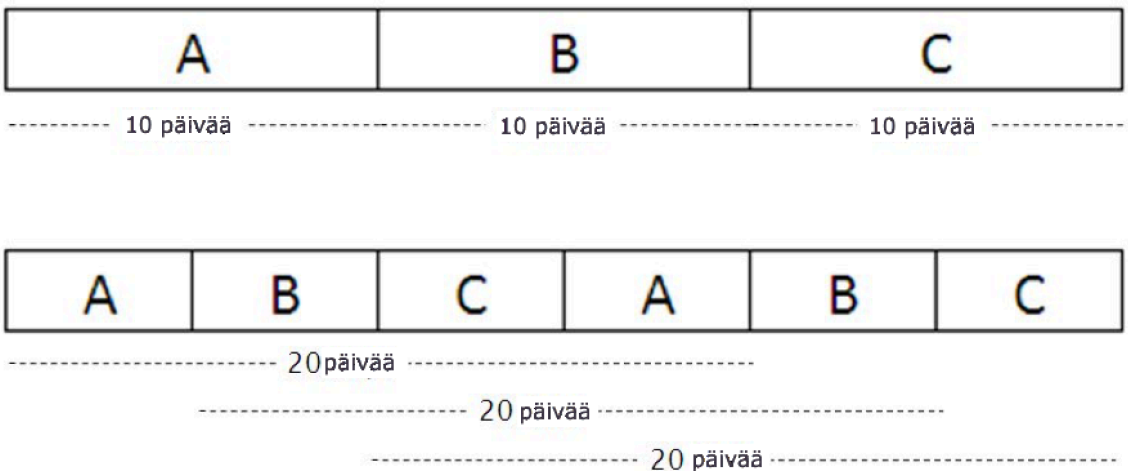
Yhtäaikaisten kehitystöiden ja projektien määrä on syytä pitää läpimenokyvyn puitteissa. Kirjallisuuden mukaan kullakin kehittäjällä ja tuoteomistajalle saa olla vain yksi työ kerrallaan käynnissä. Mikäli tuoteomistajalla on vastuullaan useita eri projekteja, kehi-

tystiimi joutuu mm. odottamaan tuoteomistajan päätöksiä. Useat yhtä aikaiset projektit vaativat myös aina enemmän kommunikointia ja projektien hallintaa. Sprintin tehtävälistalle valitut projektit riitelevät keskenään prioriteeteista ja informaatiota tulee jakaa useampaan suuntaan. Työtä tulisi jaksottaa siten, että tuotteet aidosti priorisoidaan niin, että tuotekehitysorganisaation laajuudella tehdään vain yhtä (tai maksimissaan kahta) tuotetta kerrallaan, jotta hävikki saataisiin mahdollisimman pieneksi. (Toikkanen ym. 2013, 84-85; Auer 2013, 12-13; 26 – 27; Cohn 2006)

Se että tehtäviä tai projekteja tehdään yhtä kerrallaan, on Leanin toimintatavan perusteella kaikkein tehokkainta ja johtaa teoreettisesti lyhyempiin tuotekehityssykleihin. Vaikka Scrumissa tiimin on alun perin tarkoitettu työskentelevän vain yhden projektin parissa kerrallaan, on realismia, että Scrum-tiimit joutuvat työskentelemään usean projektin parissa yhden sprintin aikana, esim. suuren asiakaskannan tai pienen tehtävälistan vuoksi. Cristoph Stettina ja Mark Smit kertovat vuonna 2016 julkaistussa julkaisussaan ”Team Portfolio Scrum: An Action Research on Multitasking in Multi-project Scrum Teams” monen yhtäaikaisen projektin vaikutuksesta tehokkaisiin työtunteihin. Monen ohjelmistoprojektin ketteräkehitys on heidän mukaansa toistaiseksi vielä uusi tutkimuksen alue, mutta tehtyjen tutkimusten mukaan tehokkaiden tuntien määrä tuotannossa saattaa kasvaa, kustannusten kasvun kustannuksella, 70%:sta 80%:in nostettaessa saman aikaisten projektien määrä kahteen. Mikäli projekteja on enemmän kuin kaksi päällekkäin, koetaan tehokkaiden tuntien määrässä tasaista laskua: kolmella samanaikaisella projektilla tehokkuus on 60%, neljällä se on tutkitusti 45% ja viidellä samanaikaisella projektilla enää 35%. Se että työntekijä tekee useaa asiaa saman aikaisesti, vaikuttaa tehokkuuden lisäksi myös työntekijän henkiseen jaksamiseen, sekä keskittymiskykyyn negatiivisesti. Häiriöt ja tehtävien vaihdot eivät maksa pelkästään aikaa. Psykologisesti ajateltuna tehtävän vaihto koostuu 3 asiasta: passiivisesta edellisen tehtävän pois sulkemisesta, uuteen tehtävään valmistumisesta ja jälkivaikutuksesta. Jälkivaikutukset voivat olla mm. vasteajan nousua tai virheiden kasvun ilmenemistä. Tehtävän vaihtaminen on vaikeampaa, mitä monimutkaisemmista tehtävistä on kyse. Lawrence Leach (2014) kirjoittaa kirjassaan ”Critical Chain Project Management” usean yhtäaikaisen tehtävän tekemisen haitoista. Hänen 400 ihmiselle teettämänsä tutkimuksen mukaan, tehtäessä vain kahta tehtävää samanaikaisesti

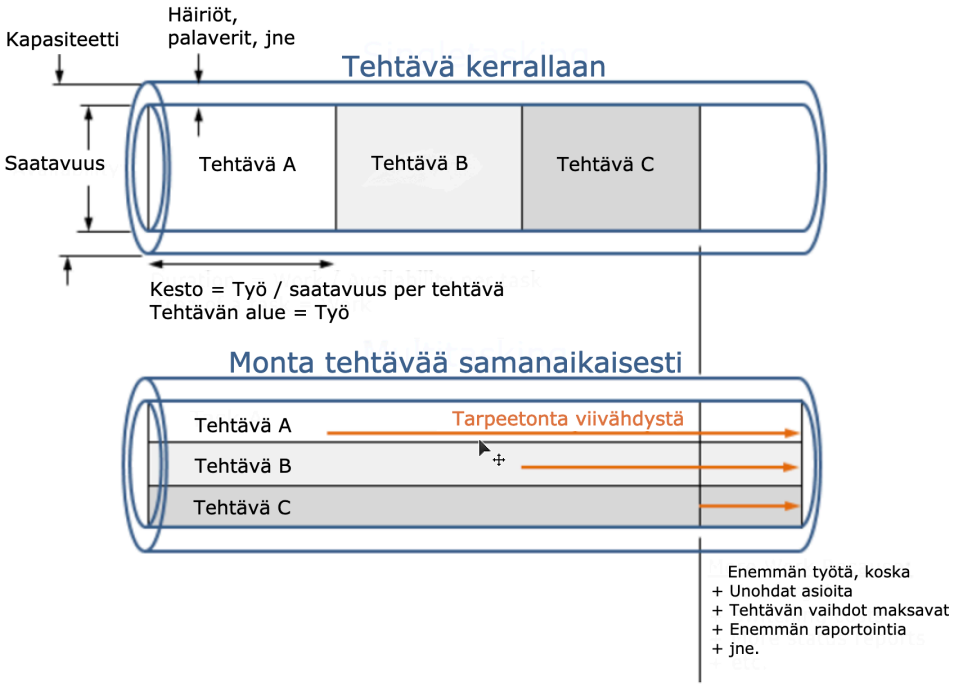
- kummankin tehtävän yksittäinen suoritusaika piteni 33-40 %
- molempien tehtävien yhteensä laskettu suoritusaika piteni 68-80%
- 23% ihmisistä teki tehtävissä enemmän virheitä.

Tutkimuksessa todettiin, että tulokset huononevat radikaalisti, mikäli yhtäaikaisia tehtäviä olisikin enemmän kuin kaksi. Mikäli Scrum-tiimi kuitenkin työskentelee useamman projektin parissa, tulee sprintin tehtävien kuitenkin vähintäänkin löytyä samasta kehitysjonosta, eikä saman aikaisesti saisi olla auki useampia tehtäviä. Usean tehtävän tekeminen samanaikaisesti saattaa auttaa muita jatkamaan työtään eteenpäin, mutta lopulta väistämättä viivästyttää ominaisuuden valmistumista. Kuvassa 14 osoitetaan, että tehtäessä tehtäviä yhtä kerrallaan, tehtävät valmistuvat nopeammin, kuin tehtäessä tehtäviä montaa yhtäaikaisesti. Tehtävä A on valmis 10 päivässä tehtäessä se kerralla valmiiksi, kun taas tehtäessä 3 työtä saman aikaisesti, se on valmis vasta 20 päivässä. (Leach 2014, 27-31; Stettina & Smit 2016; Cohn 2006; Wells & Kloppenborg 2015, 36)



Kuva 14. Tehtävien valmistumisen nopeus kun useaa tehtävää tehdään sarjassa tai rinnakkain.

Kuvassa 14 ei ole otettu huomioon esim. tehtävän vaihtoon tai kommunikointiin kuluvaa aikaa, kuten samaa asiaa monimutkaisemmin kuvaavassa kuvassa 15. Kuvan 15 on tarkoitus osoittaa, että läpivientiaikaan vaikuttavat myös mm. asiat kuten kokoukset, lisäraportointi, asioiden unohtaminen, jne.



Kuva 15. Hävikin synty ja läpivientiajan kasvaminen tehtäessä montaa tehtävää samanaikaisesti.

Liitteeen 1 taulukossa on kuvattuna yhden tuotekehitystiimin projektien tuotekehitysprosessin läpimenoaikoja. Aloitus- ja lopetusajankohdat kertovat, että tiimi on joutunut pitämään ainakin 6 projektia samanaikaisesti mukana tuotekehityssprinteissään. Kaikki projektit eivät ole olleet välttämättä aktiivisen kehityksen kohteena, mutta ovat vaatineet tiimiltä kuitenkin kommunikointia ja erillistä raportointia, sekä tukea mm. dokumentointiin, testaukseen ja julkaisuun. Lisäksi on ollut riski, että projektit palautuisivat tuotekehitystiimin työpöydälle aktiivisen kehityksen alle, vaikka aikaa projektin edellisestä aktiivisesta hetkestä olisi kulunut jo kuukausia. Asioiden unohtamisen vuoksi ja tehtävien suunnittelemattomuuden vuoksi, tehtävän vaihdot vievät tällaisessa tilanteessa erittäin paljon aikaa. Koska projekteja on niin monta työn alla samanaikaisesti, on myös selvää, että työtä ei ole tehty organisaationa oikeiden prioriteettien mukaisesti.

6.4.2 Epäonnistuneet työmääräarviot

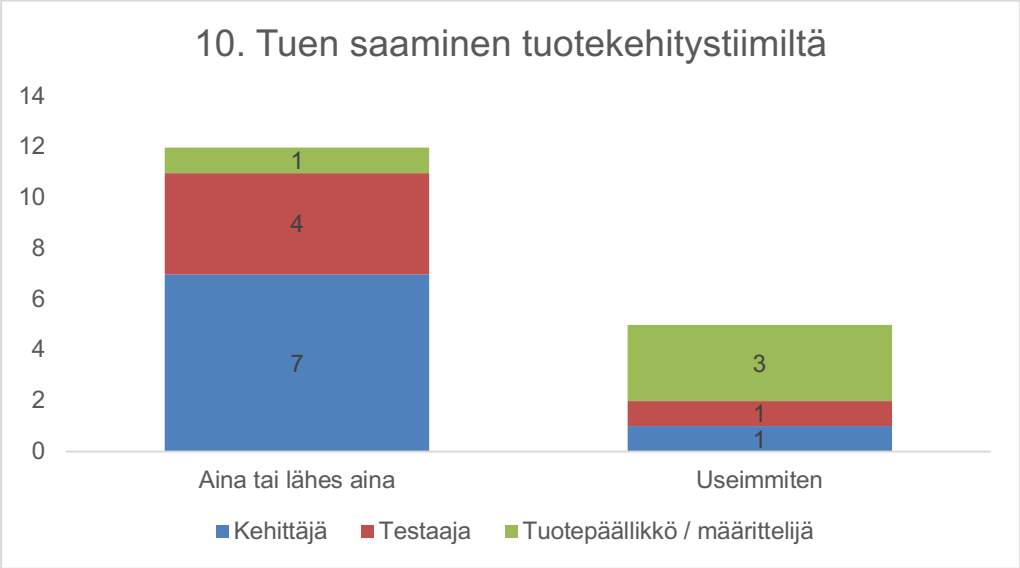
Aikataulujen arviointi on aina vaikeaa, mutta työmääräarvion tulisi osua mahdollisimman lähelle todellista toteutuvaa työmäärää, jotta ennustettavuus projektin hallinnassa säilytettäisiin, projekteja pystyttäisiin priorisoimaan ja tuotekehitystä voitaisiin suunnitel-

la hyvissä ajoin etukäteen. Työmääräarvion ollessa projektille liian suuri, oltaisiin työvoimaa voitu resursoida toisiin projekteihin aikaisemmin. Kun arvio taas on ollut liian pieni, kärsii kehityksen laatu, sillä työssä joudutaan kiirehtimään ja siten myös hosumaan, koska aikataulurajoituksen painostavat kehitystä. Lisäksi muut projektit kärsivät, kun ne eivät toteudukaan suunnitellussa aikataulussa. Molemmissa tapauksissa arvio on siis epäonnistunut. Hyvällä ennalta suunnittelulla pystytään vaikuttamaan projektin maksimaalisen arvon tuottamiseen.

Ongelma työmäärien arvioinnin suhteen ei varsinaisesti ole tuotekehitysprosessin ongelma, sillä arviot, jotka tehdään itse tuotekehitysprosessin aikana, osuvat hyvin kohdalleen. Tuotekehityssprintin tavoitteissa pysytään aina kohtalaisen hyvin, vaikka ympäristö onkin usein muuttuva. Ongelma koskettaa enemmän arviota, joka merkitään yrityksen tuotestrategiaan (roadmappiin) jo ennen tuotekehitysprosessin alkamista (liite 1). Määritellyt ominaisuudet vaativat enemmän aikaa, mitä tuotestrategian asettamat aikataulut antavat myöden. Tällöin joudutaan tekemään kompromisseja, jotka heikentävät tuotteen laatua, sekä työntekijöiden motivaatiota (liite 3). Kun yhtälöön lisätään kommunikaatio-ongelmat eri osastojen välillä, kohtaamattomat projektien prioriteetit ja niistä aiheutuva odottelu, sekä työn uudelleen määrittely ja työstö ongelmien ja kiireellisen aikataulun vuoksi, arviot työmäärästä epäonnistuvat kokonaisuutena. Tämä johtaa siihen että osastojen välinen yhteistyö tuotteen edistämiseksi toimi tehokkaasti.

Frederick Brooks kirjottaa kirjaklassikossaan ”The Mythical Man-Month” miten aikatauluviiveet aiheuttavat usein ihmisten lisäämistä projekteihin, vaikka lisätekijät aiheuttavat myös lisää kommunikoinnin tarvetta. Liiallinen kommunikaationtarve dominoi työn tekemistä työn edistymisen kustannuksella. Aiheeseen liittyen Brooks julistaa teoksessaan Brooksin säännön: ”Tekijöiden lisääminen viivästymisen korjaamiseen vain viivästyttää lisää”. (1995, 25)

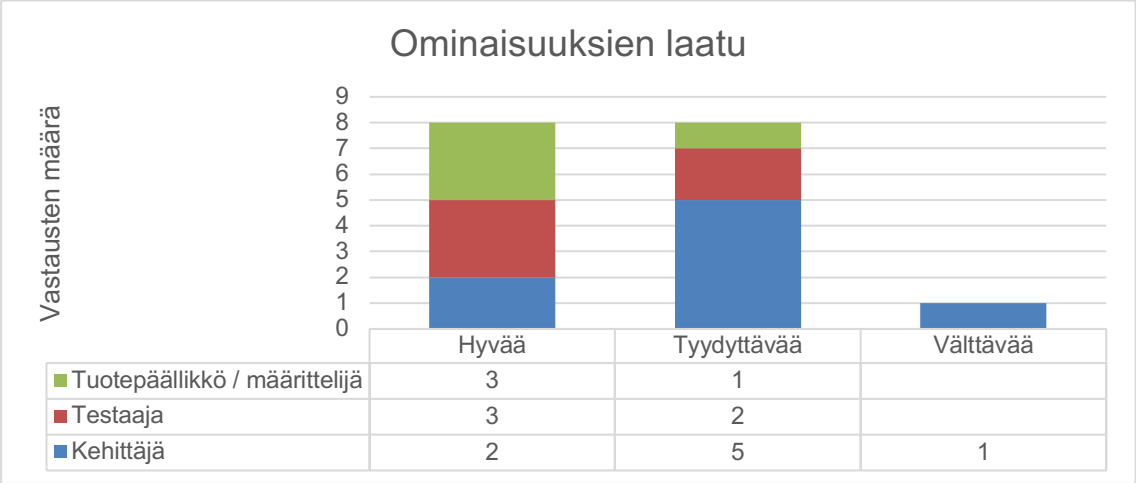
Tuotekehitystiimin rakenne koetaan haastatteluiden mukaan melko onnistuneeksi (kuvaaja 11). Kommunikaatio tiimin sisällä toimii hyvin ja apua saadaan tiimikollegoilta nopeasti. Lisätyöntekijät tiimissä tarvitsisivat myös lisäsuunnittelua sprintin sisältöön, mikä taas tarkoittaisi jälleen enemmän suunnittelutyötä, johon tuoteomistajilla ja tuotekehitystiimillä ei nykyisellään ole aikaa.



Kuvaaja 11. Haastattelun 10. kysymyksen ”Tuen saaminen tuotekehitystiimiltä” vastaukset.-

6.4.3 Kiire ja ominaisuuksien laatu

Kiire ja stressi vaikuttavat työn jälkeen ja siten koko ohjelmoitavan tuotteen laatuun. Kuvaaja 12 esittää haastattelussa (liite 3) annetut vastaukset rooleittain. Huomioitavaa on, että yksikään vastaaja ei vastannut ominaisuuksien laadun olevan kiitettävää. Mielipiteet jakautuivat yhtä ”välttävää” -vastausta lukuun ottamatta tasan hyvän ja tyydyttävän välillä. Ominaisuuksia määrittelevät tuotepäälliköt olivat ominaisuuksien laadun suhteen vähemmän kriittisiä kuin ominaisuuksia ohjelmoivat kehittäjät. Lähes kaikki vastaajat perustelivat kiireen olevan suurin negatiivinen vaikuttaja ominaisuuksien laatuun. Ominaisuuksia ei ehditä suunnitella eikä testata halutulla ja riittävällä tavalla. Etenkin ominaisuuksien laadun ollessa vain tyydyttävää, syntyy suuri uhka sille, että ne joudutaan suunnittelemaan ja toteuttamaan uudelleen. Heikko laatu vaikuttaa erityisen paljon myös tuotteen ylläpitoon sitä vaikeuttavasti. Mikäli heikko laatu kohdistuu kerättävään dataan tai sen rakenteeseen, sillä saattaa olla työllistävää vaikutusta myös raportointiin. Mikäli aikaa saataisiin suunnittelulle enemmän, ominaisuuksien laatu todennäköisesti kasvaisi.



Kuvaaja 12. Haastattelulomakkeen kysymyksen 15 ”Onko toteutettujen ominaisuuksien laatu mielestäsi keskimäärin” vastaukset rooleittain.

6.5 SWOT-analyysi

SWOT-analyysilla voidaan kuvata tuotekehitysprosessin heikkoja kohtia, sekä selvittämään mitkä ovat nykyisen prosessin vahvuudet. Taulukko 8 on SWOT-analyysi tuotekehitysprosessista haastatteluiden liitteen 3 tulosten pohjalta. Tuotekehitysprosessin hybridimallius tekee SWOT-analyysistä vaikeaa, sillä osa vastauksista on luonteeltaan sellaisia, että ne ovat vahvuuksia mikäli tuotekehitysprosessia suoritettaisiin vesiputousmallilla, mutta heikkouksia suoritettaessa puhdasta ketterää kehitystä. Osa puolestaan on vahvuuksia ketterässä kehityksessä, mutta heikkouksia vesiputousmallissa. Esimerkiksi ”muuttuva määrittely” voidaan kokea arvoa nostavaksi ketterässä kehityksessä, mutta vesiputousmallia suoritettaessa se pilaa mm. suunnitellut aikataulut ja ennalta kirjoitetut testitapaukset.

Mahdollisuudet-osio sisältää paljon sellaisia asioita, joita ketterän kehityksen lisääminen toisi tullessaan. Mahdollisuutena olisi ketterän iteratiivisen kehityksen myötä työskennellä tiiviimmässä yhteistyössä ja saada useampien katselmointien kautta enemmän ja aikaisemmin palautetta tehdystä toteutuksesta.

Uhat-osio koostuu suurimmaksi osaksi kiireen ja monen projektin parissa samaan aikaan työskentelemisen aiheuttamista uhkakuvista. Kiireen ja monen yhtäaikaisen projektin myötä laatu heikkenee, tehtäviä ymmärretään väärin ja sama työ joudutaan tekemään turhaan moneen kertaan. Uhkiin on sisällyttävä myös hybridimallius, jonka myötä prosessi sekavoituu ja siihen muodostuu paljon hävikkiä. Uhkana on, että mää-

rittelyt eivät ole riittävän selkeitä, jotta aikataulussa pysyttäisiin, eikä määrittelijöiltä kuitenkaan saada riittävästi tukea projektinhallintaan ja määrittelyiden iteroimiseen.

Tuotekehitysprosessin hyvinä puolina haastattelussa mainittiin asioita, jotka ovat ominaisia ketterälle kehitykselle: Lukittu tehtävälista, jatkuva arviointi, jatkuva testaus, nopea reagointi muutoksiin, jne. Huonoina puolina listataan mm. moneen projektiin keskittyminen saman aikaisesti, epäselvät tehtäväkuvaukset, aikataulun tiukkuus, muuttuva määrittely, jne. jotka ovat kaikki suurimmaksi osaksi projektin hallinnallisia ongelmia, joihin esim. selkeä tuoteomistajuus toisi helpotusta.

Taulukko 8. SWOT-analyysi tutkimuksen haastattelun vastauksista.

Mahdollisuudet	**Hyvät puolet**
- Joustava tekeminen - Tiiviimpi yhteistyö - Jatkuva kehittyminen - Tuotteille yhteinen modulaarinen alusta - Scrum - Ketterä kehitys - DevOps - Demotilaisuudet - Tuoteomistajuus - Käyttäjätarinat - Kehitysjonon työstäminen (Backlog Grooming) - Hyvät määrittelyt - Retrospektiivit - Yhteen projektiin keskittyminen - Tilastoinnin lisääminen - Useampia katselmointeja	- Sprintin tehtävät lukittu - Työrauha on turvattu sprintissä - Jatkuva arviointi - Jatkuva parannus - Nopea reagointi muutoksiin - Jatkuva testaus - Tuen saaminen tuotekehitystiimiltä

Uhat	**Huonot puolet**
- Sama työ joudutaan tekemään moneen kertaan - Kaikki tehtävät eivät ole kirjattuna kehitysjonoon - Tehtävät on ymmärretty väärin - Arkkitehtuuri on keskeneräinen - Prioriteetien vaihtuminen - Resurssien yliallokointi - Määrittelyn paikoittainen heikko laatu - Ominaisuuksien laatu kärsii kiireestä - Muutosvastarinta prosessin kehittämiselle - Tuotteen suorituskyky kärsii kiireestä - Dokumentaatio kärsii kiireestä - Asiakas ei tiedä mitä haluaa - Hybridimalli	- Muuttuva määrittely - Liian tiukat aikataulut - Kiire - Epäselvät tehtävät - Monen tuotteen yhtäaikainen tekeminen - Asiakas ei osallistu tuotekehitysprosessiin - Paljon hävikkiä - Epäonnistuneet työmääräarviot - Hybridimalli

7 TEHOSTETTU TUOTEKEHITYSPROSESSI

Edellisessä luvussa nostettiin esille tutkimuksessa löydettyjä tuotekehitysprosessin ongelmakohtia. Tämän luvun tarkoituksena on paneutua tarvittaviin muutoksiin tuotekehitysprosessissa, sekä organisaation kulttuurissa. Ketterän kehityksen ja DevOps-filosofian kulttuuria on yritetty jalkauttaa organisaatioon jo vuosien ajan, mutta tutkimustulokset osoittavat, että organisaatiolla on vielä kulttuurissa sisäistämistä.

7.1 Ketterä kehitys ja DevOps

Kehityshankkeen toimeksiantajan tuotekehitysprosessin on oltava ketterä, sillä siinä on pystyttävä nopeisiin muutoksiin niin laajuudessa kuin ominaisuusvaatimuksissa. Samalla sen on myös täytettävä korkealaatuisten järjestelmien toimittamisen tarpeet kustannustehokkaalla tavalla, joka ei häiritse liikaa projektiin osallistuvia asiakkaita, tuotepäällikköä tai kehitystiimiä. Organisaatio on pyrkinyt tuotekehitysprosessissansa noudattamaan ketteriä kehitysmenetelmiä, mutta on epäonnistunut sisäistämään kulttuurin ja noudattamaan niiden sääntöjä. Ketterän kehityksen tai DevOpsin kulttuuri on asia, joka tulee ymmärtää ja sisäistää koko organisaation laajuudelta. Ilman myöntyvää kulttuuria, parhaatkaan työkalut eivät ole avuksi.

DevOps-kulttuuri on mm. avoimuutta, yhdessä tekemistä ja vastuun jakamista. Siinä ei syytellä muita, vaan luotetaan toisten tekemiseen ja palkitaan onnistumisista. Yhteisen vastuun asenne on DevOps-kulttuurin osa, joka kannustaa tiiviimpään yhteistyöhön. Jos kehitystiimi jakaa vastuuta järjestelmän hoidosta koko sen elinkaaren ajan, he voivat jakaa tuotannon tuskan ja tunnistaa keinoja käyttöönoton ja ylläpidon yksinkertaistamiseksi (esim. automatisoimalla käyttöönottoa ja parantamalla ohjelman toiminnan lokitusta). Kun koko henkilöstö jakaa vastuun järjestelmän liiketoiminnallisista tavoitteista, kaikki ymmärtävät tiiviimmän yhteistyön kautta paremmin järjestelmän toiminnalliset tarpeet ja voivat auttaa vastaamaan niihin. Käytännössä yhteistyö alkaa usein lisäämällä kehittäjien tietoisuutta operatiivisista huolenaiheista. Kehityksen ja tuotannon välillä ei saisi olla esteitä. Toinen arvokas organisatorinen muutos on tukea itsenäisiä itseohjautuvia tiimejä. Tehokkaan yhteistyön varmistamiseksi kaikkien on voitava tehdä päätöksiä ja soveltaa muutoksia ilman ristiriitaisia päätöksentekoprosesseja. Ymmärrys tehtävästä asiasta kuitenkin päätöksen tekoon tarvitaan, joten kommunikaation tulee

toimia tuoteomistajan tai asiakkaan kanssa. Myös tuotantoonsiirron ja käyttöönoton automatisointi kasvattavat DevOps-kulttuurin henkeä. Kun kehitys- ja tuotantoputkea automatisoidaan, se vaatii paljon yhteistä suunnittelua tuotteen kehityksen alusta asti, jotta voidaan varmistua siitä, että automatisoinnin suorittaminen on aina yksinkertaista ja turvallista, eikä mitään tietoa tai toiminta jää puuttumaan tai hajoa toiminteen ajon yhteydessä. Myös määrittely ja tilaukset ominaisuuksista tulee tehdä siinä ymmärryksessä, että kun esim. tuotantoonsiirtoa automatisoidaan ja tuotteen konfiguraatio luetaan asiakaskohtaisista konfiguraatioista tuotantoonsiirron ajon yhteydessä, kaikkien asetusten ja säätöjen tulee olla kerralla oikein, koska ne julkaistaan käyttöön sellaisenaan tuotantopalvelimelle. (Swartout 2012)

Sisäistämällä DevOps-kulttuurin, sisäistetään samalla myös tärkeimmät piirteet siitä mitä ketterän kehityksen kulttuurin suorittamisessa tarvitaan. Ketterän kehityksen kulttuurissa päivittäisen tekemisen päätökset tulisi perustua yhteisesti ymmärrettyihin arvoihin, jotka perustuvat yrityksen yhteiseen tapaan tehdä ketterää kehitystä. Ilman todellista muutosta kulttuurissa, oikeasti ketterä organisaatio ei ole mahdollinen. Yrityksen kulttuurin muuttaminen vaatii paljon ponnisteluita, mutta sen ei kuitenkaan pitäisi koskaan olla mahdotonta. Muutos tapahtuu vähitellen. Ensimmäisiin askeleihin kuuluvat mm. yrityksen liiketoimintaidean ja asiakasarvon yhteinen ymmärtäminen. Myös laadun käsitteet, kuten valmiin määritelmä tulee yhdessä sopia ja kaikkien tulee ymmärtää se samanlaiseksi. Kaikilla tarkoitetaan koko organisaatiota, eikä pelkästään tuotekehitysprosessiin osallistuvaa henkilöstöä. Oikeasti ketterä organisaatio pyrkii aina jatkuvaan parantamiseen, mikä on myös Lean-kulttuurin peruspilareita. Kaikkien tulisi siis pelata samaan maaliin, mikä edellyttää tuotepäällikön, määrittelijöiden, tuotekehitystiimin, integraatiotiimin, testaajien ja julkaisutiimin yhteistyötä aina käyttöönottavaan asiakasvastaavien tiimiin asti. Kaikkien tulisi olla tietoisia mm. tuotekehitysprosessissa käytettävästä Scrumista, siihen liittyvistä toimintatavoista ja tapahtumista, sekä myös ymmärtää siihen asetettujen sääntöjen tärkeys. (Swartout 2012; Davis 2012)

Scrum on tuotekehitysprosessin projektinhallinnan viitekehyksenä käyttötarkoitukseensa varsin sopiva. Tuotekehitysprosessissa nykyisin käytettävät kahden viikon mittaiset sprintit eivät ole niin pitkiä, etteikö kehitys- ja tuotantosuunnitelmaa voitaisi edes siksi aikaa lukita. Asiakkaan kokemus tietojärjestelmien kehitystyöstä on terveydenhuollonalalla pientä ja ala on toiminnaltaan muuttuva, laaja ja monimutkainen, jolloin määrittelyt ovat liian herkästi muuttuvia, jotta kehitystä voitaisiin tehdä täysin kankean vesipu-

tousmallin mukaisesti. Ketteryys ja seurattavuus ovat tärkeitä avainsanoja, joihin Scrum tarjoaa ratkaisun, mikäli kulttuurinmuutos ollaan valmiita tekemään. Myös Kanbania voidaan soveltaa toimintaan etenkin ylläpidon osalta. Tärkeää kuitenkin olisi, että voitaisiin luvata varmaksi edes kahden viikon kehitysjakson pituudelta projektin etenemiseen liittyviä lupauksia, mikä ei onnistu jos sovittua kokonaisuutta muokataan.

7.2 Tuoteomistajuuden uudelleen määrittäminen

Määrittelijät ovat myyjien ja johdon lisäksi organisaation ainoa taho, joka on ollut tuotekehitysprosessin alkaessa yhteydessä asiakkaaseen. Asiakkaan prosessit ja tahtotila on ainoastaan määrittelijän tiedossa. Asiakkaan uupuessa vain määrittelijä osaa sanoa, mikä tuotteessa tuottaa arvoa ja mitä asiakas tuotteelta haluaa. Vaihtoehtoja kehittämiseen on kaksi: Tuotekehitystiimin tulee olla mukana asiakkaan kanssa käytävissä määrittelypalavereissa tai määrittelijän tulee osallistua enemmän tuotekehitysprosessiin. Mikäli tuotekehitystiimin toimintaan ei saada vakituista tuoteomistajan edustajaa, tulee määrittelijän toimia tuoteomistajana vastuullisemmin. Tuoteomistajan aktiivisempi osallistuminen pienentäisi tuotekehitysprosessissa todettua hävikin määrää merkittävästi (liite 3). Mikäli tuotekehitysprosessi ei saa tuoteomistajan tukea, prosessi epäonnistuu joka kerta (Cobb 2016; Pries & Quigley 2011; Davis 2012, 142).

Asiakkaan uupuminen tuotekehitysprosessin palavereista ei kuitenkaan aina välttämättä ole huono asia, sillä asiakkaan ollessa läsnä, palaverien tehokkuus saattaa kärsiä. Asiakas työskentelee usein eri kaupungissa, eikä hänellä ole aikaa tai osaamista osallistua ketterään kehitykseen. Projektiin liittyy usein myös yhden asiakkaan sijasta useampia asiakkaita. On kuitenkin varsin tavallista että ketterässä kehityksessä käytetään sisäistä asiakasta, joka edustaa asiakkaan tai asiakkaiden mielipiteitä ja yhteistä tahtotilaa. Asiakkaan sitouttaminen kehitystyöhön on yksi ketterän kehityksen pääperiaatteista, mutta se ei kuitenkaan tarkoita että asiakas pitäisi saada mukaan kaikkeen tekemiseen, vaan sitä, että asiakkaaseen ja heidän prosesseihinsa tulee tutustua ja ne tulee ymmärtää. Ketterässä kehityksessä tärkeää on myös luottamuksen säilyttäminen toimittajan ja asiakkaan välillä. Järvenpää ja Kovanen lähestyvät toimittajan ja ostajan luottamuspulaan liittyviä ongelmia ostajan näkökulmasta teoksessaan ”Ostajan pikaopas 2.0” (2018, 39) ja kiteyttävät hienosti luottamuksen ylläpitämisen ongelman kahteen lauseeseen: ”Mitä harvemmin osapuolet kohtaavat kasvokkain, sitä enemmän luottamuksen rappeutumisesta alkaa koitua ongelmia. Luottamuspulaa paikataan yli-

määräisellä työllä, kuten raportoinnilla, mikä ei kasvata projektin arvoa”. Lauseet pätevät täysin myös kehitystiimin ja sisäisen asiakkaan väliseen luottamukseen. (Cobb 2016, 197; Coplien & Bjørnvig 2010, 29)

Sprintin roolit ja roolien vastuut tulee kuvata prosessissa selkeästi. Tuoteomistajan vastuiksi tulisi luetella määrittelyn ja määrittelyasiakirjojen ylläpidon lisäksi, tuotekehitysjonon hoitaminen ja tuotekehitystiimin tukeminen ensisijaisesti ohi muiden töiden. Mikäli tuotekehitystiimi ei saa tarpeeksi tukea, he käyttävät turhaa aikaa määrittelyiden tulkintaan, ratkaisun pohdintaan, mahdollisen väärän ratkaisun tekemiseen ja myöhemmin väärin tehdyn ominaisuuden korjaamiseen. Pahimmassa tapauksessa asiakkaan keräämän datan eheys saattaa olla vaarassa tai kerätty data ei vastaa lainkaan asiakkaan toivetta. Tuoteomistajan tulee ottaa holhoavampi asema ja enemmän vastuuta siitä, että tuotekehitystiimi tekee työnsä oikein. Läpinäkyvyys on erityisesti tuoteomistajan intressissä., vaikka scrummaster auttaa sen toteuttamisessa. Tuoteomistajan tulisi pyytää tuotekehitystiimiä esittelemään toteuttamansa ratkaisut. Sisäisen tuoteomistajan tehtäviin tulisi listata selkeästi myös vastuu siitä, että projektiin ei valita tehtäväksi ominaisuuksia, jotka eivät ole arvon tuoton kannalta oleellisia.

7.3 Enemmän määrittelyä ja suunnittelua osaksi ketterää tuotekehitysprosessia

Määrittelyidenhallinta ketterässä kehityksessä on perinteistä kehitystä hankalampaa, johtuen mm. siitä että tuotekehitystiimi voi keskittyä vain muutaman käyttötapauksen työstämiseen kerrallaan ja tuotteen määritelmät elävät jatkuvasti jokaisen kehitysjakson alkaessa. Tuotteen käyttötapausten kirjaaminen voidaan tällaisessa tilanteessa kokea usein hankalaksi, joten tuotteen kehitysjonon eli backlogin hallintaan tarvitaan ketterässä kehityksessä perinteistä kehitystä enemmän hoitamista ja yhteistyötä. Leanin ja ketterän kehityksen avain onnistumiseen sanotaankin olevan alusta alkaen yhdessä tekeminen. (Cobb 2016, 196-197; Coplien & Bjørnvig 2010, 38)

Järvenpää ja Kovanen patistavat kirjoittamassaan teoksessa ”Ohjelmistokehityksen ostajan pikaopas 2.0” (2018) unohtamaan suunnitelman ja keskittymään jatkuvaan suunnitteluun: ”Pyrkimys muutoksen hallintaan monimutkaisilla prosesseilla ei kasvata kehitettävän tuotteen arvoa. Suunnitelmat on syytä unohtaa, sillä ne vanhenevat. Suunnittelu sen sijaan kannattaa aina.” Muutokset ja uudelleen määrittely ovat ketterässä kehityksessä aina tervetulleita (myös projektin loppuvaiheella). Muutoksia ei kuitenkaan koskaan voida tehdä ilman seuraamuksia. Muutokset vaikuttavat aina tuottee-

seen aiemmin tehtyihin ominaisuuksiin ja toiminnallisuuksiin. Lisäksi muutosten käsittely ja työstäminen vaikuttaa tehtyihin suunnitelmiin, testaukseen, sopimuksiin, jne. Muutostenhallinnassa ei ole kuitenkaan tarkoitus estää muutosten syntymistä, vaan siinä tulee keskittyä erityisesti siihen, että jokainen tehtävä muutos on tuotteen toiminnan kannalta oikeasti tarpeellinen ja tuottaa lisää arvoa. Mikäli sopimuksessa on lukittuna myös projektin aikataulu kuten hankkeen toimeksiantajalla, joudutaan ominaisuuksista karsimaan tuoreimman priorisoinnin mukaiset vähiten tärkeät ominaisuudet pois. Sopimusta tehdessä tulisi ehkä ominaisuuksien sijasta myydä tuotekehityssprinttejä, joille kuitenkin voidaan suunnitella jo alustavaa sisältöä. Tällöin aikataulun lukitsisi sprinttien määrä, eikä kiinteä lista määriteltyjä ominaisuuksia. Tuotekehityssprinttien myyminen vaatii hyvin erilaista lähestymistapaa myyntitiimiltä, sekä tuotekehitystiimejä, joissa on kaikki vaadittava osaaminen (myös integraatio-osaaminen yms.). Sprinttien myyminen vaatii myös vahvaa luottamusta asiakkaalta. Uusista vaikeuksista huolimatta, sprinttien myyminen olisi hyvin toivottava tapa toimia ja auttaisi ketteröittämään prosessia merkittävästi. (Cobb 2016, 198; Coplien & Bjørnvig 2010, 38-41; Järvenpää & Kovanen 2018)

Nykyisin tuoteomistajat ovat kiireisiä uusien tuotteiden määrittelyiden kanssa tuotekehitysprosessin aikana (liite 3). Tuotteen määrittelystä saattaa olla jo paljon aikaa, eikä määrittelyt ole tuoreessa muistissa. Jos on selvää että määrittelyt tulevat kehityksen aikana muuttumaan, olisi todennäköisesti järkevämpää järjestää enemmän katselmointeja tuotekehityksen aikana ja siirtää määrittelyyn käytettävää työaikaa enemmän osaksi tuotekehitysprosessia. Näin ollen asiat olisivat tuoteomistajallakin tuoreessa muistissa ja kommunikointiyhteys olisi auki asiakkaaseen. Tällöin määrittely- ja tuotekehitysprosessi synkronoituisivat itsestään. Asiakkaan kanssa tulee tehdä selväksi, kuinka monta tuotekehitysjaksoa he tulevat antamalla panoksellaan saamaan, mikäli kehitystyönsopimus on tehty kiinteällä hinnalla. Määrittelyprosessissa voitaisiin edelleen suunnitella asiakkaan kanssa suuret linjat, mutta määrittelyn tarkennusta voitaisiin tuoda osaksi tuotekehitysprosessia, jolloin myös tuotekehitystiimi voisi osallistua palavereihin. Suurten linjojen määrittelyn suhteen voitaisiin myös siirtyä enemmän tarpeiden ja vaatimusten määrittelyyn ja keskittyä iteroimaan useampia katselmointikierroksia. Määrittelyprosessin aikana voitaisiin asiakkaan kanssa sopia pienemmästä projektiryhmästä, joka osallistuisi määrittelyyn tuotekehitysprosessin aikana, jottei päättäjiä ole prosessissa liikaa.

7.4 Käyttäjätarinat ja kehitysjonon vastuullisempi hallinta

Muutosten kuvaaminen tulee olla selkeämpää. Muutokset tulee kirjata tuotteen tekniseen dokumentaatioon ja jokaisesta muutoksesta tulee kirjata selkeä tehtävä tuotteen kehitysjonoon. Ketteryys ei tarkoita etteikö nopeastikin tehtäviä muutoksia tarvitsisi kuvata muutostenhallintaan ja kehitysjonoon. Kirjattaessa asiat huolellisesti, säilyy myös päätösten jäljitettävyys. Myöhemmin, kun ominaisuuteen tehdään muutoksia, voidaan tarkistaa syyt ja ajankohta aiempaan muutospyyntöön, selkeästi yhdestä paikasta. Sen sijasta, että määrittelyä tehdään sähköpostein ja pikaviestimin, keskustelu kannattaisi ohjata suoraan käyttäjätarinaan tuotteen kehitysjonossa, jotta päätökset olisi löydettävissä yhdestä paikkaa, listasta, joka ensisijaisesti ohjaa tuotekehitysprosessissa toteutettavia ominaisuuksia. Kehitysjono on kaikkien saatavilla aina samasta paikasta, se kertoo ajantasaisen tilanteen tuotteen ja sen ominaisuuksien tilasta ja sieltä on nähtävissä myös, mitkä muutostoiveet ovat vielä toteuttamatta. Kun kaikki käyttävät samaa reaaliaikaista kommunikaatiokanavaa, vältytään myös turhilta kysymyksiltä ja vastaamiselta moneen kertaan, sillä tieto välittyy yhtä aikaisesti niin tuoteomistajalle, kehitystiimille, testaajalle, kuin käyttöönottoa tekevälle asiakasvastaavallekin.

Kehitysjonon sisällön kuvaamiseen tulisi käyttää käyttäjätarinoita, joilla on selkeä ja yleisesti käytetty yksinkertainen formaatti: Kuka on ominaisuuden loppukäyttäjä, mitä ominaisuus sisältää ja miksi. Käyttäjätarinaa kannattaa suosia mm. siksi, että se on helppo tapa opettaa kaikille, myös esim. loppukäyttäjille. Lisäksi se on formaatti, joka toimii sellaisenaan myös testitapaukseksi. Käyttäjätarinoita voitaisiin kirjoittaa tarinankirjoitussessioissa, joihin voi osallistua myös asiakkaan edustajia. Tapahtuman tarkoituksena olisi luoda mahdollisimman suuri määrä tarinoita, ottamatta kantaa siihen, mitä niistä lopulta toteutetaan. Käyttäjätarinoiden pohjalta on helppoa kirjoittaa myös määrittelydokumenttia.

Kehitysjonon hallinta, uusien tehtävien kuvaaminen ja kirjaaminen, sekä niiden valmistumisen seuranta ovat toimeksiantajaorganisaation tuotekehitysprosessissa nykyisellään täysin tuotekehitystiimin (tai tiimin vetäjän) työtä. Kun projekti alkaa, tuotekehitystiimin vetäjänä toimiva kehittäjä, joka hoitaa myös scrummasterin roolia, kirjaa käyttäjätarinat tehtävälistaan. Toimeksiantajan toiminnassa, tuoteomistaja, jonka velvollisuus ketterän kehityksen sääntöjen mukaan on olla vastuussa kehitysjonosta, ei välttämättä koskaan näe projektinsa tuotekehitysjonoa tai sen sisältöä. Kehitysjonon työstämiseen omistetuissa Scrum-tapahtumissa sisältöä käydään läpi, mutta mikäli tuoteomistaja ei

osallistu ko. palaveriin, on hyvin todennäköistä, että hän ei kehitettävän tuotekehitysprosessin aikana näe tai tarkasta listan sisältöä lainkaan. Tuotteen kehitysjono on tärkeä työkalu kehitystiimille, sillä se on se lista tehtäviä, jonka avulla tiimi seuraa mitä tuotteessa on vielä tekemättä. Tästä syystä, olisi hyvä että tuoteomistajan velvollisuuksiin kirjattaisiin vähintäänkin kehitysjonon sisällön tarkastus ja puuttuvien käyttäjätarinoiden lisääminen, ennen kuin tuotekehitysprosessi aloittaa projektissa tehtävän koodaamiseen. Määrittelydokumentin sijasta tulisi tehdä enemmän itse määrittelyä. Tuoteomistajan olisi hyvä käydä päivitetty lista läpi myös viimeistään päivää ennen tuotekehityssprintin päättymistä, jotta seuraavan sprintin aloituspalaverissa olisi valmiiksi karsittu ja priorisoitu lista vielä tekemättömiä, mutta kuvaukseltaan päivitettyjä, käyttäjätarinoita. Suunnittelussa ja työmäärän arvioinnissa olisi hyötyä myös ns. julkaisusuunnitelmasta, jossa suunniteltu määrä sprinttejä on valmiiksi kuvattuna alustavalla ohjelmalla. Tapahtumiin ennalta valmistautuminen tehostaa myös palaveria. (Davis 2012; Cobb 2015)

Kehitysjonon työstäminen eli groomaus (backlog grooming) on yksi Scrumin tärkeimpiä toiminteita. Tuotteen kehitysjonon työstämisen yhteydessä kehitystiimi ja tuoteomistaja käyvät yhdessä läpi tuotteen kehitysjonoa ja siihen liittyviä epäselvyyksiä. Mikäli tapahtuma pidetään tuloksen kannalta tehokkaana, epäselviä asioita ja häirintää itse kehityksen aikana tulee olemaan vähemmän. Koska tuoteomistajalla on jatkuvia kiireitä ja muita projekteja, yhdessä sovitut suunnitellut tapaamiset pitäisi olla tuoteomistajalle mieluisia, sillä tällöin hän voi suunnitella enemmän tekemistään, eikä joudu niin paljoa spontaanin häirinnän kohteeksi. (Davis 2012; Cobb 2015).

7.5 Yhteen projektiin keskittyminen

Projektien rinnakkaista työstämistä tulisi pyrkiä välttämään ja koko organisaation tulisi keskittyä ensisijaisesti tuotekehityksessä olevaan tuotteeseen. Organisaation prosessit eivät pysy synkronissa, kun tuotteita tehdään liian montaa kerrallaan. Suurin synkronointiongelma tuotekehityksellä tällä hetkellä tuotepäälliköiden ja integraatiotiimin kanssa. Mikäli tuoteomistaja on ollut aktiivisemmin tuotekehitysprosessissa mukana, myös tuotteen integrointi asiakkaan järjestelmiin on ollut jouhevampaa. Yhteen projektiin keskityttäessä tuotteiden laatu paranisi, koska tehokkaiden tuntien määrä lisääntyisi, virheiden määrä pienenisi ja voitaisiin pohtia suurempien kokonaisuuksien koko elinkaarta, sekä keskittyä yhdenlaiseen osa-alueeseen kerrallaan, eikä vuorotellen ajoittain jopa hyvin erilaisiin tuotteisiin. Tuoteomistajien määrän lisääminen vähentäisi

yhden tuoteomistajan vastuulle jäävien projektien määrää. Tuoteomistajien lisääminen kuitenkin todennäköisimmin nostaisi myös samaan aikaan tehtävien projektien määrää, mikä jälleen vaikeuttaa synkronointia tuotekehitysprosessin ja muiden prosessien välillä. (Luku 6)

Syyksi usean projektin rinnakkaiseen tekemiseen on annettu usein liiketoiminnalliset syyt. Projekteja tulee saada alulle, jotta niistä saataisiin tuloja ja pysyttäisiin tuotestrategian (roadmap) määrittelemässä aikataulussa. Luvussa 6 käsiteltyjen asioiden valossa, yhteen projektiin kerrallaan keskittymällä päästään kuitenkin vähintäänkin yhtä hyvään tulokseen, sillä tällöin projekteja saadaan myös valmiiksi nopeammin. Kun keskittyminen keskitetään yhteen tai edes pienempään määrään projekteja, hävikki tulisi pienenemään ja tuote saadaan aikataulullisesti nopeammin tuotantokäyttöön.

7.6 Tilastoinnin lisääminen

Yksi jatkuvan toimittamisen ja DevOpsin tärkeimpiä asioita on ympäristöön liittyvien asioiden jatkuva seuranta ja tilastointi. Jotta DevOps ja jatkuva parantaminen toimisi, se on jopa välttämätöntä. Tilastointiin ja seurantaan on olemassa paljon erilaisia työkaluja. Työkalujen löytäminen ei usein koidu hankalaksi, vaan usein ongelmaksi osoittautuu seurattavien asioiden keksiminen, sekä niiden vaatimat kustannukset. Tärkeintä jatkuvassa toimittamisessa ja DevOpsissa ovat tehokkuuden ja tuotantokapasiteetin parantaminen, koska ne ovat sellaiset tekijät jotka vaikuttavat suoraan siihen, kuinka nopeasti tuotteet voidaan toimittaa markkinoille ja kuinka nopeasti yritys alkaa saada niistä kehityksestä tuottoa. Tilastoinnissa kannattaakin keskittyä sellaisiin mittareihin, jotka seuraavat tuotannon läpimenoaikoja ja auttavat hidastavien tekijöiden arvioinnissa. (Swartout 2012, 91, 148)

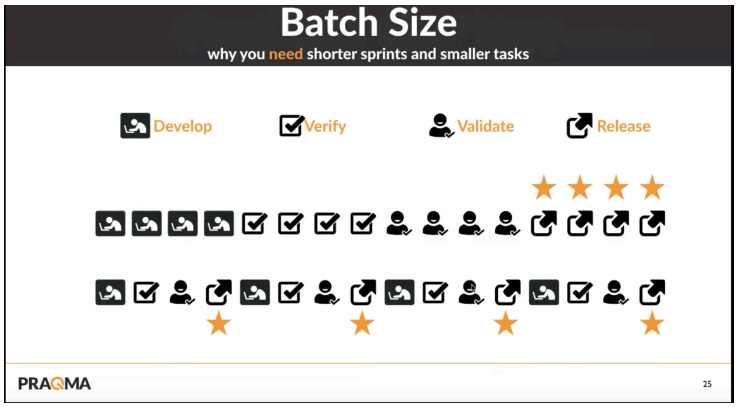
Retrospektiivit ovat tapahtumia, joissa pyritään aina keskustelemaan ja kehittämään edellisen kehitysjakson ongelmia. Tapahtuman aikana kerätään hyvin ja huonosti menneet asiat listaan ja sovitaan prosessin kehittämisen jatkoaskelista. Retrospektiivin tuloksena on luoda tehtävälista kehitettävistä asioista, joiden toteutumista tulee myös seurata. Tapahtumassa käydään läpi myös tulokset aiemmin sovituista asioista. Haastattelun tuloksista (liite 3) selviää, että sprintin retrospektiivit eivät kuitenkaan ole kovin tehokkaassa käytössä, sillä osa vastaajista ei ole koskaan osallistunut yhteenkään sprintin retrospektiiviin ja osa sanoo että retrospektiivissä vain listataan asioita, mutta itse kehittyminen jää tapahtumatta. Oleellista olisi tarkastaa myös sprintin raportit, ku-

ten työmäärän arvioinnin onnistuminen esim. suunnitellun ja toteutuneen työmäärän välisestä suhteesta tai suunnittelemattomien tehtävien työmäärästä sprintissä. Sprintin retrospektiivien lisäksi organisaatiossa tulisi järjestää myös laajempia retrospektiivejä jatkuvan kehittymisen tueksi. (Swartout 2012, 46 - 47)

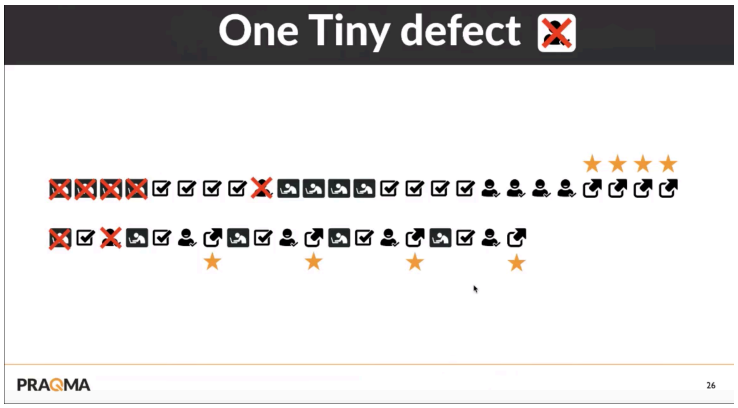
7.7 Pienemmät, ketterämmät ja useammin katselmoitavat ohjelmaversiot

Projektin pilkkominen pieniin useammin julkaistaviin osiin, yksinkertaistaa suunnittelua, auttaa riskien hallinnassa, sekä helpottaa tuotteen jatkokehittämistä. Pienemmät versiot siis mahdollistavat tuotteen nopeamman kehittämisen, monimutkaisemman ja harvemmin julkaistavan tuotteen sijasta. Projektin ominaisuudet tulee pilkkoa alle viikon työllä tehtäviin pienempiin selkeisiin ominaisuuksiin, jotta ominaisuudet saadaan mahtumaan tuotekehityssprintin aikana tehtävään julkaisuun ja tuotteeseen saadaan valmiita toteutuksia nopeasti ja se voidaan julkaista useammin. Näin tuote saadaan useammin nähtäville tai käyttöön ja myös palautetta saadaan nopeammin ja useammin, jolloin myös uusiin toiveisiin voidaan reagoida nopeammin. (Forsgren ym. 2018, 85; Auer 2013, 14; Swartout 2012, 72)

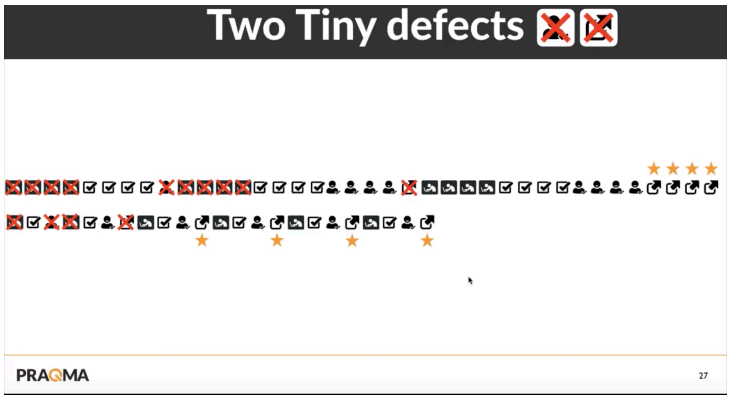
Kehittäminen pienemmillä ohjelmaversioilla (tai lyhyemmillä kehitysjaksoilla) auttaa myös säästämään aikaa. Kuvat 17, 18 ja 19 ovat IT-alan konsultointiyhtiön Praqman opastusmateriaalista ja ne kuvaavat eroa lyhyemmässä ja pidemmässä kehitysjaksossa kehittämisessä. Kuvassa 17 toiminnoiltaan identtinen ohjelmaversio on julkaistu kerralla tai 4 osassa ilman virheitä, eikä valmistumisajassa tällöin ole eroja. Kuvassa 18 on havaittu 1 ohjelmointivirhe validointivaiheessa ja kuvassa 19 on havaittu 2 virhettä, joista toinen validoinnissa ja toinen julkaisun yhteydessä. Kuvien ensimmäinen ikoni tarkoittaa kehittämistä, toinen verifiointia (todentamista), kolmas validointia (kelpuutusta) ja neljäs julkaisua. Mikäli tuote pilkottaisiin pienempiin palasiin, saataisiin palaute tuotteen toimivuudesta aikaisemmin ja virheisiin voitaisiin reagoida nopeammin. Lopputuloksena pienemmillä ohjelmaversioilla tehty projekti on kokonaisuudessaan valmis nopeammin kuin suuremmilla ohjelmaversioilla tehty. Katselmoinnissa syntyneet uudet toiveet tulee priorisoida, jolloin keskitytään suurimman arvon työstämiseen, sillä myös toivotussa tai sovitussa aikataulussa on pysyttävä.



Kuva 16. Tuote valmiiksi yhdellä tai useammalla julkaisulla ilman virheitä (Praqma 2017).



Kuva 17. Tuote valmiiksi yhdellä tai useammalla julkaisulla, kun projektin aikana löydetään yksi pieni virhe (Praqma 2017).



Kuva 18. Tuote valmiiksi yhdellä tai useammalla julkaisulla, kun projektin aikana löydetään kaksi pientä virhettä (Praqma 2017).

Myös Paul Swartout kirjoittaa samasta pienempien versioiden ja lyhyempien sprinttien aiheuttamasta efektistä kirjassaan Continuous Delivery and DevOps: A quickstart guide (2012). Hän kirjoittaa että pienemmät versiot auttavat yksinkertaistamaan muutoksia sekä siten parantamaan tuotteen laatua. Hän mainitsee käytännön auttavan myös ominaisuuksien ja muutosten käyttöönotossa kun käyttöönotettavaa on vähemmän kerrallaan.

Kehittämishankkeessa kehitettävän tuotekehitysprosessin tuotekehityssprintin lyhentäminen ei ole mahdollista, sillä jo 2 viikon mittainen sprintti aiheuttaa haastatteluiden mukaan hankaluuksia ja tarpeetonta painetta jatkuvasta palaveeraamisesta. Useammin tehtävät asiakaskatselmoinnit sen sijaan ovat asia, mihin organisaation tulisi toiminnassaan pyrkiä. Liitteen 3 haastatteluvastausten perusteella, sitä on organisaatiossa myös toivottu.

7.8 Aiemmin tehdyn tehokkaampi hyödyntäminen

Ensimmäistä ohjelmaa suunniteltaessa suunnittelutyö usein tehdään huolellisesti ja hillitysti. Vastaan tulee mutkia ja viimeistelyä viimeistelyn perään. Ajatellaan, että toimintoja tullaan tarvitsemaan myös tulevaisuudessa toisien projektien parissa. Frederick Brooks kirjoittaa Mythical Man-month teoksessaan toisen suunnittelukierroksen olevan vaarallisin suunnittelutyö jonka arkkitehti tai suunnittelija tekee, sillä kolmatta tai myöhäisempiä ohjelmia tehdessään, hänen aiemmat kokemuksensa vahvistavat järjestelmien väliset yleiset ominaisuudet ja niiden erot määrittelevät ne kokemuksensa osat, jotka ovat erityisiä, eikä yleistettäviä. Yleisesti toista ohjelmaa yli suunnitellaan ja mukaan yritetään ottaa kaikki ideat ja kikat jotka ensimmäisessä ohjelmassa nostettiin esiin, jolloin lopputuloksena on vain iso kasa asioita, jotka eivät välttämättä ole ohjelmalle lainkaan tarpeellisia.

Mikäli arkkitehti on tietoinen ylisuunnittelun riskistä, on hänellä hyvät mahdollisuudet onnistua. Ohjelmille yhtenäisistä osista kannattaa tehdä ohjelmamoduuleja, mikä lisää ohjelmien luotettavuutta ja siirrettävyyttä. Mitä enemmän kahden eri ohjelman välillä on samaa, sitä enemmän siihen on suoritettuna valmista suunnittelu- ja laadunvarmistustyötä. Tästä hyvänä esimerkkinä toimivat esim. liitteen 1 tuotteet F, E ja D, jotka on tehty samalle toimialalle ja jakavat siksi pitkälti saman koodipohjan, mutta jokainen on aina edellistään hieman laajempi kokonaisuus. Huolimatta kasvaneesta kokonaisuudesta, tuotteet saatiin tuotekehityksessä aina edellistä nopeammin valmiiksi. Samalla

myös laadun määrä kasvoi, sillä takana oli aina enemmän ja enemmän laadun testausta ja palautetta tuotepohjan käytöstä.

Modulaarisuus tuo myös sääntöjä ja ohjeistusta uuden tuotteen suunnitteluun. Ei ole järkeä keksiä samaan tarkoitukseen käytettävää toiminnallisuutta uudelleen jokaisen projektin yhteydessä, vaan pyrkiä määrittelemään tuote siten, että aiempaa voidaan hyödyntää. Sen lisäksi että hyödytään aiemmasta testauksesta ja todetusta laadusta, myös koodaaminen, dokumentointi, tuotteen ylläpito ja julkaisu helpottuvat ja nopeutuvat.

8 TUTKIMUSMENETELMIEN ANALYSOINTI

Tutkimuksen keskeinen osa on tutkimusaineisto. Hyvä tutkimusaineisto ei takaa onnistunutta tutkimusta, mutta hyvänkin tutkimusasetelman tulee vastata asetettuun tutkimusongelmaan. Tutkimusmenetelmien analysoinnilla pyritään tarkastelemaan, miten tutkimuksessa onnistuttiin ja miten tuloksista olisi voinut saada hyödyllisempiä.

Tutkimuksessa käytettiin sekä kvalitatiivisia että kvantitatiivisia tutkimusmenetelmiä. Tutkimuksella selvitettiin ensin tuotekehitysprosessin lähtötilanne, jonka jälkeen keskityttiin tunnistamaan ja rajaamaan tuotekehitysprosessiin liittyviä ongelmia. Lähtötilannetta kartoitettiin mittarein, sekä digitaalisella haastattelulomakkeella.

Koska tämän kehittämishankkeen puitteissa ei käyttöönotettu uutta tehostettua tuotekehitysprosessia, vaan pelkästään esiteltiin kehitysideoita, ei voida varmaksi mitata tehostamisen tuomaa hyötyä tai sitä, nopeutuiko prosessi halutut 30%. Koska nykyisessä tuotekehitysprosessissa voidaan osoittaa olevan runsaasti hävikkiä, on selkeillä kehitysaskeleilla mahdollista kuitenkin saada huomattavia tuloksia aikaiseksi. Liitteen 3 mittaustulokset osoittavat, että jo pelkästään poistamalla odotus tuotekehitysprosessista, voidaan saada 15 – 20% tehohyöty. Vähentämällä yhtä aikaisten projektien määrää, tehokkaiden työtuntien määrä voidaan tutkitusti nostaa jopa useilla kymmenillä prosenteilla, kuten luvussa 6 todetaan. Myös se, että luvussa 6 todetaan toimeksiantajan tuotekehitysprosessissa esiintyvän ainakin 8 yleisintä ketterän kehityksen kohtalokasta ongelmaa, kertoo siitä, että 30% tehostaminen esitetyin parannuksin olisi mahdollista.

8.1 Lähtötilanteen kartoittaminen

Lähtötilannetta kartoitettaessa, on kvantitatiivisessa tutkimuksessa tehdyissä mittauksissa käytetty satunnaista joukkoa toimeksiantajan projekteja, jotka ovat useamman tuotekehitystiimin työstämiä. Projektit on valittu usean vuoden aikaväliltä ja ovat keskenään samankaltaisia, kaikki samalle tuotekehitysalustalle tehtyjä. Koska tuotekehitysprosessia työstetään usean tuotekehitystiimin voimin, tiimien rakenne ja velositeetti tuovat hieman hajontaa mittaustuloksiin ja mielipiteisiin haastatteluvastauksissa. Vaikka tutkimuksessa on käytetty satunnaista joukkoa projekteja, projektit ovat keskenään samankaltaisia ja mahdollisimman samankaltaisten tiimien työstämiä. Tuotekehitystiimit on aina pyritty pitämään rakenteeltaan yhtäläisinä, niin kooltaan, rakenteeltaan kuin

toimintatavoiltaan. Tuotestrategian (roadmapin) mukainen työlista on myös pyritty pitämään tuotekehitystiimien välillä samankaltaisena eli yhtä suurena ja tiimien välillä keskenään vertailtavana. Tiimien sisäinen yhteistyö on toiminut kaikissa tuotekehitystiimeissä erittäin hyvin, mikä käy ilmi myös haastattelututkimuksen tuloksista (liite 3).

Kehittämishankkeen päämääränä oli tutkia, miten toimeksiantajan tuotekehitysprosessia voitaisiin tehostaa ja lyhentää prosessin läpimenoaikaa jopa 30%. Mittaustulokset läpivientiajoista ovat kehittämishakkeen kannalta erittäin oleellisia ja kertovat tarkasti, missä kohtaa prosessia ongelmia esiintyy: Suurimmat ongelmat esiintyvät prosessin loppuvaiheella katselmoinnin jälkeen ns. projektihäntiä tehdessä. Koska läpimenoaikaa voidaan lyhentää mm. laadusta tinkimällä, on kehittämishankkeessa mitattu myös laatua. Virheiden määrä on riippuvaista siitä, kuinka paljon työtä ja muutoksia tehdään. Koska yrityksen koko on kasvanut, on myös tuotteiden määrä ollut jatkuvassa nousussa, mikä johtaa suurempaan määrään työtä ja suurempaan määrään virheitä. Toisaalta, koska alusta on pysynyt tuotteiden välillä yhteisenä, mitä enemmän yhteneviä ominaisuuksia alusta tuotteille tarjoaa, sitä vähemmän uutta työtä ja uutta testausta tarvitaan. Kun muutoksia tehdään vähemmän, myös laatu pysyy paremmin hallinnassa. Tästä syystä tutkimukseen on otettu mukaan myös kolmas mittari, joka mittaa katselmointimuutosten työmäärää. Katselmointimuutokset eivät kerro siitä työmäärästä, mitä tuotekohtaisia ohjelmakoodimuutoksia tai lisäyksiä on tuotteeseen on jouduttu tekemään ennen katselmointia, mutta se kertoo kuitenkin määrittelyn laadusta ja siitä, onko työtä tehty turhaan. Tuotekehitysprosessin kehitysjakson aikaisten virheiden esiintymisestä ei toimeksiantajalla ole toistaiseksi dataa saatavilla, sillä bugien kirjoittamista tuotekehitysjonoon on jopa tietoisesti pyritty vähentämään, jotta säästyttäisiin turhalta dokumentoimiselta. Virheitä on pyritty korjaamaan yhä enemmän suullisella käskynannolla, heti niiden löydyttyä. Virheiden määrää olisi kuitenkin hyvä tilastoida myös varsinaisen kehitysjakson ajalta, edes ajoittain, jotta voitaisiin seurata, kuinka paljon uusien määrittelyiden toteuttaminen keskimäärin tuo mukanaan virheitä tuotteeseen.

Läpivientiaikaa mitattaessa on käytetty liitteen 1 kuvaamia projekteja, jotka ovat pieni satunnainen joukkoa kaikista toimeksiantajaorganisaation projekteista. Liitteen 1 projektit ovat kaikki yhden ja saman tiimin työstämiä, joten velositeetin kehittyminen on puhtaasti yhden tiimin kehittymistä. Myös tuotteiden määrittelymuutosten määrää mitattaessa on käytetty satunnaista joukkoa projekteja, mutta tällöin projektilistassa on ollut tasaisesti kaikkien tuotekehitystiimien työstämiä projekteja. Laatua ja virheiden määrää

mitattaessa käytettiin tuotekehitystiimien yhteistä dataa ilmenneistä bugeista vuosien varrelta.

Lähtötilanteen kartoittamisessa käytettiin myös digitaalista haastattelulomaketta, jonka kysymykset pohjautuivat luvussa 5 esiteltyihin ennalta asetettuihin olettamiin. Haastattelussa on aina riskinä, että vastaaja ei ymmärrä kysymystä tavalla, jolla kysyjä olisi sen halunnut tulla ymmärretyksi. Tästä syystä tilastoitaville kysymyksille on tarkentavana kysymyksenä kysyttyä myös vastauksen perustelua. Perustelua tulkitsemalla, voidaan olla varmempia siitä, onko kysymys ymmärretty oikein. Perusteluihin tarkoitetuilla vapaan tekstin kysymyksillä on myös ollut tarkoituksena kerätä haastateltavilta mahdollisia kehitysehdotuksia tuotekehitysprosessin ja toiminnan tehostamiseksi. Haastattelussa annetut vastaukset vahvistivat selkeästi olettamusten paikkansapitävyyden, joten sillä päästiin odotetun kaltaiseen tulokseen. Mikäli vastaukset olisivatkin poikenneet olettamuksista, olisi kuitenkin perusteluiden avulla saatu todennäköisesti riittävästi tietoa ongelmasta ja sen lähteestä.

Ongelmien löytämisessä voitaisiin käyttää myös paria nykyisin ei seurantakäytössä olevaa mittaria. Scrumista tutut tuotteen ja sprintin edistymiskaaviot, joiden avulla voidaan seurata muutosten vaikutusta ja työn edistymistä, olisi hyvä lisätä tuotekehitysprosessissa säännöllisesti seurattaviin mittareihin. Kehitystiimin päivittäessä tuotteen kokonaistyömäärä jokaisessa Scrumin mukaisessa päiväpalaverissa, voidaan mm. arvioida todennäköisyyttä sprintin tavoitteen saavuttamiselle, sekä arvioida jäljellä olevaa kokonaistyömäärää. (Pries & Quigley 2011, 48; Schwaber & Sutherland 2017, 16)

8.2 Ongelmakohtien rajaaminen

Avoimen keskustelukulttuurin vuoksi organisaatiossa ei ole ollut ongelmia ongelmakohtien löytämisessä, vaan niiden johtaneiden syiden tunnistamisessa ja ratkaisemisessa. Organisaatio on mitannut eri osastojen projektiin käyttämiä aikoja päivätarkkuudella aina myynnistä käyttöönottoon asti, joten projektin läpivientiin käytetyt kokonaisajat eri osastoilta on ollut organisaatiolla tiedossa ja helposti saatavilla. Tämän kehittämishankkeen yhteydessä tuloksia on tarkennettu tuomalla esiin mittaustuloksia myös projektin työstön ajalta (ks. liitteen 1 väliaikamittaukset, kuten odotukseen käytetyt ajat, jne.). Mitatut tulokset ovat olleet koostamatta, eikä niitä ole hyödynnetty tuotekehitysprosessia tutkittaessa tai parannettaessa. Organisaation yleiset mittarit eivät ole kertoneet, missä kohtaa kutakin prosessia pullonkaulat alkavat muodostua, vaikka ovatkin

kielineet jonkinlaisesta hävikistä prosessin aikana. Tulosten mukaan, tuotekehitysprosessissa projektin alkuvaihe saattaa mennä todella vauhdikkaasti, mutta yleensä loppuvaiheessa edistyminen hidastuu radikaalisti. Mitattaessa vain prosessin läpimenoaikaa, ei olla saatu tietoa siitä, missä prosessin vaiheessa prosessia ja mitä toimenpiteitä tehtäessä läpimeno on alkanut hidastumaan. Tässä kehittämishankkeessa tehdyn tutkimuksen mukaan voidaan osoittaa, että läpimeno hidastuu merkittävästi toimeksiantajan projektien loppuvaiheessa, ns. projektin häntää työstettäessä.

Luvuissa 5 ja 6 on käsitelty haastattelussa annettuja vastauksia. Tutkimus vahvisti käsitystä olettamuksista, mutta tarkensi myös ongelmien juurisyitä. Vaikuttavimmat tunnistetut ongelmat olivat useat yhtäaikaiset projektit, tuoteomistajan puuttuminen, sekä edellä mainituista johtuva odottelu, jonka vuoksi projekteja aloitetaan vain enemmän ja enemmän.

8.3 Tulosten luotettavuus

Haastattelututkimus asetettuja olettamuksia vasten antoi odotetun kaltaisen tuloksen ja vahvisti käsitystä siitä, missä suurimpien ongelmien oletettiinkin olevan.

Haastattelussa annetuista perusteluista voidaan huomata, että kaikki eivät ole ymmärtäneet tilastoitavia kysymyksiä samalla tavalla. Esimerkiksi tarkentavat kysymykset eri sprintti palaverien tärkeydestä on voitu tulkita siten, että tuoteomistajat ovat voineet vastata kokevansa hyödyttömiksi palaverit, joissa ei käsitellä heidän tuotteitaan, vaikka haastattelukysymyksellä odotettiinkin vastausta nimenomaan vain niistä palavereista, joissa tuoteomistajan projekteja käsitellään. Sprinttipalaverien tärkeyttä käsittelevillä kysymyksillä haettiin vastausta siihen, ymmärretäänkö niiden tärkeys nykyisessä tuotekehitysprosessissa. Odotusarvona oli, että kehittäjät vastaavat palaverien olevan erittäin tärkeitä, koska he kaipaavat lisää tukea tuotteen määrittelyiden tulkintaan ja että tuoteomistajat vahvistaisivat, että eivät syystä tai toisesta voi palavereihin toivotulla panoksella osallistumaan. Vastaukset perusteluineen olivatkin keskimäärin odotetun kaltaiset, vaikka kysymys olisikin ymmärretty hieman väärin.

Tulkittaessa vastauksia kysymyksistä liittyen tuotteen ja määrittelyn laatuun, tulee pohtia, mistä kulmasta vastaaja asiaa tarkastelee. Keskityttäessä tuotteen ominaisuuksien laatuun, kehittäjät olivat laadun suhteen kriittisempiä kuin tuoteomistajat. Tuoteomistajat ovat tyytyväisiä, kun ominaisuus toimii määritellyn kaltaisena, mutta kehittäjälle se ei

välttämättä riitä, vaan kehittäjä arvostelee tuotosten uudelleen käytettävyyttä, tehokkuutta, sekä arvostaa toteutusten selkeyttä ja tuotteiden yhdenmukaisuutta. Haastattelun vastaukset osoittavat, että ajatusmaailmassa on ristiriita: Määrittelijät toivovat pystyvänsä määrittelemään uusia ominaisuuksia jotka palvelevat mahdollisimman tehokkaasti asiakasta, kun taas kehittäjät pyrkivät pitämään muutosten määrän mahdollisimman pienenä ja tuotteen mahdollisimman yhdenmukaisina. Ongelmana voidaan todeta olevan mm. ennalta asetetut aikataulut, jolloin uuden tekemiseen ja laadun varmistamiseen ei jää tarpeeksi aikaa. Tällöin jousto tapahtuu määrittelyssä ja joitain ominaisuuksia joudutaan jättämään pois. Kysymykset liittyen määrittelyn laatuun voivat puolestaan olla väärin ymmärrettyjä mm. siksi, että ketterää kehitystä suoritettaessa, ennalta lukittua määrittelydokumenttia ei toivota olevan lainkaan, vaan määrittelyä tulisi tehdä itse tuotekehityksen lomassa. Mikäli vastaaja ajattelee että tuotekehitysprosessia ei tule enempää ketteröittää, hän toivonee tarkempia määrittelyitä, kun taas ketteryyttä kannattavalle vastaajalle kelpaa varsin hyvin myös vähäisempi määrittely. Koska hajonta ei ole iso ja vastaukset perusteluineen ovat yhdenmukaisia muiden kysymysten vastausten kanssa, voidaan todeta, että määrittelytyössä kuitenkin on parannettavaa: Joko tulee tehdä tarkempia määrittelyitä, jolloin määrittelyä ja siihen liittyvää tuen antoa ei tarvita yhtä paljon osaksi tuotekehitystä tai sitten tuotekehitysprosessia tulee tukea paljon enemmän jatkuvalla suunnittelulla.

Tulokset vaikuttavat olevan yhdenmukaiset myös organisaation aiemmin tehtyjen tutkimusten kanssa. Organisaatiossa on ollut tapana pitää kehityskeskustelut vähintään kerran vuodessa. Positiivisena palautteena jokaisen tiiminjäsenen kehityskeskusteluista on jäänyt kehut ja kiitokset tuotekehitystiimin jäsenten keskinäisestä kemiasta ja kommunikoinnista. Työtehtävät koetaan jakautuvan tasaisesti ja oma työ on koettu suurimmaksi osaksi mielekkääksi. Ongelmalliseksi kehityskeskusteluissa on koettu kommunikointi toisten toimintojen kanssa, mikä tuotekehitystiimillä tarkoittaa siis lähinnä yhteyttä tuotteen määritteleviin ja tuotteesta vastuussa oleviin tuotepäälliköihin, sekä julkaisutiimiin. Ongelmaksi on koettu myös jatkuva kiire. Samanlaisen tuloksen on antanut myös organisaation teettämä Parempi työyhteisö -kysely, jossa on kysytty mm. työyhteisön toimivuudesta ja työryhmien välisestä yhteistyöstä. Parempi työyhteisö -kysely on teetetty yrityksen koko henkilökunnalle ja sen vastausprosentti on ollut 71%.

Anonyymius pienessä organisaatiossa on varsin läpinäkyvää, mikä saattaa vaikuttaa tulosten luotettavuuteen negatiivisesti. Vastausten antaja on helposti rajattavissa ja tunnistettavissa, mikä saattaa karsia tuloksista jyrkimmät mielipiteet. Organisaatio on

kulttuuriltaan avoin kaikenlaisille kehityskeskustelulle, mikä puolestaan lieventää kynnystä tuoda oma mielipide esille. Haastattelun valintakysymykset saattavat myös olla hieman johdattelevia. Johdattelevuutta on kuitenkin pyritty lieventämään perusteluun tarkoitetuilla vapaan sanan lomakekentillä. Haastattelututkimukseen osallistui yhteensä 17 henkilöä, jotka olivat tutkitulta alueelta useista tuotekehitystiimeistä. Haastattelulomake jaettiin 25 henkilölle, joten vastaukset kattoivat yli kaksi kolmasosaa alueen henkilöstöstä.

9 YHTEENVETO

Tämä kehittämishanke toimi Turun ammattikorkeakoulussa suoritettavan ylemmän korkeakoulututkinnon opinnäytetyönä, Teknologiaosaamisen johtamisen -koulutusohjelmasta. Kehittämishankkeen tavoitteena oli terveydenhuollon tietojärjestelmätoimittajana toimivan toimeksiantajan tuotekehitysprosessin merkittävä nopeuttaminen, työkuormaa pienentämällä sekä prosessia tehostamalla. Hankkeeseen kuului nykyisen tuotekehitysprosessin ongelmien kartoittaminen, sekä ongelmakohtien kehittäminen. Nykyisen tuotekehitysprosessin ongelmakohtia tutkittiin kolmella mittarilla, sekä digitaalisella haastattelulomakkeella. Mittareina käytettiin projektien läpivientiaikaa, tuotteiden laatua, sekä hyväksyntäkatselmoinnin jälkeen tehtävien muutosten määrää. Läpivientiaikoja tutkimalla voitiin todeta, että odotus luo eniten hävikkiä tuotekehitysprosessin läpiviennissä. Tutkimuksen rajaamiseksi, haastattelu suoritettiin tiettyjä ennalta asetettuja olettamia vasten. Haastattelua käytettiin tutkimuksessa keinona vahvistamaan oletettujen ongelmien olemassaolo ja niiden laajuus, mutta myös keräämään kehitysehdotuksia tutkittavan tuotekehitysprosessin parissa työskenteleviltä.

Haastattelun avulla haettiin varmennusta erityisesti seuraaville neljälle ennalta asetetulle olettamalle:

1. Tuotekehityksen ketterät menetelmät koetaan kyllä tarpeellisiksi, mutta niitä ei tehdä akateemisesti oikein.
2. Kaikki eivät ymmärrä tuotekehityksen menetelmiä, niiden käyttötarkoitusta tai niiden tärkeyttä.
3. Odotus on tuotekehityksen kannalta suurin hävikkiä tuottava ongelma.
4. Suurin odotusta aiheuttava ongelma on, että asiakkaan tahtotilaa ei saada tuotua tuotekehitykselle asti niin, että tuotekehitys ymmärtäisi sen

Vastauksista voidaan todeta, että kaikkiin oletettuihin ongelmiin saatiin myös varmistus. Haastattelun tulokset kertovat, että kaikki kyselyyn osallistuneet kehittäjät kokevat asiakkaan tahtotilan ymmärtämättömyyden suurimmaksi odotusta aiheuttavaksi asiaksi. Toisena paljon odotusta aiheuttavana tekijänä esille tuotiin tuotekehitystiimiä auttavien eri roolien prioriteettien kohtaamattomuus tuotekehityksen prioriteettien kanssa. Kiire ja tuoteomistajan hankala tavoittaminen tai täydellinen puuttuminen tuotekehitysprosessista, koettiin projektin aikataulun ja tuotteen laadun kannalta erityisen ongelmalliseksi.

Yhtenä merkittävänä haasteena tuotekehitysprosessissa todettiin asiakkaan tai loppukäyttäjän hankala tavoittaminen, mikä tekee myös mahdottomaksi asiakkaan sitouttamisen mukaan tuotekehitysprosessiin. Asiakasta paikataan tuotekehitysprosessissa sisäisellä tuoteomistajan roolilla, jossa toimivat kunkin tuotteen määrittelijät. Toisena suurena haasteena koettiin jatkuva tarve muutoksille. Erityisesti näiden haasteiden vuoksi, tuotekehitysprosessin työstämisessä käytetään ns. hybridimallia, joka yhdistää ketterän kehityksen menetelmiä osaksi perinteisiä ohjelmistokehitysmenetelmiä. Tuotekehitysprosessiin tarvittaisiin ketterämpää toimintaa, jossa on vähemmän lukittuja ominaisuuksien määrittelyitä ja enemmän itse määrittelyä osana tuotekehitysprosessia, jolloin myös tuoteomistajana toimivat määrittelijät saataisiin sitoutettua suurempaan rooliin tuotekehityksessä.

Toimeksiantajaorganisaatiossa on käyttöönotettu Scrum-viitekehys ja DevOps -työkaluja osaksi tuotekehitystä, mutta organisaation laajuinen kulttuuri ketterän kehityksen ja DevOpsin tukemiseen puuttuu. Kulttuurin uupuminen tulee tekemään myös uuden tehostetun tuotekehitysprosessin jalkauttamisesta hidasta ja hankalaa. Jotta ketterä kehitys toimisi, tulisi koko organisaation olla ketterä. Koko organisaation tulisi jakaa yhteiset arvot ja jakaa vastuuta yhdessä tekemisestä. Kun koko henkilöstö jakaa vastuun järjestelmän liiketoiminnallisista tavoitteista, kaikki ymmärtävät tiiviimmän yhteistyön kautta paremmin järjestelmän toiminnalliset tarpeet ja voivat auttaa vastaamaan niihin.

Päämääränä oli nopeuttaa tuotekehitysprosessia jopa 30%. Esitetyt parannus ehdotusten pitäisi tehostaa prosessia työssä esitettyyjen tutkimusten pohjalta useita kymmeniä prosentteja, mutta todellinen tulos voidaan mitata vasta, kun uusi tehostettu prosessi otetaan käyttöön ja uudet tulokset mitataan. Tutkimuksen tuloksia voidaan hyödyntää erityisesti työssä esitetyn tehostetun tuotekehitysprosessin käyttöönoton yhteydessä tai parhaita jalkautuskeinoja tutkiessa, mutta myös toimeksiantajaorganisaation ulkopuolella kaikessa ohjelmistotuotekehityksessä terveydenhuollon alalla. Koska puhtaalla vesiputousmallilla työskenteleminen terveydenhuollon alalla on tuotekehityksen näkökulmasta liian kankeaa, tämän hankkeen tutkimustyötä voitaisiin täydentää mm. seuraavilla tutkimuksilla:

1. Miten terveydenhuoltoalan ohjelmistokehityksen ostajat ja ohjelmien loppukäyttäjät saataisiin sitoutettua paremmin osaksi ketterää tuotekehitystä.
2. Miten määrittelyprosessia kannattaisi tehostaa, jotta tuotekehitys saisi mahdollisimman paljon tukea suunnitteluun.

3. Millaisia automaatioita ja DevOps-työkaluja tuotekehityksessä voitaisiin käyttää prosessia tehostettaessa.
4. Miten laadun varmistuksesta saataisiin sulavampaa ja tehokkaampaa, jotta testausautomaatiolla saataisiin tehostettua tuotekehitysprosessia.

Tutkimuksen tekeminen on ollut mielenkiintoista ja erityisesti tuotekehityksen päivittäistä tekemistä koskettavaa, vaikkakin työympäristön vastahakoinen kulttuuri on hankaloittanut aiheesta keskustelua ja kehitysehdotusten koettamista käytännössä. Tutkimuksen tulosten perusteella, oikein tehtynä ketterällä kehityksellä päästäisiin tämän työn kuvaamassa työympäristössä huomattavasti parempiin tuloksiin kuin perinteisiä tai itse räätälöityjä menetelmiä käyttäen. Haasteeksi muodostuvat ongelmat uuden prosessin käyttöönottamisessa, sillä kulttuurimuutos vaatii paljon työtä. Jatkuvan parantamisen kautta sekä yhdessä tekemällä, niin tuotekehitysprosessia kuin muitakin organisaation prosesseja, sekä sidosryhmien hyödyntämistä, saataisiin tehostettua huomattavan paljon. Kulttuurin muutoksella todennäköisesti olisi vaikutusta myös työilmapiirin parantumiseen ja mielekkyyden lisääntymiseen.

LÄHTEET

1. Agile Alliance. 2001. Agile Manifesto. Viitattu 1.12.2018. www.agilealliance.org > agile101 > the agile manifesto.
2. Atlassian 2018. Continuous delivery. Continuous integration vs. continuous delivery vs. continuous deployment. Viitattu 3.12.2018. https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd.
3. Auer, A. 2013. Ketterää kehitystä. Oy Finn Lectura Ab.
4. Braude, E. Bernstein, M. 2011. Software Engineering: Modern Approaches. Wave-land Press, Inc.
5. Brooks, F. 1995. The Mythical Man-Month. Addison Wesley Longman, Inc
6. Canty, D. 2015. Agile for Project Managers. Auerbach Publications.
7. Chin, G. 2003. Agile Project Management: How to Succeed in the face of Changing Project Requirements.
8. Cobb, C. 2015. The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach. John Wiley & Sons, Inc.
9. Cohn, M. 2006. Agile Estimating and Planning, Pearson Education, Inc.
10. Coplien, J. & Bjørnvig, G. 2010. Lean Architecture: For Agile Software Development. John Wiley & Sons, Inc.
11. Crosby, P. 1979. Quality is Free. McGraw-Hill.
12. Cunningham, L. Viitattu 1.1.2019. https://www.agilealliance.org/8-reasons-why-agile-projects-fail/.
13. Cunningham, L. 2015. 8 Reasons Why Agile Projects Fail. Viitattu 18.12.2018. https://www.agilealliance.org/8-reasons-why-agile-projects-fail/.
14. Davis, B. 2012. Agile Practices for Waterfall Projects: Shifting Processes for Competitive Advantage. J. Ross Publishing.
15. Forsgren, N.; Humble, J. & Kim, G. 2018. Accelerate: The Science of Lean Software and DevOps: Buildin and Scaling High Performing Technology Organizations.
16. Haverila M., Uusi-Rauva E., Kouri I., Miettinen A. 2009. Teollisuustalous. Tampere: Infacts Oy.
17. Hughes, R. 2013. Agile Data Warehousing Project Management: Business Intelli-gence Systems Using Scrum. Elsevier Inc.
18. Järvenpää, J. Kovanen, P. 2018. Ohjelmistokehityksen ostajan pikaopas 2.0. Tam-pere: Eräsalon Kirjapaino Oy
19. Layton, M. & Ostermiller, S. 2017. Agile Project Management For Dummies. John Wiley & Sons, Inc.
20. Leach, L. 2014. Critical Chain Project Management. Artec House.

21. Lähteenmäki, I. 2006. Tekonivelpotilaan seurantajärjestelmän käyttöönotto kirurgi-sessa sairaalassa. Satakunnan ammattikorkeakoulu.

22. Mäkelä, K. 2006. Terveydenhuollon tietotekniikka: Terveyden ja hyvinvoinnin sovel-lukset. Helsinki: Talentum

23. Poppendieck, M. & Poppendieck, T. 2007. Implementing Lean Software Develop-ment: From Concept to Cash. Addison Wesley.

24. Pries, K. & Quigley, J. 2011. Scrum Project Management. CRC Press.

25. Royce, W. 1970. Managing the Development of Large Software Sys-tems. Proceedings of IEEE WESCON.

26. Ruuska, K. 1999. Projekti hallintaan. Jyväskylä: Gummerus Kirjapaino Oy.

27. Schwaber, K. & Sutherland, J. 2017. Scrum Guide. Viitattu 30.1.2019. https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Finnish.pdf.

28. Shestakova, M. 2014. Why Agile Doesn't Work or 12 Fatal Mistakes. Viitattu 1.1.2019. https://r-stylelab.com/company/blog/web-development/12-fatal-mistakes-in-agile-development.

29. Stettina, C. & Smit, M. 2016. Team Portfolio Scrum: An Action Research on Multi-tasking in Multi-project Scrum Teams.

30. Swartout, P. 2012. Continuous Delivery and DevOps: A Quickstart guide. Packt Pub-lishing Ltd.

31. Terveyden ja hyvinvoinninlaitos (THL). 2018. Terveydenhuollon tietojärjestelmät ovat entistä älykkäämpiä – auttavat jopa diagnoosien tekemisessä. Viitattu 4.12.2018. https://thl.fi/fi/-/terveydenhuollon-tietojarjestelmat-ovat-entista-alykkaampia-auttavat-jopa-diagnoosin-tekemisessa.

32. Toikkanen, T.; Auer, A.; Auer, L.; Heinäsmäki, M.; Höttilä, J.; Kalliala, E.; Laanti, M.; Laine, K.; Lekman, L.; Miinalainen, P.; Naski, H.; Piiparinen, T.; Puhakka, H.; Pyhä-järvi, M.; Pääkkönen, T.; Rosti, J.; Räisänen, S.; Sora, H.; Taipale, M.; Talvio, J.; Tanninen, A.; Toivola, T.; Toro, K.; Valsta, A.; Väyrynen, V.; Weissenberg, M. 2013. 398 vuotta ketteriä kokemuksia. Espoo: Systeemityöyhdistys Sytyke.

33. Vadapalli, S. DevOps: Continuous Delivery, Integration and Deployment with Devops: Dive into the core DevOps strategies. Packt Publishing.

34. Vanderjack, B. 2015. The Agile Edge: Managing Projects Effectively Using Agile Scrum. Business Expert Press.

35. VersionOne. 2015. 9th annual state of agile report. Viitattu 18.12.2018. https://explore.versionone.com/state-of-agile/9th-annual-state-of-agile-report-2.

36. VersionOne. 2018. 12th annual state of agile report. Viitattu 18.12.2018. https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report-.

37. Värpiö V. 2017. Sisäisen laadun kehittäminen suurivolyymisessa tuotannossa. Tampe-re.

38. Wells, K. & Kloppenborg, T. 2015. Project Management Essentials. Business Expert Press, LLC.

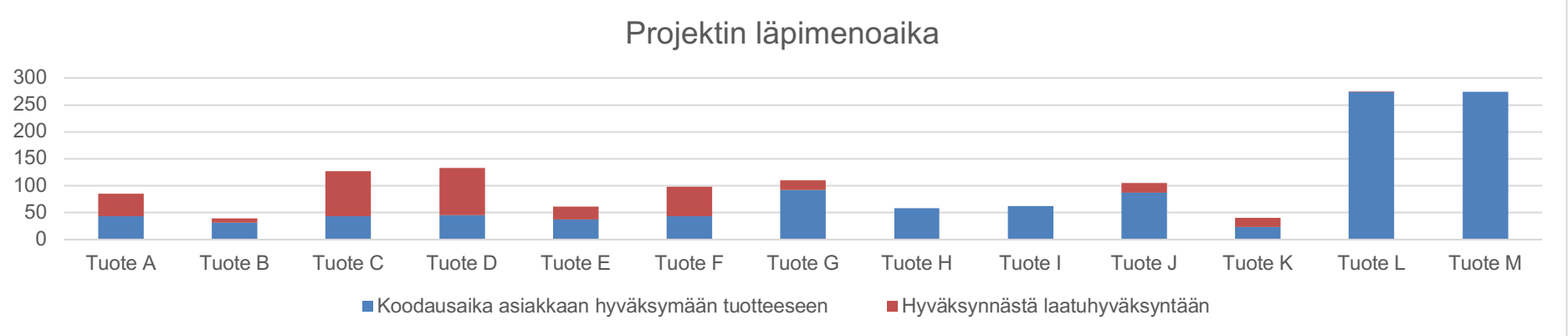
39. Wikimedia Commons. 2009. Scrum Process. Viitattu 29.1.2019. https://upload.wikimedia.org/wikipedia/commons/5/58/Scrum_process.svg.

40. Wikimedia Commons. 2013. Kanban Principles. Viitattu 29.1.2019. https://upload.wikimedia.org/wikipedia/commons/a/ab/Kanban_principles.svg.

41. Wysocki, R. 2013. Effective Project Management: Traditional, Agile, Extreme. John Wiley & Sons, Inc.

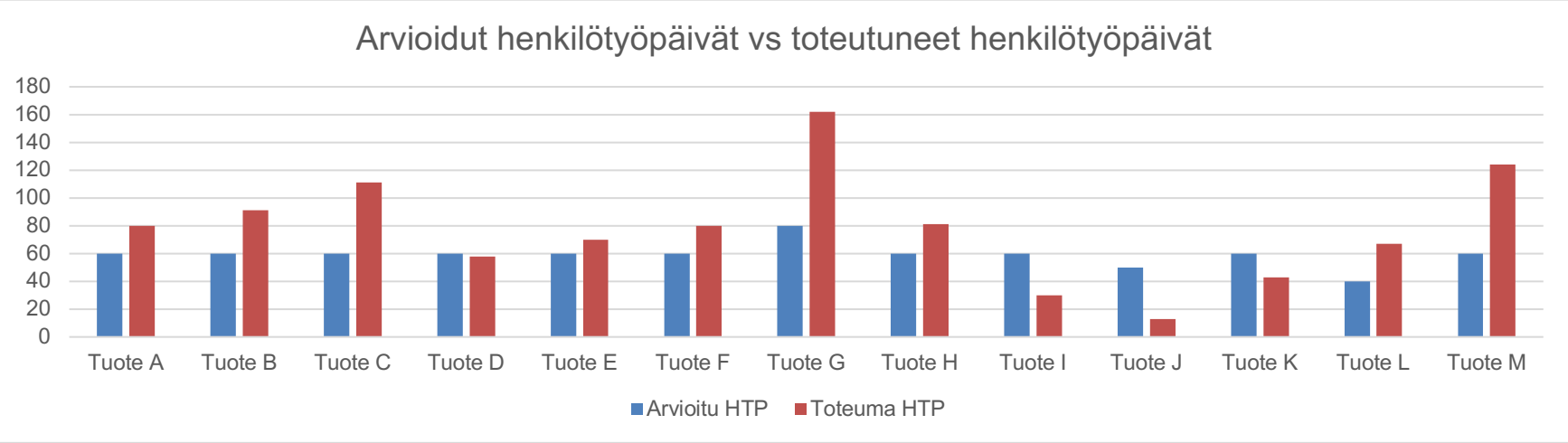
Läpimenoajat

Vaihe	Tuote A	Tuote B	Tuote C	Tuote D	Tuote E	Tuote F	Tuote G	Tuote H	Tuote I	Tuote J	Tuote K	Tuote L	Tuote M
Kickoff / tuotten esittely	13.7.2018	28.5.2018	8.5.2018	9.4.2018	19.3.2018	29.1.2018	17.11.2017	29.9.2017	3.7.2017	8.5.2017	20.9.2017	20.12.2016	20.12.2016
Koodaamisen aloittaminen	16.7.2018	29.5.2018	8.5.2018	9.4.2018	19.3.2018	13.2.2018	27.11.2017	16.10.2017	19.7.2017	26.5.2017	13.9.2017	20.12.2016	20.12.2016
Katselmoitava versio valmis	20.8.2018	18.9.2018	2.10.2018	7.5.2018	12.4.2018	29.3.2018	15.2.2018	24.11.2017	14.9.2017	29.6.2017	5.10.2017	20.9.2017	18.9.2017
Hyväksyntäkatselmointi	28.8.2018	29.6.2018	20.6.2018	24.5.2018	25.4.2018	28.3.2018	27.2.2018	13.12.2017	19.9.2017	21.8.2017	6.10.2017	20.9.2017	20.9.2017
Hyväksyntäkatselmointimuutokset valmiit	11.9.2018	11.9.2018	13.7.2018	28.5.2018	4.5.2018	28.3.2018	8.3.2018	9.1.2018	19.9.2017	7.9.2017	9.10.2017	20.9.2017	20.9.2017
Asiakashyväksyntä	28.8.2018	29.6.2018	20.6.2018	24.5.2018	25.4.2018	28.3.2018	27.2.2018	13.12.2017	19.9.2017	29.8.2017	6.10.2017	20.9.2017	20.9.2017
Laatuhyväksyntä	23.10.2018	19.9.2018	5.10.2018	24.8.2018	28.5.2018	22.5.2018	26.3.2018	9.1.2018	19.9.2017	25.9.2017	26.10.2017	21.9.2017	20.9.2017
Julkaisu	23.10.2018	12.10.2018	9.10.2018	27.9.2018	20.8.2018	15.8.2018	22.4.2018	23.2.2018	29.9.2017	4.10.2017	28.10.2017	26.9.2017	26.9.2017
Päivät yhteensä	**102**	**137**	**154**	**171**	**154**	**198**	**156**	**147**	**88**	**149**	**38**	**280**	**280**
Koodausaika	**35**	**112**	**147**	**28**	**24**	**44**	**80**	**39**	**57**	**34**	**22**	**274**	**272**
Koodausaika asiakkaan hyväksymään tuotteeseen	**43**	**31**	**43**	**45**	**37**	**43**	**92**	**58**	**62**	**87**	**23**	**274**	**274**
Hyväksynnästä laatuhyväksyntään	**42**	**8**	**84**	**88**	**24**	**55**	**18**	**0**	**0**	**18**	**17**	**1**	**0**
Laatuhyväksynnästä julkaisuun	**0**	**23**	**4**	**34**	**84**	**85**	**27**	**45**	**10**	**9**	**2**	**5**	**6**



Projektin arvioidut henkilötyöpäivät suhteessa projektin toteutuneisiin henkilötyöpäiviin

Nimi	Arvio HTP	Toteuma HTP	Kuvaus
Tuote A	60	80	Uudenlainen lähestyminen. Julkinen etäkäyttöliittymä. Laaja sisältö.
Tuote B	60	91	Uusi erikoisala. Uusia integraatioita ja erittäin vaikeat kuvaajat
Tuote C	60	111	Uusi erikoisala, monimutkaiset kustomoidut lomakkeet. Uusia kuvaajia.
Tuote D	60	58	Kolmas tuote tälle erikoisalalle. Paljon yhtenäistä koodia. Laajin kolmesta
Tuote E	60	70	Toinen tuote tälle erikoisalalle. Paljon yhtenäistä koodia. Hieman edellistä isompi
Tuote F	60	80	Uusi erikoisala, jossa paljon uutta uudelleen käytettävää koodipohjaa
Tuote G	80	162	Julkinen käyttöliittymä. Paljon sisäistä kehitystä vaativa
Tuote H	60	81	Laaja kohdevalikoima - paljon kustomointia.
Tuote I	60	30	Yksinkertainen
Tuote J	50	13	Yksinkertainen ja suoraviivainen
Tuote K	60	43	Suoraviivainen ja simppeli. Oma pieni erikoisalansa
Tuote L	60	124	Iso pohjakoodin päivitys. Laaja käyttöönotto
Tuote M	40	67	Iso pohjakoodin päivitys. Laaja käyttöönotto. Kopio



HAASTATTELULOMAKE

1. Vastaaja (rooli)

- Kehittäjä
- Tuotepäällikkö / määrittelijä
- Testaaja

2. Mikä asia vaikuttaa mielestäsi eniten tuotekehitysprojektin aikataulun venymiseen?

3. Mikä asia vaikuttaa mielestäsi eniten tuotekehitysprojektin lopputuotoksen laatuun?

4. Mikä asia mielestäsi vaikuttaa eniten projektin ominaisuuksiin ja toimintoihin (määrä, laatu)?

5. Onko asiakkaan tahtotilan tuotekehitykselle asti saaminen suurin odotusta aiheuttava ongelma?

** Ketterät kehitysmenetelmät edellyttävät että tuotekehitys ymmärtää tehtävän. ** Odotus on yksi Lean filosofian hukkatyypeistä.*

- Kyllä
- Ei

6. Mikä on mielestäsi suurin odotusta aiheuttava ongelma?

Vastataan mikäli valitsit kysymyksessä 5 vaihtoehdon "Ei"

7. Kuinka poistaisit tuotekehitysprojektissa odotusta aiheuttavan ongelman?

8. Mikä estää poistamasta odotusta aiheuttavaa ongelman?

9. Saatko TUOTEPÄÄLLIKÖILTÄ riittävästi tukea ja tietoa työhösi?

- Aina tai lähes aina

- Useimmiten
- Harvemmin

10. Saatko TUOTEKEHITYSTIIMILTÄ riittävästi tukea ja tietoa työhösi?

- Aina tai lähes aina
- Useimmiten
- Harvemmin

11. Saatko TESTAAJILTA riittävästi tukea ja tietoa työhösi?

- Aina tai lähes aina
- Useimmiten
- Harvemmin

12. Kuinka selkeää eri roolien / toimijoiden välinen tehtävienjako mielestäsi on?

Toimijat: tuotepäällikkö, ohjelmoija, testaaja, käyttöönottaja, asiakasvastaava, integraatiospesialisti

- Erittäin selkeää
- Selkeää
- Sekavaa
- Erittäin sekavaa

13. Onko tuotteen ominaisuuksien määrittelyn laatu, selkeys ja sisältö mielestäsi keskimäärin

- Kiitettävää
- Hyvää
- Tyydyttävää
- Välttävää

14. Perustele antamasi vastaus määrittelyn laadusta

Lyhyt perustelu ja parannusehdotus

15. Onko toteutettujen ominaisuuksien laatu mielestäsi keskimäärin

- Kiitettävää
- Hyvää
- Tyydyttävää
- Välttävää

16. Perustele antamasi vastaus toteutettujen ominaisuuksien laadusta

Lyhyt perustelu ja parannusehdotus

17. Koetko tuotekehityssprintin ALOITUSPALAVERIN hyödylliseksi tapahtumaksi?

Sprintin aloituspalaverissa tuotekehitystiimi ja tuotepäällikkö sopivat yhdessä alkavan sprintin tehtävät.

- Aina
- Useimmiten
- Harvoin
- En juuri koskaan

18. Perustele ALOITUSPALAVERIN hyödyllisyydestä antamasi vastaus

Lyhyt perustelu. Mitä kehittäisit tapahtumassa?

19. Koetko tuotekehityssprintin PÄÄTÖSPALAVERIN hyödylliseksi tapahtumaksi?

Sprintin päätöspalaverissa tuotekehitystiimi esittelee tuotepäällikölle (ja muulle yleisölle) sprintin aikana julkaisemansa tuotoksen

- Aina
- Useimmiten
- Harvoin
- En juuri koskaan

20. Perustele PÄÄTÖSPALAVERIN hyödyllisyydestä antamasi vastaus

Lyhyt perustelu. Mitä kehittäisit tapahtumassa?

21. Koetko tuotekehityssprintin RETROSPEKTIIVIN hyödylliseksi tapahtumaksi?

Sprintin retrospektiivissä tarkastellaan prosessin näkökulmasta mikä sprintin aikana sujui hyvin ja mitä voitaisiin parantaa seuraavassa sprintissä.

- Aina
- Useimmiten
- Harvoin
- En juuri koskaan

22. Perustele RETROSPEKTIIVIN hyödyllisyydestä antamasi vastaus

Lyhyt perustelu. Mitä kehittäisit tapahtumassa?

23. Koetko tuotekehityssprintin GROOMING-tapahtumat hyödyllisiksi tapahtumiksi?

Grooming-tapahtuma on tuotteen kehitysjonon työstöä. Grooming tarkoittaa yksityiskohtien, työmääräarvioiden ja kehitysjonon kohtien keskinäisen järjestyksen lisäämistä.

- Aina
- Useimmiten
- Harvoin
- En juuri koskaan

24. Perustele GROOMINGIN hyödyllisyydestä antamasi vastaus

Lyhyt perustelu. Mitä kehittäisit tapahtumassa?

HAASTATTELUVASTAUKSET

1. Vastaaja (rooli)

Vaihtoehdot: Testaaja / Kehittäjä / Tuotepäällikkö

Vastaaja	Vastaus
Testaaja 1	Testaaja
Tuotepäällikkö 1	Tuotepäällikkö / määrittelijä
Kehittäjä 1	Kehittäjä
Tuotepäällikkö 2	Tuotepäällikkö / määrittelijä
Kehittäjä 2	Kehittäjä
Kehittäjä 3	Kehittäjä
Kehittäjä 4	Kehittäjä
Kehittäjä 5	Kehittäjä
Kehittäjä 6	Kehittäjä
Testaaja 2	Testaaja
Kehittäjä 7	Kehittäjä
Kehittäjä 8	Kehittäjä
Testaaja 3	Testaaja
Testaaja 4	Testaaja
Tuotepäällikkö 3	Tuotepäällikkö / määrittelijä
Testaaja 5	Testaaja
Tuotepäällikkö 4	Tuotepäällikkö / määrittelijä

2. Mikä asia vaikuttaa mielestäsi eniten tuotekehitysprojektin aikataulun venymiseen?

Vastaaja	Vastaus
Testaaja 1	Määrittelyn muutokset kehityksen loppuvaiheessa ja muutosten vaillinainen kirjaaminen määrittelyyn.
Tuotepäällikkö 1	Tuotekehitysprosessia ajatellen ihan alusta lähtien ensimäinen ongelma on se, että myynti myy yli 60htp:n [ohjelmia]. Toisena ongelmana on määrittelyprojektin kesto, joka myydään määrittelyprojektina. Määrittelyn kestoa ei ole millään tavalla rajattu. Kolmantena tulee tuotteen rajaus. Tuotteiden työmäärä on harvoin alle tai yhtä suuri kuin 60 htp.
Kehittäjä 1	Toteutustyö on vie aikaa koska käyttämämme framework ja arkkitehtuuri on keskeneräinen, mahdollistaa helpot virheet.
Tuotepäällikkö 2	Liian tiukka aikataulu tuotekehityksen eri osa-alueilla. Mitään osa-aluetta ei saada tehtyä kunnolla. Tuotetta ei katselmoida riittävästi, dokumentaatiota ei saa luovuttaa asiakkaille hyväksyttäväksi/ tutustuttuvaksi, jolloin [tuotemäärittelyyn] jää virheitä/ asioita joita asiakas ei ole ajatellut loppuun saakka.
Kehittäjä 2	Epäselvät tehtävät ja niihin vastauksen odottelu.
Kehittäjä 3	Määrittelyiden heikko taso. Hinkatataan edestakaisin yksinkertaisia asioita.

Kehittäjä 4	Eniten vaikuttaa mielestäni epäselvyydet määrittelyssä. Projektin alussa kaikki tarvittavat määritykset eivät ole valmiita. Määrittelyjä voisi tarkentaa esim. grooming-palavereissa, mutta näihin ei tunnu löytyvän aikaa. Syyn voi olla se, että tuotekehitysprojektin aikana mm. määrittelijät ovat jo määrittelemässä muita tuotteita. Puutteet määrittelyssä johtavat siihen, että kehittäjä joutuu tulkitsemaan mitä halutaan, testaaja testaa tuotteen oman näkemyksensä mukaan ja tästä seuraa pitkä pompottelu kehittäjän ja testaajan välillä. Tämä olisi vältettävissä jos määrittelyt olisivat yksikäsitteisempiä ja ennen kaikkea valmiita ennen toteutusta.
Kehittäjä 5	Epäselvyys siitä, mitä kaikkea lopulliseen tuotteeseen kuuluu.
Kehittäjä 6	speksien puutteet
Testaaja 2	Muutokset määritykseen, myöhäisessä ajankohdassa löytyvät ongelmat ja työn keskeytyminen tärkemmillä tehtävillä.
Kehittäjä 7	Määrityksen tulkittavuus / vajaus
Kehittäjä 8	- keksitään lisää / muutetaan vaatimuksia lennosta - vanhat projektit palaavat kummittelemaan (tämä on onneksi vähentynyt) - prioriteetit vaihtuvat arvaamatta - testaajat eivät ehdi testaamaan - valmiit projektit eivät tuntemattomasta syystä löydä tietään deployment-ohjelmaan
Testaaja 3	Aikataulut eivät aina ole räätälöityjä testauksen näkökannasta katsottuna. Esimerkiksi [Tuote] määrittelyssä on suuria eroja projektien välillä ja niiden testaamiseen voi kulua aikaa reilusti kun verrataan yksinkertaisempiin säännöstöihin.
Testaaja 4	Jatkuvasti vaihtuvat prioriteetit ja tästä johtuva yleinen sähläys, kuten ongelmat pallon siirrossa. Tehdään rinnakkain kymmentä asiaa ja aivomme eivät toimi optimaalisesti.
Tuotepäällikkö 3	Aika ei välttämättä riitä aina täydelliseen määrittelyyn. Asiakas on kiireinen, eikä heillä ole aikaa paneutua kunnolla määrittelytyöhön, joka taas lisää muutostoiveita sekä väärinymärryksiä.
Testaaja 5	Tuotekehitysprojektin viimeistelyvaiheessa voidaan joltain osin todeta määrittely puutteelliseksi ja tarvitaan uusia ominaisuuksia / toimintoja.
Tuotepäällikkö 4	Resurssien yliallokointi. Kun kokoajan allokointi on 100%, ei ole varaa pieneenkään ongelmaan ilman vaikutusta tuleviin projekteihin.

3. Mikä asia vaikuttaa mielestäsi eniten tuotekehitysprojektin lopputuotoksen laatuun?

Vastaaja	Vastaus
Testaaja 1	Määrittelyn laatu, annettu aika ja tekijöiden sitoutuneisuus.
Tuotepäällikkö 1	Määrittelyn laatu, yhteistyö tuotekehityksen ja määrittelijöiden/tuotepäälliköiden kanssa sekä testauksen laatu. Jos on pakko valita yksi, niin määrittelyn laatu.
Kehittäjä 1	Rima on matalalla joka osa-alueella.
Tuotepäällikkö 2	Katselmointien puute, määrittelyt usein vanhoja, jopa 2-vuotta ennen kuin pääsee kehitykseen. Asiakas ei ole saanut dokumentaatiota jotta voisia tutustua siihen kunnolla ennen määrittelyn hyväksyntää. Kehityksen aikana usein tingitään ominaisuuksista ja asiakas ei saa halumiansa ominaisuuksia heti käyttöön. Testaus deployment vaiheessa usein riittämätöntä, joka johtaa hotfixeihin/ after addeihin.

Kehittäjä 2	Jatkuva kiire. Asioita ei ehditä suunnitella tarpeeksi kun ne on jo julkaistava.
Kehittäjä 3	Kiire
Kehittäjä 4	Kiire. Muun muassa edellisessä kysymyksessä käsiteltyjen syiden vuoksi projektin testauksessa saattaa tulla kiire. Mielestäni testaus tulisi tehdä enemmän agile-mallisesti, missä testaaja testaa kehittäjän tuotoksia heti kun ne on toteutettu. Tällä hetkellä testaus tulee liian myöhään.
Kehittäjä 5	Asiakkaalle tiedon jakaminen ja asiakkaan osallistuminen kehitysprosessin aikana.
Kehittäjä 6	testaajien työ
Testaaja 2	Kiireinen aikataulu
Kehittäjä 7	Määrityksen selkeys / määrittelijän kommentit, jos ei dokumentaatio päivity
Kehittäjä 8	- ainainen kiire, tulee helpommin hutiloitua - projektit ovat niin samanlaisia keskenään että motivaatio niiden tekemiseen laskee ja jälki huononee
Testaaja 3	Ihmisten asenteet laadun suhteen. Toiset pitävät "ehjää" toiminnallisuutta riittävänä, eivätkä piittää ulkonäöstä tai logiikasta niin paljoa. Lisäksi jotkut noudattavat määrittelyä orjallisesti käyttämättä omaa harkintakykyään.
Testaaja 4	Jos yksi asia pitää sanoa niin SPEKSIN TASO.
Tuotepäällikkö 3	Aika ja määrittely...
Testaaja 5	Tuotekehitysprojektissa toteutettavien ominaisuuksien ja toimintojen yksinkertaisuus. Yksinkertaisuus johtaa parempaan lopputuotoksen laatuun.
Tuotepäällikkö 4	Asiakkaan prosessin monimuotoisuus.

4. Mikä asia mielestäsi vaikuttaa eniten projektin ominaisuuksiin ja toimintoihin (määrä, laatu)?

Vastaaja	Vastaus
Testaaja 1	Asiakkaan toivomukset ja miten niihin määrittelijä hyväksyy kaikki toivomukset.
Tuotepäällikkö 1	Yhteistyö määrittelijöiden/tuotepäälliköiden ja tuotekehittäjien kanssa sekä aikataulu. Aikataulu vaikuttaa suoraan molempien tekijöiden asenteeseen tuotekehitysprojektia kohtaan.
Kehittäjä 1	En ymmärrä kysymystä. Mikä on kysymyksen projekti ja tuotekehitysprojekti? Vastaan silti: määrittelijän OLETTAMAT frameworkkimme ominaisuudet ja määrittelijän KUVITTELEMAT työaika-arviot.
Tuotepäällikkö 2	Uusia ominaisuuksia "ei saa" toteuttaa vaikka ne ovat usein välttämättömiä asiakkaalle. Esimerkiksi kunnon taulukkokomponenttia ei ole vieläkään tehty, nyt meillä on varmaan 10 erillistä toteutusta.
Kehittäjä 2	Iso osa ominaisuuksia saadaan valmiiksitehtynä, mutta aikataulut on projekteilla niin tiukat, että yksikin räätälöinti vaikuttaa joko projektin pitemenmiseen tai ominaisuuksien karsimiseen.
Kehittäjä 3	Mitä asiakas luulee tarvitsevansa
Kehittäjä 4	Tuotteen vaaditut ominaisuudet tulisi olla paremmin priorisoituja. Tällä hetkellä priorisointi puuttuu eli jos aikataulun kanssa tulee ongelmia, saattaa tärkeä toiminto jäädä tekemättä ja aikaa on käytetty turhaan ominaisuuteen jolla ei ole merkitystä. Olisi hyvä esimerkiksi luokitella vaaditut ominaisuudet eri luokkiin: * Toiminto, jota ilman tuotetta ei voida toimittaa * Toiminto, joka on asiakkaalle tärkeä, mutta ei ihan pakollinen * Toiminto, joka ei ole tärkeä, mutta toivottavaa saada mukaan

Kehittäjä 5	Määrittelijän selkäranka. Aikataulu on kuitenkin jokaisessa tuotteessa about sama, joten tuotteen määritelty laajuus oikeastaan samalla määrittää laadun.
Kehittäjä 6	asiakaslupaukset
Testaaja 2	Määrittely
Kehittäjä 7	Aikataulu, tuotekehityksen sisäinen kommunikaatio
Kehittäjä 8	vaikea sanoa
Testaaja 3	Tuote on yleensä yhtä hyvä kuin määrittely.
Testaaja 4	Käytettävissä oleva aika tietenkin?
Tuotepäällikkö 3	Hmm..
Testaaja 5	Määrittely (määrään) ja sen yksiselitteisyys (laatuun).
Tuotepäällikkö 4	Asiakkaan prosessin monimuotoisuus.

5. Onko asiakkaan tahtotilan tuotekehitykselle asti saaminen suurin odotusta aiheuttava ongelma?

Vaihtoehdot: Kyllä / Ei

Vastaaja	Vastaus
Testaaja 1	**Kyllä**
Tuotepäällikkö 1	Ei
Kehittäjä 1	Kyllä
Tuotepäällikkö 2	Ei
Kehittäjä 2	Kyllä
Kehittäjä 3	Kyllä
Kehittäjä 4	Kyllä
Kehittäjä 5	Kyllä
Kehittäjä 6	Kyllä
Testaaja 2	Ei
Kehittäjä 7	Kyllä
Kehittäjä 8	Kyllä
Testaaja 3	Ei
Testaaja 4	Ei
Tuotepäällikkö 3	Kyllä
Testaaja 5	Ei
Tuotepäällikkö 4	Ei

6. Mikä on mielestäsi suurin odotusta aiheuttava ongelma?

Vastaaja	Vastaus
Tuotepäällikkö 1	Asiakas ei itsekään tiedä mitä haluaa, ennenkuin saa valmiin, katselmoidun tuotteen käyttöönsä. Asiakkaiden silmät aukeavat ns. liian myöhään tuotekehitysprojektin prosessiin nähden.

Tuotepäällikkö 2	Vastauksia odotetaan tuotekehityksessä heti, eikä ymmärretä että se ei ole mahdollista. Sairaalamaailmassa lääkärit ovat kiinni erilaisissa potilaan hoidon kriittisissä toiminnoissa ja tuotteen on hyvin pieni osa hänen työssään. Usein me [organisaatiossa] emme ymmärrä tätä. Toinen väärä olettama on se että TP pystyisi aina vastaamaan kaikkeen heti. TP:llä on useita tehtäviä yrityksessä, koska palvelee organisaation eri haaroja horisontaalisesti. Yrityksessämme ei mennä puhtaasti ketterin menetelmin vaan enemmän nykyään vesiputousmallilla. Tällöin kaikkia ajatuksia ketteristä menetelmistä ei voida enään soveltaa tuotekehitysprosesiin, tämä pitäisi ymmärtää ja sisäistää tuotekehityksessä. Scrum ei toimi, koska asikas ei ole saatavilla kuten scumissa pitäisi olla. Roviolla tämä on helppoa kun asiakkaana on koko maailman teinit. Summana voisin todeta että isoin ongelma on: Kysymyksiä ei esitetä kootusti vaan tipotellen ja kysymyksiä ei osata esittää [tuotemäärittelyn] pohjalta riittävän ajoissa, vaan käynnissä olevan tuotekehityksen aikana.
Testaaja 2	Virheet ja niiden korjauksen odotteleminen. Lisäksi päivitysversioissa prioriteetit(ts. työjärjestys) muuttuvat nopeasti.
Testaaja 3	Epäselvä määrittely ja sen tulkitseminen ensin kehittäjän puolesta ja sen jälkeen testaajan.
Testaaja 4	Kenen kannalta? Yleisesti meidän "[organisaation] tilauskanta" jos tilauksesta lopulliseen toimitukseen menee jopa vuosia. Testaajan kannalta suurta odottelua tapahtuu mm. kun tuote siirtyy lopulta toimitusvaiheeseen ja toisaalta lopullista asiakashyväksyntääkin odotellaan joskus kovasti. Kysymykseen 5 voisi siis kyllä vastata myös "Kyllä".
Testaaja 5	Tuotekehitysprojektiin osallistuvilla tahoilla ei ole välttämättä samat prioriteetit.
Tuotepääkkiö 4	Eri tiimien (koodaus, integraatiot, testaus) välinen yhteistyö / aikataulujen synkronointi.

7. Kuinka poistaisit tuotekehitysprojektissa odotusta aiheuttavan ongelman?

Vastaaja	Vastaus
Testaaja 1	Tiivistämällä kehittävän tiimin, määrittelijän ja asiakkaan yhteistyötä
Tuotepäällikkö 1	Määrittelyaikataulu lukittaisiin (määrittelytyö on myytävä, kiinteä htp), kun määrittely on tehty, koodataan ketterästi (tai niin ketterästi kuin mahdollista huomioiden asiakkaiden aikataulut) ja otetaan tuotteen käyttöön. Katselmoinnit otettaisiin pois käytöstä ja tuotteiden oikeanlaisuuden varmistamisen vastuu siirrettäisiin asiakkaalle. Käyttöönoton jälkeen sovittaisiin "after add" tai "extra release", joka vastaisi nykykatselmointeja.
Kehittäjä 1	Tarjoamalla asiakkaalle ennakkoon valmistellut 1-3 esimerkkiä valmiista toteutettavissa olevista (jo valmiista!) ratkaisusta. Lisäksi tarvitaan Agile/Scrum-metodologian tuoteomistaja ja mielellään paikalle joka aamupalaveri tai muuten päivittäinen kommunikaatio.
Tuotepäällikkö 2	[Tuotemäärittelyn] esikatselu ja siihen keskittyminen hyväksyntävaiheessa, katselmointeja asikkaan kanssa lisää. Kysymyksien esittäminen kootusti. Enemmän groomingia, koska usein kehittäjän sähköpostilla esittämä kysymys on epäselvä, sekä vaikeasti ymmärrettävissä.
Kehittäjä 2	Kaikkien tulisi keskittyä yhteen ja samaan projektiin saman aikaisesti.
Kehittäjä 3	Asiakkaan suurempi osallistuminen tuotteen tuotekehitysprosessissa

Kehittäjä 4	Määrittelijät sitoutuisivat projektiin enemmän kehitysprojektin aikana. Tuotekehityksen kannalta määrittelijä on ainoa henkilö, joka voi välittää asiakkaan tahtotilan. Kehityksen alla olevaan projektiin pitäisi määrittelijöiltä varata huomattavasti enemmän aikaa ja pitää myös asiakkaat tietoisina projektin aikana tulleista muutoksista/epäselvyyksistä.
Kehittäjä 5	Enemmän puheellista kommunikointia kehityksen-määrittelyn-asiakasvastaavan-asiakkaan välillä.
Kehittäjä 6	määrittelyiden laajamittainen groomaminen projektin kick-offin jälkeen
Testaaja 2	Enemmän aikaa tehdä huolellisempaa työtä. Testaamista aikaisemmassa vaiheessa, esimerkiksi konteilla heti kun task valmistuu.
Kehittäjä 7	Riittävä keskustelu määrittelijöiden ja tuotekehityksen välillä, jotta määrittelijät tietää mitä voi tehdä ja torpata mahdottomat tehtävät ja tuotekehitys tietää mitä pitää pystyä tekemään
Kehittäjä 8	lisää määrittelijöitä tai nykyisille vähemmän tekemistä! ei niitä välillä saa vastaamaan mihinkään ja täydellistä & virheetöntä speksiä en ole vielä nähnyt
Testaaja 3	Määrittelijän pitäisi istua kehitystiimin välittämättömässä läheisyydessä 7,5h päivässä. En väitä, että on toteutettavissa, mutta mielestäni ainoa apu.
Testaaja 4	Asiakkaan tahtotilaan kommentoiden: valmiimmat speksit, joihin asiakas on sitoutunut ettei sitten tule viime vaiheen muutoksia. Riittävästi katselmointeja. Määrittelijöiden pitäisi myös itse katselmoida QA:lla tuotoksia enemmän ja toimia kehitystiimeissä tiiviimmin.
Tuotepäällikkö 3	En tiedä.
Testaaja 5	Tuotekehitysprojektiin osallistuvien tahojen prioriteetit tulisi linjata yhtenäisiksi.
Tuotepäällikkö 4	Tuotteen käyttöönotolle määritellään slotti viikkotasolla jo ennen koodauksen aloittamista, mistä lasketaan taaksepäin kunkin osuuden valmistumisen deadline. Kullekin osuudelle varataan tarvittaviin tiimeihin aikaslotti työn tekemiselle siten, että slotti ei muutu koodauksen aloittamisen jälkeen.

8. Mikä estää poistamasta odotusta aiheuttavaa ongelman?

Vastaaja	Vastaus
Testaaja 1	hyvin organisoiduilla yhteisillä tapaamisilla ja seuranta palavereilla
Tuotepäällikkö 1	Nykyinen prosessi, jota olemme sitoutuneet noudattamaan.
Kehittäjä 1	Määrittelijöiden väitetty aikapula, jolloin he eivät ehdi saattaa asioita nopeammin valmiiksi.
Tuotepäällikkö 2	Liian kova tahti tuotekehtiyksessä.
Kehittäjä 2	Muutosvastarinta ja haluttomuus kokeiluun.
Kehittäjä 3	Asiakkaalla ei ole aikaa osallistua tuotekehitysprosessiin tarpeellisella tavalla
Kehittäjä 4	Määrittelijöiden työkuorma uusien ohjelmien määrittelyssä.
Kehittäjä 5	Kiire/tahtotila.
Kehittäjä 6	määrittelijöiden kiireet
Testaaja 2	Kireät aikataulut ja kiireellisemmät taskit.
Kehittäjä 7	Aikataulutuksessa ei ole tarpeeksi sisäistä kommunikaatiota
Kehittäjä 8	väsyneet tekosyyt, kuten "on vaikea löytää rooliin ketään"
Testaaja 3	Määrittelijöiden kiireellinen aikataulu ja monet päällekkäiset projektit.
Testaaja 4	Nähdäkseni asiakkaan aikataulut ja vaihtumiset, sekä määrittelijöiden kymmenet rinnakkaiset projektit.

Tuotepäällikkö 3	En tiedä.
Testaaja 5	En osaa sanoa.
Tuotepäällikkö 4	Resurssien yliallokointi.

9. Saatko TUOTEPÄÄLLIKÖILTÄ riittävästi tukea ja tietoa työhösi?

Vaihtoehdot: Aina tai lähes aina / Useimmiten / Harvemmin

Vastaaja	Vastaus
Testaaja 1	Useimmiten
Tuotepäällikkö 1	Useimmiten
Kehittäjä 1	Harvemmin
Tuotepäällikkö 2	Aina tai lähes aina
Kehittäjä 2	Harvemmin
Kehittäjä 3	Harvemmin
Kehittäjä 4	Harvemmin
Kehittäjä 5	Aina tai lähes aina
Kehittäjä 6	Useimmiten
Testaaja 2	Useimmiten
Kehittäjä 7	Useimmiten
Kehittäjä 8	Harvemmin
Testaaja 3	Harvemmin
Testaaja 4	Useimmiten
Tuotepäällikkö 3	Useimmiten
Testaaja 5	Useimmiten
Tuotepäällikkö 4	Aina tai lähes aina

10. Saatko TUOTEKEHITYSTIIMILTÄ riittävästi tukea ja tietoa työhösi?

Vaihtoehdot: Aina tai lähes aina / Useimmiten / Harvemmin

Vastaaja	Vastaus
Testaaja 1	Aina tai lähes aina
Tuotepäällikkö 1	Aina tai lähes aina
Kehittäjä 1	Aina tai lähes aina
Tuotepäällikkö 2	Useimmiten
Kehittäjä 2	Aina tai lähes aina
Kehittäjä 3	Aina tai lähes aina
Kehittäjä 4	Useimmiten
Kehittäjä 5	Aina tai lähes aina
Kehittäjä 6	Aina tai lähes aina
Testaaja 2	Aina tai lähes aina
Kehittäjä 7	Aina tai lähes aina
Kehittäjä 8	Aina tai lähes aina
Testaaja 3	Aina tai lähes aina
Testaaja 4	Useimmiten
Tuotepäällikkö 3	Useimmiten

Testaaja 5	Aina tai lähes aina
Tuotepäällikkö 4	Useimmiten

11. Saatko TESTAAJILTA riittävästi tukea ja tietoa työhösi?

Vaihtoehdot: Aina tai lähes aina / Useimmiten / Harvemmin

Vastaaja	Vastaus
Testaaja 1	Aina tai lähes aina
Tuotepäällikkö 1	Useimmiten
Kehittäjä 1	Aina tai lähes aina
Tuotepäällikkö 2	Useimmiten
Kehittäjä 2	Aina tai lähes aina
Kehittäjä 3	Aina tai lähes aina
Kehittäjä 4	Useimmiten
Kehittäjä 5	Aina tai lähes aina
Kehittäjä 6	Aina tai lähes aina
Testaaja 2	Aina tai lähes aina
Kehittäjä 7	Aina tai lähes aina
Kehittäjä 8	Aina tai lähes aina
Testaaja 3	Aina tai lähes aina
Testaaja 4	Aina tai lähes aina
Tuotepäällikkö 3	Useimmiten
Testaaja 5	Aina tai lähes aina
Tuotepäällikkö 4	Useimmiten

12. Kuinka selkeää eri roolien / toimijoiden välinen tehtävienjako mielestäsi on?

Vaihtoehdot: Erittäin selkeää / Selkeää / Sekavaa / Erittäin sekavaa

Vastaaja	Vastaus
Testaaja 1	Sekavaa
Tuotepäällikkö 1	Selkeää
Kehittäjä 1	Selkeää
Tuotepäällikkö 2	Selkeää
Kehittäjä 2	Sekavaa
Kehittäjä 3	Sekavaa
Kehittäjä 4	Sekavaa
Kehittäjä 5	Selkeää
Kehittäjä 6	Sekavaa
Testaaja 2	Selkeää
Kehittäjä 7	Selkeää
Kehittäjä 8	Selkeää
Testaaja 3	Selkeää
Testaaja 4	Erittäin selkeää
Tuotepäällikkö 3	Selkeää

Testaaja 5	Selkeää
Tuotepäällikkö 4	Erittäin selkeää

13. Onko tuotteen ominaisuuksien määrittelyn laatu, selkeys ja sisältö mielestäsi keskimäärin

Vaihtoehdot: Kiitettävää / Hyvää / Tyydyttävää / Välttävää

Vastaaja	Vastaus
Testaaja 1	Tyydyttävää
Tuotepäällikkö 1	Hyvää
Kehittäjä 1	Välttävää
Tuotepäällikkö 2	Hyvää
Kehittäjä 2	Hyvää
Kehittäjä 3	Tyydyttävää
Kehittäjä 4	Tyydyttävää
Kehittäjä 5	Hyvää
Kehittäjä 6	Välttävää
Testaaja 2	Tyydyttävää
Kehittäjä 7	Tyydyttävää
Kehittäjä 8	Tyydyttävää
Testaaja 3	Tyydyttävää
Testaaja 4	Välttävää
Tuotepäällikkö 3	Hyvää
Testaaja 5	Hyvää
Tuotepäällikkö 4	Hyvää

14. Perustele antamasi vastaus määrittelyn laadusta

Vastaaja	Vastaus
Testaaja 1	määrittely vaillinaista ja jättää tulkinnalle tilaa. Tämä taas aiheuttaa vääränlaista toimintaa, asiakkaan kannalta.
Tuotepäällikkö 1	Oma määrittely on niin hyvää kuin se on mahdollista olla nykyisessä mallissa. Kolmen muun määritelijän määrittelyistä tai niiden ajantasaisuudesta en osaa ottaa kantaa, koska ei ole aikaa tutkia asiaa.

Kehittäjä 1	Määrittelijä ohjeistaa joskus tarkasti, mutta silti virheellisesti (esim. puhelinnumero-kenttä ei voi olla Integer-kenttä). Joitakin asioita ei ole mietitty ollenkaan (määrittelyesimerkki: "tehdään kuten muissakin tämän alan [ohjelmissa]"). Määrittelyn tekeminen on varmasti hankalaa kaikkine kenttälistauksineen ja mock-up-työkalun screenshotteineen. Määrittely tekee kovasti turhaa työtä speksatessaan tarkasti kenttätyyppejä, jotka eivät kuitenkaan vastaa frameworkkimme kenttätyyppejä ja sovelluskehittäjä "tulkitsee". Lisäksi esim. labratuloksista ei tunnu tietävän kukaan, ja listoja voi olla [tuotemäärittelyssä], [konfiguraatiossa] ja ehkä myös hassusti nimetyssä [asiakaskonfiguraatiossa]. ERITYISESTI [Tuote] määrittelyt ovat farssi, monin verroin pahemmat, vaikkakin saman tyyliset, ongelmat kuin lomakekenttien määrittelyssä. PARANNUSEHDOTUS: Arvailun sijaan pitää jättää määrittelemättä ja tuoda esim. toteuttava tiimi (tai sen edustaja) mukaan suunnitteluun. Ei saa arvailla. Integraatioasiat pitää selvittää ennen kun sanotaan että [tuotemäärittely] on valmis. TAVOITE: [tuotemäärittelyn] on oltava selkeä ja yksikäsitteinen, eikä suuntaa-antava ohje, josta voi poiketa rajustikin kun toteutusvaiheessa vasta asia mietitään.
Tuotepäällikkö 2	Kun meillä on olemassa ns. tuotealusta, toteutamme asioita sen pohjalta, tällöin alustasta tulee jo tietyt raamit joita ei ole takoituksenmukaista kuvata aina uudelleen ja uudelleen. Uusien ominaisuuksien ja toimintojen kuvaamiseen on sovittu otettavaksi käyttöön malli jossa on user storyt ja vaatimukset taulukossa esitettynä. Jokainen TP tekee sen minkä ajan sallimissa rajoissa on mahdollista. Onnistumisprosentti [ohjelmissa] on hyvä vaikka menemme paikotellen "kevyellä" speksillä.
Kehittäjä 2	Määrittely on useimmiten laadukasta, mutta yksittäiset aukot siellä täällä määrittelydokumenttia viivästyttävät paljon projektia. Myös katselmointimuutoksia tulee liian usein liian paljon.
Kehittäjä 3	Useasti määrittelyt ovat puutteellisia (Puuttuu oleellista infomaatiota). Lomakkkeet eivät ole ihan mietitty kokonaisuuden kannalta toimiviksi, [Tuote] määrittelyt ovat puutteellisia
Kehittäjä 4	Määrittelyssä on liikaa tulkinnanvaraisia asioita ja osa asioista saattaa olla pitkään avoinna tuotekehitysprojektin aikana.
Kehittäjä 5	Aina löytyy epäkohtia, mutta ottaen huomioon määrittelyiden laajuuden (ja että kaikkien asioiden ajattelu etukäteen on oikeasti ihan suht vaikea homma), tuotteen pystyy aika valmiiksi tekemään määrittelydokumentin avulla.
Kehittäjä 6	määrittelyt ovat puutteelisiä eikä niitä pidetä ajantasalla uusista muutoksista
Testaaja 2	[tuotemäärittely]dokumentti on harvoin ajantasalla, toiset määrittelijät päivittävät dokumenttia tunnollisemmin kuin toiset. Lisäksi taskeissa pitäisi olla tarkempi kuvaus tehtävästä taskista.
Kehittäjä 7	Määrittelyt eroavat määrittelijöiden välillä. yhtenäisempi määrittely, jota päivitetään myös odottaviin projekteihin on tärkeää. Tuotekehityksen kommunikointia uusista mahdollisuuksista ja vanhojen ominaisuuksien muutoksista pitää myös parantaa.
Kehittäjä 8	riippuu aika paljon määrittelijästä. xy-kromosomeilla varustetuilta tulee syystä tai toisesta keskinkertaisesta surkeaan vaihtelevaa jälkeä, xx:iltä paljon parempaa. sitä en tiedä, miksi ensin mainittujen pitää tehdä kaikki hutiloiden.
Testaaja 3	Määrittely on välillä erittäin vanhaa eikä ajantasalla. Lisäksi määrittelyssä on tasoeroja eri henkilöiden välillä. Määrittelyyn pitäisi saada selkeät guidelinet sekä määrittelyn ylläpidon/päivityksen merkitystä pitää korostaa entisestään.

Testaaja 4	Hyvää speksiä ei tarvitse groomata ja se on aina ajan tasalla. Nyt speksiä on [tuotemäärittely]-dokkarissa, [konfiguraatiossa], [projektinhallintatyökalussa], sähköposteissa, [dokumentointialustassa], integraatiomiesten päässä jne. Helpompaa olisi kun speksi siirtyisi heti [tuotemäärittely]dokumentteihin kun se on mietitty. Meillä on lisäksi tuotteita, joille ei "edes yritetä saada ajantasaista speksiä", kuten [Tuote].
Tuotepäällikkö 3	Aika ei riitä täydelliseen määrittelyyn.
Testaaja 5	Määrittelyn laatu on keskimäärin hyvää, kun valinta tehdään hyvän ja tyydyttävän välillä.
Tuotepäällikkö 4	Nykyisellä dokumentaatiotasolla onnistumme 90% tapauksissa. Jäljelle jääneen 10% onnistumisen tavoittelu dokumentaatiota parantamalla lisäisi 50% dokumentaatioon kuluvaa aikaa, ja silti jäljelle jäisi x%.

15. Onko toteutettujen ominaisuuksien laatu mielestäsi keskimäärin

Vaihtoehdot: Kiitettävää / Hyvää / Tyydyttävää / Välttävää

Vastaaja	Vastaus
Testaaja 1	Tyydyttävää
Tuotepäällikkö 1	Hyvää
Kehittäjä 1	Välttävää
Tuotepäällikkö 2	Hyvää
Kehittäjä 2	Hyvää
Kehittäjä 3	Tyydyttävää
Kehittäjä 4	Tyydyttävää
Kehittäjä 5	Tyydyttävää
Kehittäjä 6	Tyydyttävää
Testaaja 2	Hyvää
Kehittäjä 7	Hyvää
Kehittäjä 8	Tyydyttävää
Testaaja 3	Tyydyttävää
Testaaja 4	Hyvää
Tuotepäällikkö 3	Tyydyttävää
Testaaja 5	Hyvää
Tuotepäällikkö 4	Hyvää

16. Perustele antamasi vastaus toteutettujen ominaisuuksien laadusta

Vastaaja	Vastaus
Testaaja 1	Kaikkia ominaisuuksia ei kehitetä ja ajatella loppuun, vaan annetaan testaajien testata ominaisuudet kuntoon.
Tuotepäällikkö 1	Ajoittain ominaisuuksien toiminnallisuuksissa joudutaan tyytymään hyvään, vaikka mahdollisuus olisi kiitettävään. Tähän vaikuttaa tuotealustan ominaisuudet sekä tuotekehitysprojekteihin annettu aika. Joissain tapauksissa myös tuotekehittäjän asenne, halu ja osaaminen vaikuttaa asiaan myös. Kaikki tuotekehittäjämme eivät ole koodaustaidoiltaan samalla viivalla.

Kehittäjä 1	Koodi on huonoa, käytettävyys on huonoa, [tuotemäärittely] on huono. Keinot päästä huonosta hyvään: Koodikatselmointi ja -laatuohjeistus, käytettävyystestaus ja [tuotemäärittely] kuntoon + jollekin PO-tyyliin vastuu teknisesta laadusta ja loppukäyttäjän kokemuksen laadusta.
Tuotepäällikkö 2	Tuotekehitys, sekä testaus ymmärtää hyvällä tasolla [tuotemäärittelyn] sisältöä ja kysyy tarvittaessa TP:ltä. Harvemmin esiintyy isoja puutteita joihin tarvitsee asiakkaan tai TP:n puuttua.
Kehittäjä 2	Tuotosta ei ehditä tarpeeksi suunnitella ja testata. Lopputulos on aina kompromissi.
Kehittäjä 3	Monesti samaa ominaisuutta kopioidaan tuotteesta toiseen eikä tehdä siitä yleiskäyttöistä pluginia.
Kehittäjä 4	Laatu on mielestäni niin hyvää kuin tällä kehitystahdilla ja prosesseilla on mahdollista. Projekteilla pitäisi olla enemmän aikaa tuottaakseen hyvää laatua.
Kehittäjä 5	Harvemmin voi sanoa kiitettäväksi kun useimmiten niissä on parannettavaa / näyttävät vähän karmeilta. Kuitenkaan surkeita toiminnallisuuksia nyt harvemmin edes päästetään ulos asti.
Kehittäjä 6	suorituskykyyn liittyvät ongelmat jäävät käytännössä aina ratkaisematta "ensimmäisessä versiossa"
Testaaja 2	Virheitä löytyy mielestäni aika paljon, mutta näitä korjataan kiitettävästi emmekä useimmiten joudu menemään tuotantoon softalla, jossa on suuri määrä bugeja avoinna. Arvostaisin kyllä, jos pystyisimme jotenkin laskemaan löytyvien virheiden määrä, lääkkeenä tähän voisi olla yksikkötestit ja automaattiset regressiotestit.
Kehittäjä 7	Kun määrittely on selvä ja yksityiskohtainen on tuotettu laatu testauksen avustuksella vähintään hyvää.
Kehittäjä 8	ainakin itsellä ensimmäinen jotenkuten toimiva ratkaisu on se lopullinen, ei täällä ehdi miettimään ja hieromaan.
Testaaja 3	Aikataulu ei mahdollista täydellistä ja pitkää hivelyä
Testaaja 4	Olemme menestyvä kasvuyritys. Emme olisi jos tuotteemme laatu ei olisi keskimäärin edes hyvää. Hyvyyteen vaikuttaa voimakkaasti tietenkin testaajat :)
Tuotepäällikkö 3	Aika ei riitä tai ei ole teknisesti mahdollista.
Testaaja 5	Toteutettujen ominaisuuksien laatu on keskimäärin hyvää, kun valinta tehdään kiitettävän ja hyvän välillä.
Tuotepäällikkö 4	Pystymme täyttämään asiakkaan tarpeen suurimmassa osassa ominaisuuksista, mutta osassa käytettävyyden kustannuksella.

17. Koetko tuotekehityssprintin ALOITUSPALAVERIN hyödylliseksi tapahtumaksi?

Vaihtoehdot: Aina / Useimmiten / Harvoin / En juuri koskaan

Vastaaja	Vastaus
Testaaja 1	Aina
Tuotepäällikkö 1	Aina
Kehittäjä 1	Useimmiten
Tuotepäällikkö 2	Useimmiten
Kehittäjä 2	Useimmiten
Kehittäjä 3	Aina

Kehittäjä 4	Aina
Kehittäjä 5	Aina
Kehittäjä 6	Useimmiten
Testaaja 2	Aina
Kehittäjä 7	Harvoin
Kehittäjä 8	Harvoin
Testaaja 3	Useimmiten
Testaaja 4	Aina
Tuotepäällikkö 3	Useimmiten
Testaaja 5	Useimmiten
Tuotepäällikkö 4	En juuri koskaan

18. Perustele ALOITUSPALAVERIN hyödyllisyydestä antamasi vastaus

Vastaaja	Vastaus
Testaaja 1	palaverit ovat hyvin ennalta suunniteltuna tehokkaita
Tuotepäällikkö 1	Lisäisin vähintään yhden vastuullisen (käyttöönottavan) asiakasvastaavan sekä tuotepäällikön läsnäolopakon aloituspalavereihin.
Kehittäjä 1	Se on yksi harvoista asioista jolloin määrittelijän kanssa voidaan kommunikoida tehokkaasti, ja useinhan kick-off antaa paljonkin muutostarpeita. Eli [tuotemäärittely] on vanhentunut heti esittelypäivänä, eli speksi on puutteellinen sillä hetkellä kun koodauksen oletetaan alkavan.
Tuotepäällikkö 2	Palaveri on ok, sisältö vaihtelee sprinttipalaverin pitäjien kesken merkittävästi. Kevyellä mallilla vdettynä ne toimivat hyvin, jos mennään taskitasolle niin tuntuu että aikaa palaa turhaan ja palaverin kesto pitenee oleellisesti. TP:tä kiinnosta isommat kokonaisuudet mm. lomakkeiden valmius-aste, integraatioiden valmiusaste, listauksen valmiusaste, yhteenvedot ja potilaskortin tilanne, kertomuksien tilanne jne...
Kehittäjä 2	Aloituspalaveri on tärkeä tilaisuus jossa sovitut asiat turvaavat kehittäjän työrauhaa, sekä mahdollistavat ennustettavuuden ainakin 2 viikoksi.
Kehittäjä 3	Tässä palaverissa päätetään mitä kahden viikonaikana luvataan tekevämme yhdessä projektipäälliköiden kanssa, joten erittäin hyödyllinen.
Kehittäjä 4	Aloituspalaverissa määritellään mitä seuraavan sprintin aikana luvataan tehdä. Tämä on tilaisuus missä tuoteomistaja ja tiimi päättävät mitä asioita edistetään. Aloituspalaverin jälkeen kaikilla asianomaisilla pitäisi olla yhteinen näkemys siitä mitä voi odottaa tapahtuvan seuraavan sprintin aikana. Tiimille tämä antaa selkeät raamit mihin työt priorisoidaan.
Kehittäjä 5	Tilanne saadaan selvennettyä eri ryhmille / kehittäjille. Tarpeen kyllä kahden tai ehkä jopa viikonkin välein, jotta tiedetään missä mennään suuremmassa mittakaavassa. Samalla tulee hyvää infoa takaisinpäin jos ei muuten ole tietoa siirtynyt (katselmoinnit, ym).
Kehittäjä 6	tavoitteen asettaminen ja tehtävän työn rajaaminen
Testaaja 2	Saas-puolella näitä kokouksia ei ole, joten en osaa vastata tähän kysymykseen.
Kehittäjä 7	Jos yrityksemme käyttäisi Scrum:ia kuten kuuluisi, siitä olisi apua.
Kehittäjä 8	tässä tiimissä ei sovita mitään tiukkaa ohjelmaa, joten palaverin hyöty on vähän kyseenalainen.
Testaaja 3	Kickoff antaa hyvin yleiskuvan tuotteesta, mutta ei paljon enempää.

Testaaja 4	Kyllä se on hyvä tietää mitä on tuleman ja groomata speksiä jos siinä vaiheessa.
Tuotepäällikkö 3	Mielestäni on hyvä tietää missä mennään ja mitä tullaan seuraavaksi tekemään.
Testaaja 5	On erittäin hyödyllistä sopia yhdessä alkavan sprintin tehtävät.
Tuotepäällikkö 4	Tuotepäällikölle ei ole tärkeää tuotteen ominaisuuksien tekojärjestys.

19. Koetko tuotekehityssprintin PÄÄTÖSPALAVERIN hyödylliseksi tapahtumaksi?

Vaihtoehdot: Aina / Useimmiten / Harvoin / En juuri koskaan

Vastaaja	Vastaus
Testaaja 1	Aina
Tuotepäällikkö 1	Useimmiten
Kehittäjä 1	En juuri koskaan
Tuotepäällikkö 2	Useimmiten
Kehittäjä 2	Harvoin
Kehittäjä 3	Aina
Kehittäjä 4	Harvoin
Kehittäjä 5	Aina
Kehittäjä 6	Harvoin
Testaaja 2	Aina
Kehittäjä 7	Harvoin
Kehittäjä 8	Useimmiten
Testaaja 3	Harvoin
Testaaja 4	Useimmiten
Tuotepäällikkö 3	Useimmiten
Testaaja 5	Useimmiten
Tuotepäällikkö 4	Harvoin

20. Perustele PÄÄTÖSPALAVERIN hyödyllisyydestä antamasi vastaus

Vastaaja	Vastaus
Testaaja 1	palaverit ovat hyvin ennalta suunniteltuna tehokkaita
Tuotepäällikkö 1	Lisäisin vähintään yhden vastuullisen (käyttöönottavan) asiakasvastaavan sekä tuotepäällikön läsnäolopakon päätöspalavereihin.
Kehittäjä 1	Tuotepäällikköä tai muuta yleisöä ei yleensä ole.
Tuotepäällikkö 2	Palaveri on ok, sisältö vaihtelee sprinttipalaverin pitäjien kesken merkittävästi. Kevyellä mallilla vedettynä ne toimivat hyvin, jos mennään taskitasolle niin tuntuu että aikaa palaa turhaan ja palaverin kesto pitenee oleellisesti. TP:tä kiinnosta isommat kokonaisuudet mm. lomakkeiden valmius-aste, integraatioiden valmiusaste, listauksen valmiusaste, yhteenvedot ja potilaskortin tilanne, kertomuksien tilanne jne...
Kehittäjä 2	Demoja ei pidetä oikeastaan koskaan, koska yleisöä ole palaverissa paikalla. Tapahtuman tulisi olla tilaisuus jossa tuotos esitetään kerran, eikä sitä tarvitse esitellä jokaiselle myöhemmin erikseen.

Kehittäjä 3	Saadaan status siitä mitä saatiin tehtyä ja mitä jäi tekemättä. Myös mahdolliset demot ovat hyviä tässä.
Kehittäjä 4	Päätöspalaveri olisi erittäin hyödyllinen JOS paikalle saadaan ulkopuolisia sidosryhmiä JA tiimi esittelee tehdyt työt. Tällä hetkellä päätöspalaveriin ei osallistuta eikä tehtyjä ominaisuuksia demota. Nämä varmastikin riippuvat toisistaan eli jos toisen korjaa, niin toinenkin korjaantuu (?).
Kehittäjä 5	-
Kehittäjä 6	ongelmista ja tehdyistä virheistä harvemmin opitaan
Testaaja 2	Saas-puolella näitä kokouksia ei ole, joten en osaa vastata tähän kysymykseen.
Kehittäjä 7	Jos yrityksemme käyttäisi Scrum:ia kuten kuuluisi, siitä olisi apua.
Kehittäjä 8	tiimi esittelee esimiehelle aikaansaannoksiaan. tuotepäälliköitä harvemmin näkyy.
Testaaja 3	Palaverissa pitäisi demota enemmän.
Testaaja 4	Tässä on piste missä tiimi demoaa tuotoksen määrittelijälle, ja siinä vaiheessa saadaan tärkeää tietoa olemmeko tehneet oikeita asioita.
Tuotepäällikkö 3	Mielestäni on hyvä tietää missä mennään.
Testaaja 5	On hyödyllistä esitellä sprintin aikana julkaistu tuotos muille. Näin varmistutaan siitä, että tuotekehitys on edennyt oikeaan suuntaan. Tuotekehityssprintin päätöspalaverien lisäksi hyödyllinen olisi tuotekehitysprojektin päätöspalaveri, jossa esiteltäisiin tuotekehitysprojektin tuotos.
Tuotepäällikkö 4	Perustoiminnallisuuksia ei ole tarve katselmoida.

21. Koetko tuotekehityssprintin RETROSPEKTIIVIN hyödylliseksi tapahtumaksi?

Vaihtoehdot: Aina / Useimmiten / Harvoin / En juuri koskaan

Vastaaja	Vastaus
Testaaja 1	Aina
Tuotepäällikkö 1	Harvoin
Kehittäjä 1	En juuri koskaan
Tuotepäällikkö 2	Harvoin
Kehittäjä 2	Useimmiten
Kehittäjä 3	Useimmiten
Kehittäjä 4	Useimmiten
Kehittäjä 5	En juuri koskaan
Kehittäjä 6	Harvoin
Testaaja 2	Aina
Kehittäjä 7	En juuri koskaan
Kehittäjä 8	Harvoin
Testaaja 3	En juuri koskaan
Testaaja 4	Harvoin
Tuotepäällikkö 3	Useimmiten
Testaaja 5	Useimmiten
Tuotepäällikkö 4	Useimmiten

22. Perustele RETROSPEKTIIVIN hyödyllisyydestä antamasi vastaus

Vastaaja	Vastaus
Testaaja 1	palaverit ovat hyvin ennalta suunniteltuna tehokkaita. Ja tällä voidaan oppia virheistä ja välttää ne jatkossa ja toki myös toistaa hyvin menneitä asioita.
Tuotepäällikkö 1	En osallistu ko tapahtumiin, jonka vuoksi vastasin edelliseen en juuri koskaan.
Kehittäjä 1	Asia käsitellään liian kevyesti tai ei ollenkaan.
Tuotepäällikkö 2	en tiedä ko. palaverista mitään
Kehittäjä 2	Retrospektiivi on erittäin tärkeä tapahtuma jossa on tarkoitus oppia virheistä. Ongelmat usein listataan, mutta niiden korjaamiseksi tehdään liian vähän.
Kehittäjä 3	Tässä saadaan asioita mikä meni hyvin ja mitä pitäisi parantaa. Aina varsinkin parantaminen jää action pointtien tasolle.
Kehittäjä 4	Retrospektiivi on hyödyllinen jos tiimi oikeasti keskittyy asiaan. Parantaisin retrospektiivejä siten, että pyrkisimme paremmin identifioimaan parannusehdotukset ja seuraamaan että nämä toteutetaan.
Kehittäjä 5	Ei koeta yhdessä epäkohtien/positiivisten asioiden käsittelyä hyödylliseksi. Nämä asiat tulevat yleensä ilmi paljon nopeammin ja asiaan reagoidaan heti (jos tarpeen/mahdollista), joten palaveria asioista ei pidetä.
Kehittäjä 6	pelkkä asioista puhuminen ei riitä muutoksen aikaan saaminseksi
Testaaja 2	Saas-puolella näitä kokouksia ei ole, joten en osaa vastata tähän kysymykseen. Yleisesti ottaen pidän retrospektiivejä ehkä hyödyllisimpinä tapaamisina, joita on.
Kehittäjä 7	Jos yrityksemme käyttäisi Scrum:ia kuten kuuluisi, siitä olisi apua. Yrityksessämme ei ole todellista product owneria, ei teamin mahdollisuutta valita mitä tehdään eikä siten myöskään dynaamista Scrumia. Jos tiimillä ei ole vaikutusmahdollisuutta siihen mitä tehdään ja 'product owneria' ei kiinnosta muulloin kuin erikseen tilattuina aikoina, jos silloinkaan ei Scrum myöskään toimi.
Kehittäjä 8	asioista voi valittaa niin kauan, että äänijänteet turpoavat, mutta mitään parannuksia mihinkään ei näiden tapahtumien pohjalta tehdä.
Testaaja 3	Emme ole keskustelleet siitä mikä meni hyvin ja mikä huonosti.
Testaaja 4	Juuri testaajana en näe tälle lisäarvoa.
Tuotepäällikkö 3	Itse olen osallistunut vain yhden kerran, joten en osaa sen tarkemmin kommentoida.
Testaaja 5	On hyödyllistä pysähtyä pohtimaan mitkä asiat sujuivat hyvin ja mitä voitaisiin parantaa.
Tuotepäällikkö 4	Tällöin voidaan tunnistaa dokumentaation tason nostamisen tarve.

23. Koetko tuotekehityssprintin GROOMING-tapahtumat hyödyllisiksi tapahtumiksi?

Vaihtoehdot: Aina / Useimmiten / Harvoin / En juuri koskaan

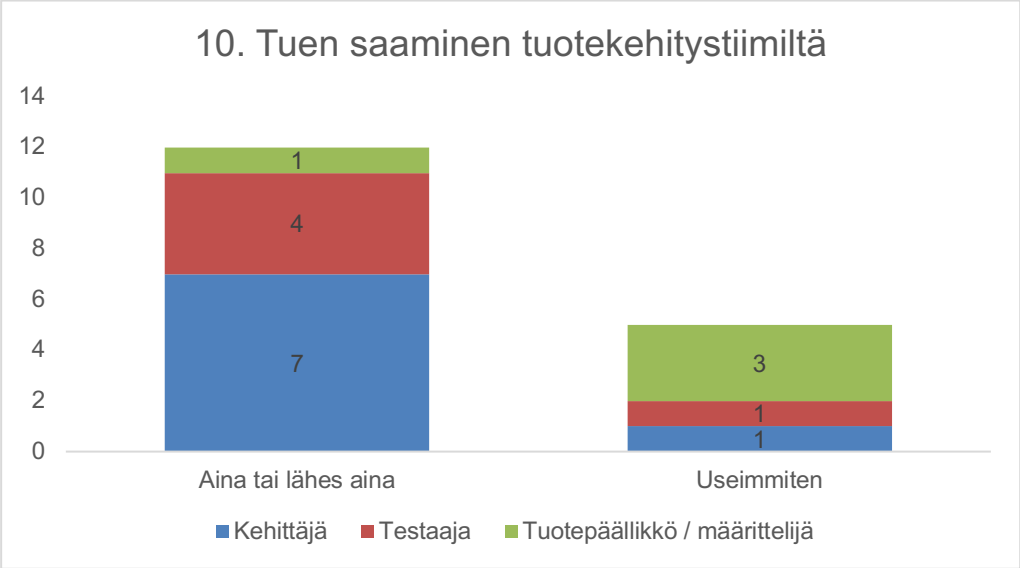
Vastaaja	Vastaus
Testaaja 1	Aina
Tuotepäällikkö 1	Aina
Kehittäjä 1	Useimmiten
Tuotepäällikkö 2	Aina
Kehittäjä 2	Harvoin

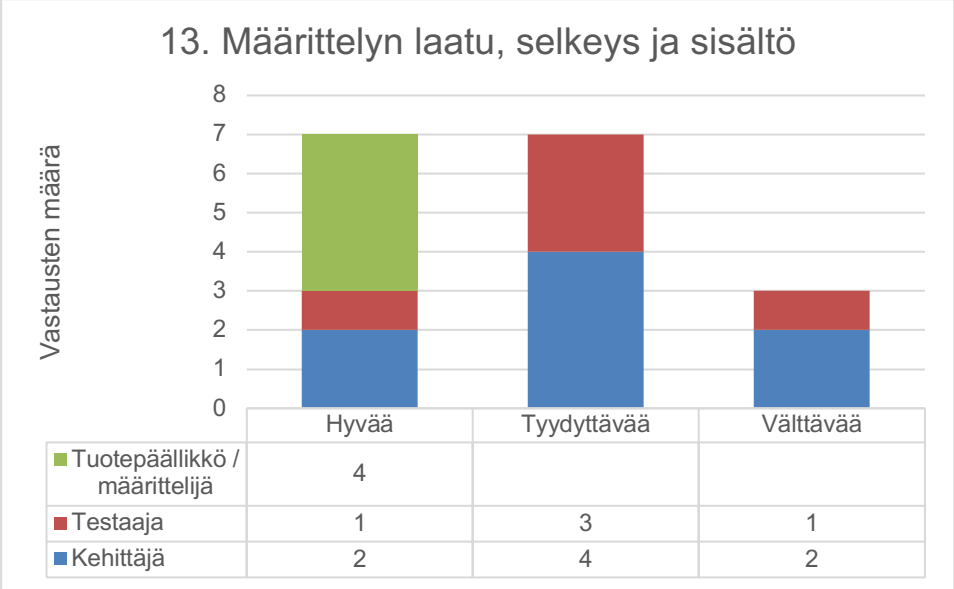
Kehittäjä 3	Aina
Kehittäjä 4	Aina
Kehittäjä 5	Aina
Kehittäjä 6	Aina
Testaaja 2	Aina
Kehittäjä 7	Aina
Kehittäjä 8	Harvoin
Testaaja 3	Useimmiten
Testaaja 4	Aina
Tuotepäällikkö 3	Aina
Testaaja 5	Aina
Tuotepäällikkö 4	Aina

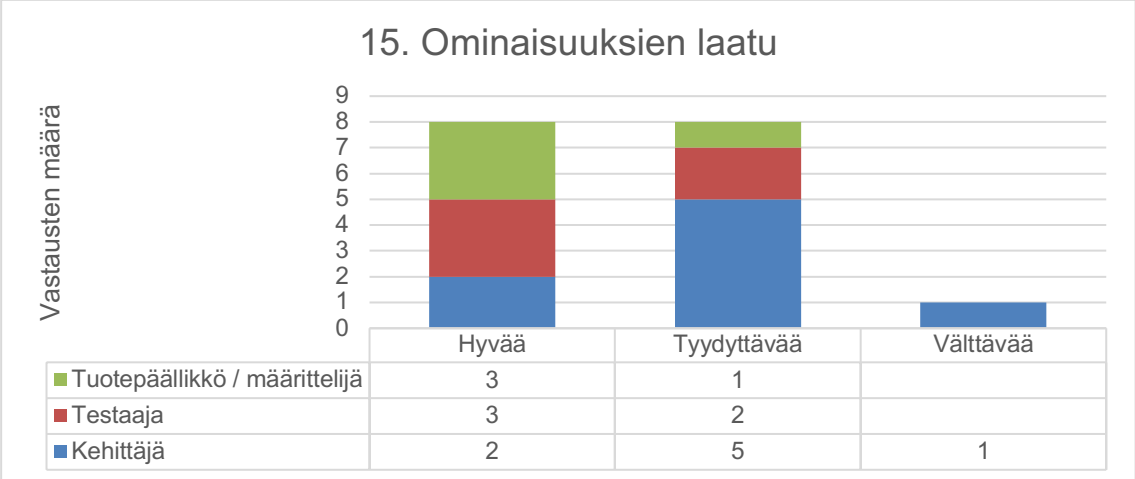
24. Perustele GROOMINGIN hyödyllisyydestä antamasi vastaus

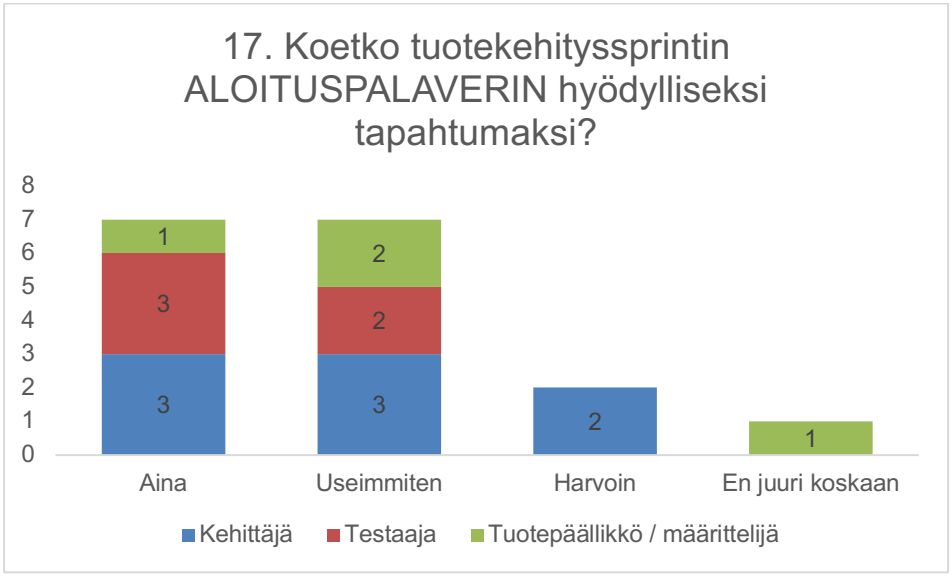
Vastaaja	Vastaus
Testaaja 1	Tässä palaverissa saadaan määrittelyn ongelmakohdat taklattua.
Tuotepäällikkö 1	Lisäisin vähintään yhden vastuullisen (käyttöönottavan) asiakasvastaavan läsnäolopakon tai groominpöytäkirjojen läpikäynnin ennen yhteydenottoja.
Kehittäjä 1	Määrittelijän kanssa kommunikointi on olennaista, vaikka grooming voisi olla enemmän tiimin sisäistä kysymyksen mukaista toimintaa. Tapahtuu hyvin vähän, mutta on tarpeellista.
Tuotepäällikkö 2	Keskustelemalla selviää puolin ja toisin oleelliset asiat, jotka sposteissa usein jää epäselviksi. Groomauksiin tulisi keskittyä enemmän ja kaikki epäselvät asiat käydä siellä läpi s-postin sijaan.
Kehittäjä 2	Groomingia on erittäin vaikeaa saada järjestetyksi tai sitä joudutaan odottamaan liian pitkään. Työt eivät etene odotellessa. Grooming pitäisi olla sprintin vakio-ohjelmassa ja kaikkien tulisi kunnioittaa tapahtumaa.
Kehittäjä 3	Tässä voidaan käydä läpi kaikki yksityiskohdat, mitkä useimmiten eivät selviä määrittelystä.
Kehittäjä 4	Tämä on tällä hetkellä ainoa paikka, missä voidaan keskittyneesti yhdessä poistaa epäselvyydet yms. Kehittäisin tätä siten, että meillä ylipäätään olisi taas grooming palavereita ja näihin valmistauduttaisiin kunnolla.
Kehittäjä 5	Täysin pakollista, varsinkin jos asiat ovat suuria tai niitä on useita. Muussa tapauksessa useimmiten kyllä soitan asiasta tietävälle.
Kehittäjä 6	paras paikka hankkia vastauksia ja jakaa tietoa projektin aikana
Testaaja 2	Saas-puolella näitä kokouksia ei ole, joten en osaa vastata tähän kysymykseen.
Kehittäjä 7	Ainoa tilaisuus jossa voi olla varma että määrittelijä (Product Owner) on saatavilla ja keskittyy juuri tähän tuotteeseen.
Kehittäjä 8	ei minua kehittäjänä lopulta juurikaan kiinnosta, mikä on tuotteen työmääräarvio. se on jonkun muun murhe murehtia niistä. minä joka tapauksessa istun tuolilla ja kirjoitan koodia.
Testaaja 3	Groomingit ovat hyviä ja niitä pitäisi pitää tasaisin väliajoin.
Testaaja 4	Kyllä tätä on syytä tehdä. Erityisesti kannattaa groomata määrittelijän / tuotepäällikön läsnä ollessa.
Tuotepäällikkö 3	Hyödyllinen..
Testaaja 5	Grooming-tapahtumat ovat aina tuntuneet hyödyllisiltä.

Tuotepäällikkö 4	Mahdolliset kysymykset voidaan selventää ilman tarvetta yleisluonteisesti nostaa dokumentaation tasoa.

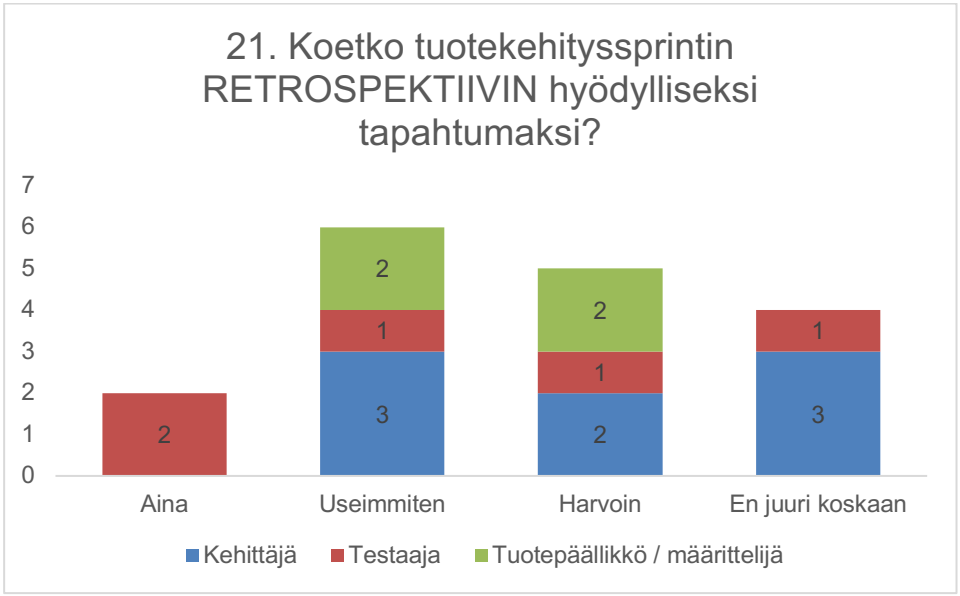

10. Tuen saaminen tuotekehitystiimiltä
14
12
10
8
6
4
2
0
1
4
7
3
1
1
Aina tai lähes aina
Useimmiten
Kehittäjä
Testaaja
Tuotepäällikkö / määrittelijä

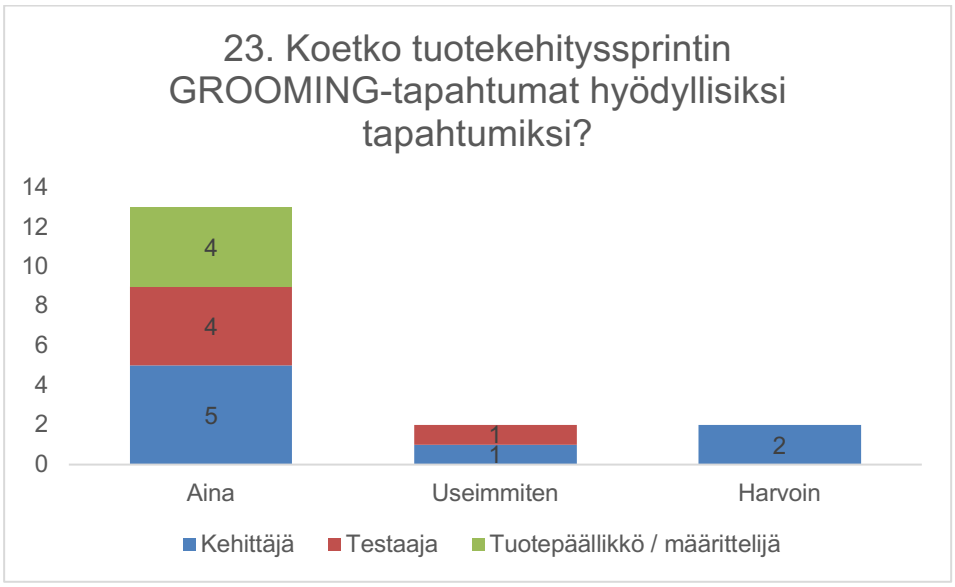

13. Määrittelyn laatu, selkeys ja sisältö
Vastausten määrä
8
7
6
5
4
3
2
1
0
Hyvää
Tyydyttävää
Välttävää
Tuotepäällikkö / määrittelijä 4
Testaaja 1 3 1
Kehittäjä 2 4 2


15. Ominaisuuksien laatu
Vastausten määrä
9
8
7
6
5
4
3
2
1
0
Hyvää
Tyydyttävää
Välttävää
Tuotepäällikkö / määrittelijä 3 1
Testaaja 3 2
Kehittäjä 2 5 1


17. Koetko tuotekehityssprintin ALOITUSPALAVERIN hyödylliseksi tapahtumaksi?
8
7
6
5
4
3
2
1
0
1
3
3
2
2
3
2
1
Aina
Useimmiten
Harvoin
En juuri koskaan
Kehittäjä
Testaaja
Tuotepäällikkö / määrittelijä


19. Koetko tuotekehityssprintin PÄÄTÖSPALAVERIN hyödylliseksi tapahtumaksi?
7
6
5
4
3
2
1
0
2
2
3
2
1
1
1
4
1
Aina
Useimmiten
Harvoin
En juuri koskaan
Kehittäjä
Testaaja
Tuotepäällikkö / määrittelijä


21. Koetko tuotekehityssprintin RETROSPEKTIIVIN hyödylliseksi tapahtumaksi?
7
6
5
4
3
2
1
0
2
3
1
2
2
1
2
3
1
Aina
Useimmiten
Harvoin
En juuri koskaan
Kehittäjä
Testaaja
Tuotepäällikkö / määrittelijä


23. Koetko tuotekehityssprintin GROOMING-tapahtumat hyödyllisiksi tapahtumiksi?
14
12
10
8
6
4
2
0
5
4
4
1
1
2
Aina
Useimmiten
Harvoin
Kehittäjä
Testaaja
Tuotepäällikkö / määrittelijä