



**SAVONIA**

OPINNÄYTETYÖ - YLEMPI AMMATTIKORKEAKOULUTUTKINTO  
TEKNIIKAN JA LIIKENTEEN ALA

# TUOTTEIDEN LAADUN KEHIT- TÄMINEN JULKAISUJEN AVULLA

TEKIJÄ/T: Petteri Hakkarainen

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Teknologiaosaamisen johtamisen tutkinto-ohjelma	
Työn tekijä(t) Petteri Hakkarainen	
Työn nimi Tuotteiden laadun kehittäminen julkaisujen avulla	
Päiväys 30.11.2017	Sivumäärä/Liitteet 44+5
Ohjaaja(t) Pertti Varis (Savonia-amk), Tommi Rissanen (Mediconsult Oy)	
Toimeksiantaja/Yhteistyökumppani(t) Mediconsult Oy	
Tiivistelmä <p>Mediconsult Oy on kotimainen vuonna 1975 perustettu vaativiin terveyden- ja sosiaalihuollon tietojenkäsittely- ja konsultointitehtäviin keskittynyt ICT-palveluyritys. Työ toteutettiin tuotekehityksen Kanta- ja integraatioyksikössä, joka keskittyy erityisesti kansallisen Potilastiedon arkiston ja sähköisen reseptin toteutuksiin. Opinnäytetyön tavoitteena oli saada oikea kuva tehtävän ohjelmistotuotteen nykyisestä laadusta ja pyrkiä parantamaan sitä uusien julkaisujen avulla. Manuaalisia työvaiheita haluttiin karsia ja korvata niitä sovellusautomaatiikkaa, sovellusrobotiikkaa ja tekoälyä hyödyntämällä. Julkaisussa manuaalisen työn väheneminen parantaa julkaisujen laatua ja vähentää inhimillisiä virheitä.</p> <p>Teoriaosuudessa käytiin läpi eri laatuun vaikuttavia näkökulmia, yrityksen käytössä olevia tuotekehityksen prosessimalleja sekä laatuun vaikuttavia työkaluja siinä laajuudessa, mitä on tarpeellista ymmärtää sovelluskehitysprosessia. Työssä myös analysoitiin eri julkaisujen prosessiin kuluva aikaa sekä työvaiheisiin menevää aikaa. Tätä analysointia hyödynnetään ennakoitaessa kehitysprosessiin menevää aikaa ja tulevaa julkaisuaikataulua. Julkaisu-prosessin nopeutuminen nostaa palvelun laatua, sillä ajoittain on tarvetta julkaista ominaisuuksia lyhyellä varoitusaikalla.</p> <p>Opinnäytetyön tuloksena syntyi kahdeksan konkreettista toimenpidettä, jolla laatua saadaan vielä parannettua. Ohjelmistorobotiikkaa ja tekoälyä voidaan hyödyntää myös yrityksen muussa toiminnassa. Jatkuvan parantamisen näkökulmasta luotuja tutkimusmenetelmiä pystytään hyödyntämään myös tulevaisuudessa.</p>	
Avainsanat Laadunvalvonta, laadunvarmistus, laadun kehittäminen, ohjelmistokehitys, tekoäly	

Field of Study Technology, Communication and Transport			
Degree Programme Master's Degree Programme in Engineering Knowledge Management			
Author(s) Petteri Hakkarainen			
Title of Thesis Development of product quality through publications			
Date	30.11.2017	Pages/Appendices	44+5
Supervisor(s) Pertti Varis (Savonia-amk), Tommi Rissanen (Mediconsult Oy)			
Client Organisation /Partners Mediconsult Ltd.			
<p>Abstract</p> <p>Mediconsult Oy is a Finnish ICT service company focused on computing and consulting services in the health and social care industry in 1975. The work was carried out in the product development unit, which focuses especially on the implementation of the Patient Data Repository and Electronic prescription. The aim of the thesis was to get the right picture of the current quality of the software product and to try to improve it with the new publications. The manual work phases needed to be reduced and replaced by using application automation, application robotics and artificial intelligence. Reduced manual work improves publishing quality and reduces human errors.</p> <p>The theoretical section of the thesis examines aspects of different quality, the product development process models used by the company and the tools that influence quality, to the extent necessary to understand the application development process. The thesis also analyzed the time spent on the various publications and the time spent working on the work. This analysis can be used to anticipate the time needed for the development process and the future release timetable. The speed of the publishing process increases the quality of the service, as at times there is a need to publish features at short notice.</p> <p>The result of the thesis was the creation of eight concrete measures to improve the quality. Software biotechnology and artificial intelligence can also be used in other activities of the company. From the point of view of continuous improvement, the research methods created can be utilized in the future as well.</p>			
Keywords quality control, quality assurance, the quality of development, artificial intelligence			

## ESIPUHE

Tämä työ toteutettiin Mediconsultin Kanta- ja integraatiot tuotekehitysyksikössä tutkimuksellisenä kehittämistyönä huhtikuun – marraskuun 2017 aikana. Opinnäytetyön tavoitteena oli kehittää tuotekehityksen tuotteiden julkaisuja niin, että tuotteiden laatu paranee ja julkaisu on mahdollisimman pitkälle automatisoitu prosessi.

Terveystuotteen tuotekehitystä velvoittaa lait, erilaiset määräykset ja standardit sekä yrityksen oma laatujärjestelmä. Uudistetun julkaisuprosessin avulla pyritään huomioimaan nämä vaatimukset niin, että niistä tulisi luonteva osa tuotekehitystä. Tarkoitus on integroida nämä vaatimukset tehokkaaseen ketterään kehittämiseen.

Tuotteiden paremmalla laadulla saavutetaan kustannussäästöjä ja asiakastyytyväisyys paranee vähenevien ohjelmavirheiden johdosta. Tietyn julkaisun ennustettavuus on myös yksi asia mikä paranee laadun paranemisen myötä.

Perusongelmana on, että julkaisuaikataulut eivät pidä ja julkaisua joudutaan siirtämään löydettyjen virheiden takia. Tämä aiheuttaa myös ongelmia asennuksissa: pahimmillaan asennus joudutaan tekemään lyhyessä aikavälissä useaan otteeseen, joka aiheuttaa kustannuksia ja töiden organisointia uudelleen.

## SISÄLTÖ

1	JOHDANTO .....	7
1.1	Keskeiset käsitteet .....	7
1.2	Toimeksiantaja .....	8
2	LAATU .....	9
2.1	Laatu käsitteenä .....	9
2.2	Mikä on laadunhallintajärjestelmä?.....	11
2.3	Kehittämistehtävän taustaa .....	12
2.3.1	Työn lähtökohdat .....	12
2.3.2	Laatumittarit.....	13
2.3.3	Kehittämis- ja tutkimuskohteet.....	14
2.3.4	Kehittämistehtävän tavoite ja tarkoitus.....	14
2.4	Rajaukset .....	14
3	TERVEYDENHUOLLON OHJELMISTOKEHITYS .....	15
3.1	Lääkintälaitteen ohjelmistotuotanto.....	16
3.2	Yrityksen laatujärjestelmä .....	16
3.2.1	Tuotteen toteuttaminen tuotekehityksessä.....	18
3.3	Tuotekehityksen käytössä oleva prosessimalli .....	18
3.3.1	Scrum .....	18
3.4	Yrityksen tuotekehityksen ohjelmistot .....	20
3.4.1	Atlassian Jira .....	20
3.4.2	Git.....	21
3.4.3	Atlassian Crucible & FishEye .....	22
3.4.4	Jenkins.....	22
3.4.5	Slack .....	23
3.4.6	Testauksen työkalut .....	23
4	OHJELMISTOTUOTTEEN VALMISTUSLAATU.....	24
4.1	Valmistuksen arvovirtakuvaus (Value Stream Mapping) .....	24
4.1.1	Valmistuksen prosessin suorituskyky .....	24
4.2	Tuotelaatu.....	26
4.3	Valmistuslaatu .....	27
4.4	Asiakaslaatu .....	28

5	TUTKIMUSKOHTEET JULKAISUISSA .....	29
5.1	Tuotteen julkaisu .....	29
5.2	Kanta-adapteri versio 4.2.0.....	30
5.2.1	Version käyttöönotto asiakkaalla .....	31
5.3	Versio 4.2.1 .....	32
5.4	Versio 4.2.2 .....	33
5.4.1	Version käyttöönotto asiakkaalla .....	34
5.5	Versio 4.2.3.....	34
5.5.1	Version käyttöönotto asiakkaalla .....	35
5.6	Versio 4.3.0.....	35
5.7	Yhteenveto eri mittareista ja laadusta julkaisuissa.....	36
6	KEHITTÄMISKOHTEITA LAADUN PARANTAMISEEN TERVEYDENHUOLLON OHJELMISTOKEHITYKSESSÄ .....	38
6.1	Koodin katselmointikäytännöt .....	38
6.2	Uudet ominaisuudet potilastiedon arkistossa ja sähköisessä reseptissä.....	38
6.3	Asennusohjeiden ja konfiguraatiomuutosten dokumentointi.....	38
6.4	Versiojulkaisussa tehtävät testitapaukset ja testausympäristöt.....	38
6.5	Retrospektiivin tulokset .....	39
6.6	Prosessiin kuuluva aika .....	39
6.7	Julkaisu ja julkaisukuvauksen automatisointi.....	39
6.8	Tekoälyn ja ohjelmistorobotin hyödyntäminen .....	40
7	YHTEENVETO JA JATKO.....	43
	LÄHTEET JA TUOTETUT AINEISTOT .....	45
	LIITE 1: PROSESSIKUVAUS VALMISTUKSESTA.....	46
	LIITE 2: OHJELMISTOROBOTIN VUOKAAVIO .....	47
	LIITE 3: DIALOGFLOW KOMENTOJEN OPETUS .....	48
	LIITE 4: ROBOTIN PALAUTTAMAA TIETOA JULKAISUSTA .....	49
	LIITE 5: ESIMERKKI LUONNOLLISESTA KESKUSTELUSTA OHJELMISTOROBOTIN KANSSA .....	50

## 1 JOHDANTO

### 1.1 Keskeiset käsitteet

Keskeisimmät käsitteet Suomen kansallisista terveydenhuollon tietojärjestelmistä ja tuotekehityksestä ovat:

#### **Potilastietojärjestelmä**

Potilastietojärjestelmä sisältää kaikki vaatimaan potilas- ja asiakastiedon hallintaan tarvittavat ominaisuudet tietojen jokapäiväisestä käsittelystä aina hallinnon raportointiin. Suomessa on useita erilaisia potilastietojärjestelmiä erilaisilla ominaisuuksilla. Potilastietojärjestelmä käsitetään lääkintälaitteeksi.

#### **Potilastiedon arkisto**

Potilastiedon arkisto on Kelan hallinnoima tietojärjestelmä, jota käytetään potilastietojärjestelmällä. Potilastiedon arkistolla on keskeinen rooli tietojen välittämisessä terveydenhuollon palvelujen antajien kesken. Kansalainen näkee terveydenhuollossa kirjattuja tietoja Omakanta-palvelun kautta. Potilastiedot arkistoidaan teknisesti yhtenevässä muodossa, mikä mahdollistaa tietojen siirrettävyyden järjestelmästä toiseen ja parantaa tietojen saatavuutta potilaan hoitoon osallistuvissa toimintayksiköissä. (Kanta 2017)

#### **Ammattikortti, organisaatiokortti tai toimikortti**

Terveydenhuollon ammattikortti on väestörekisterikeskuksen myöntämä kortti henkilöiden luotettavaan tunnistamiseen ja potilasasiakirjojen sekä lääkemääräysten sähköiseen allekirjoittamiseen. Jokaisella terveydenhuollon ammattilaisella on oma korttinsa. Potilastietojärjestelmän sähköisen lääkemääräyksen käyttöönotto edellyttää ammattikorttia. Ammattikortteja on myös saatavana varakortteina. Varakorttia käytetään silloin, kun terveydenhuollon toimikortti on rikkoutunut tai ei ole muista syistä työntekijällä käytettävissä (esim. unohtunut kotiin). Organisaatiokortti on organisaatioiden ja yhteisöjen käytössä ja sillä on mahdollista esimerkiksi suorittaa työasemaan kirjaantuminen organisaatiokorttia käyttäen.

#### **Sähköinen lääkemääräys**

Lääkäri määrää ja allekirjoittaa henkilökohtaisella toimikortilla kaikki reseptit sähköisesti. Ne tallennetaan keskitettyyn tietokantaan, jota kutsutaan Reseptikeskukseksi. Reseptikeskuksen rekisterinpitäjä on Kela. Reseptikeskus sisältää kaikki reseptit ja apteekkien niihin tekemät toimitusmerkinnät. Reseptikeskuksen tietojen perusteella mikä tahansa apteekki voi toimittaa lääkkeet. (Kanta 2017)

#### **Kanta-adaptori**

Mediconsultin tuotekehitysyksikön tekemä sovellus mikä siirtää Mediconsultin potilastietojärjestelmään kirjatut tiedot Potilastiedon arkistoon ja reseptikeskukseen.

## Version käyttöönotto

Version käyttöönotto sisältää kaikki toimet, että asiakas voi käyttää tietojärjestelmää. Käyttöönoton seurauksena otetaan käyttöön uusi tai parannettu järjestelmä.

### 1.2 Toimeksiantaja

Mediconsult Oy on kotimainen vuonna 1975 perustettu vaativiin terveyden- ja sosiaalihuollon tietojenkäsittely- ja konsultointitehtäviin keskittynyt ICT-palveluyritys. Olemme useita vuosikymmeniä toimittaneet ratkaisuja ja tarjonneet palveluja toimintakriittisiin ympäristöihin. Yhtiö tutkii tällä hetkellä mahdollisuutta laajentaa toimintaa ulkomaille. Yhtiö toimii yksinomaan terveydenhuollon ja sosiaalihuollon sektorilla ja sen päätoimiala on terveyden- ja sosiaalihuollon tietojärjestelmät ja niihin liittyvä konsultointi. Pääsääntöisesti yrityksen tekemä konsultointi liittyy sen tarjoamien ratkaisujen käyttöönottoihin ja niissä liittyviin käyttöönottavien organisaatioiden toiminnan kehittämiseen ja prosessien uudelleen järjestelyyn ja toiminnanohjaukseen. Samoin osallistumme ja konsultoimme asiakkaitamme kansallisten säännösten huomioimisessa omassa toimintaympäristössään. (Mediconsult 2017)

Yhtiön kotipaikka on Helsinki ja se on kokonaan suomalaisessa omistuksessa. Yhtiön toimipisteet sijaitsevat Helsingissä, Kuopiossa, Joensuussa ja Salossa. Suomalaisen omistuksen lisäksi on yhtiön tuotekehitys sekä ohjelmistotuotanto ja palvelut täysin Suomessa tapahtuvia. Mediconsult on kasvava yritys ja työllistämme tällä hetkellä vähän yli 100 työntekijää. Toiminta on kasvanut vuosittain n. 20 %, samoin henkilöstö. Yrityksellä on riittävät resurssit ja monipuolista osaamista, joten pystymme hyvin vastaamaan tulevaan sote-muutokseen, tuottamaan ratkaisumme ja palvelumme käyttöön kustannustehokkaasti ja toimintaa tehostavasti sekä auttamaan asiakkaitamme saamaan parhaan hyödyn muutoksesta. Yli puolet yrityksen henkilökunnasta työskentelee tuotekehityksessä ja uusien ratkaisujen kehittämisen parissa. Tutkimuksen ja tuotekehityksen kustannusten osuus yrityksen liikevaihdosta on vuosittain noin 30–35 %. Tuotekehityksen Kanta- ja integraatioyksikkö keskittyy erityisesti kansallisen Potilastiedon arkiston ([www.kanta.fi](http://www.kanta.fi)) ja sähköisen reseptin toteutuksiin.

Mediconsult Oy:n henkilökunta koostuu eri alojen – pääasiallisesti terveydenhuollon ja tietotekniikan – ammattilaisista. Heillä on monipuolista osaamista teollisuuden eri aloilta ja asiantuntemusta suomalaisen terveydenhuollon vaatimuksista ja standardeista. Näin pystytään huomioimaan vaatimukset ohjelmistoille ja erityisesti lainsäädännöllinen ympäristö ja viitekehys toimitettaessa järjestelmiä suomalaisille asiakkaille toiminta-alan muuttuvassa ympäristössä. Toiminnan hallinto on pidetty pienenä. Sille on edellytyksenä organisaation sisäisesti hyväksytyt tehokkaat työ- ja toimintatavat. (Mediconsult 2017, 5)

Järjestelmäkehitys on organisoitu sekä vesiputousmallin- että ketterän kehityksen menetelmin. Kehitystyössä on osallisina yrityksen omat erikoisalojen osaajat sekä sidosryhminä eri alojen ammattilaisia. Myös asiakkaat ja käyttäjät ovat tärkeä ryhmä kehitystyön ohjaajana. Ketterien ohjelmistomenetelmien myötä voidaan ohjelmistokehitykseen ottaa jatkuvasti uusia tuoteideoita ja toteutettavia ohjelmistotöitä ja näiden toimitusaika on erittäin lyhyt, parhaimmillaan 1–3 viikkoa.



## 2 LAATU

### 2.1 Laatu käsitteenä

Laadusta on monta eri tulkintaa näkökulmasta riippuen. Ei ole olemassa yksikäsitteistä laadun määritelmää. Sen voi nähdä monelta suunnalta ja eri ihmiset näkevät laadun eri tavoin. Voidaan myös sanoa, että laatu on vaatimuksenmukaisuutta. Laatu on siis sitä, että asiakkaan vaatimukset, odotukset, tottumukset ja tarpeet täyttyvät. Hyvä laadun määritelmä voisi olla esimerkiksi seuraavanlainen:

”Laatu on kaikki ne ominaisuudet ja piirteet, jotka tuotteella tai palvelulla on ja joilla se täyttää asiakkaan odotuksia, vaatimuksia tai tottumuksia, olivatpa ne sitten ilmaistuja tai piilossa olevia” (Pesonen 2007, 36).

Aina kaikkien laatuksiteereiden täyttäminen ei ole mahdollista, eikä välttämättä taloudellisesti kannattavaa, mutta lähtökohtana tulee olla kuitenkin asiakkaan odotukset. Olennaisen tärkeää on siis selvittää ja tietää mitä asiakas todella odottaa. Palvelubisneksessä palvelun laadun määrittäminen on erityisen vaikeaa, koska asiakkaat eivät välttämättä osaa etukäteen sanoa, mitä he haluavat. (Pesonen, 2007, 37).

Yleisesti laadulla ymmärretään asiakkaan tarpeiden täyttämistä yrityksen kannalta mahdollisimman tehokkaalla ja kannattavalla tavalla. Laatuun liittyy myös tarve jatkuvaan parantamiseen, niin nopeasti kuin kehitys sallii. Lecklin (2002, 19) toteaa, että kokonaisvaltainen laatujohtaminen koostuu asiakkaiden ja niiden tyytyväisyyden hallinnasta, korkealaatuisesta toiminnasta ja markkinoiden ja asiakkaiden ymmärtämisestä. Laatuun myös liittyy usein automaattisesti positiivinen ennakoasenne. Hyvin usein kuitenkin puhuttaessa ohjelmistotuotteiden laadusta leimaa niitä tietty negatiivinen ennakoasenne.

Ohjelmistoteollisuudessa normaalisti julkaisukuvaukset listaavat tunnetut ohjelmiston viat ja niiden mahdolliset kiertotavat. Jos tätä tapaa vertaa esimerkiksi autoteollisuuteen, niin voisi olla vaikeaa myydä autoa, jos käyttöoppaassa olisi kerrottu, että kojelauta natisee, hansikaslokero ei pysy kiinni ja lukko jäätyy talvisin (Haikala ja Märijärvi 2004, 192).

Paul Lillrank (Lillrank 1998, 34) esittää kuusi erilaista laatuominaisuutta, joita pyrittiin huomioimaan tämän työn toteutuksessa. Näitä laatuominaisuuksia on käytetty myös Mediconsultin laatuksikirjassa.

- Valmistuslaatu keskittyy valmistusprosessiin ja varmistaa tuotteiden valmistuksen määritysten mukaan. Prosessia kehittämällä virheet pyritään ennakoidaan ja välttämään.
- Tuotelaatu korostaa suunnittelun osuutta tuotteen laadun määrittämisessä.
- Arvolaadussa korkein laatu on sillä tuotteella, joka antaa parhaimman kustannus-hyötysuhteen eli parhaan arvon sijoitetulle pääomalle.

- Kilpailulaatu. Laatu on riittävä, kun se on yhtä hyvä kuin kilpailijoilla. Tätä parempi laatu on yli-laatua ja resurssien tuhlausta.
- Asiakaslaatu. Asiakkaiden tarpeet tyydyttävä laatu on hyvää laatua.
- Ympäristölaatu. Laatua voidaan mitata myös ympäristön ja yhteiskunnan kannalta. Tuotteen suunnittelussa tulee myös sen elinkaari, resurssien käyttö suunnittelusta hävittämiseen asti ottaa huomioon.

Ylläolevat laadun näkemykset ovat Paul Lillrankin näkemyksiä, eivätkä ne edusta sellaisenaan Medi-consultin arvoja ja tavoitteita. Kilpailulaatu voi olla myös kilpailuetu. Kilpailijoita parempi laatu voi olla perusteltua, jos asiakkaat ovat siitä valmiita maksamaan tai valitsemaan tuotteen hyvän laadun takia. Näistä laatuominaisuuksista erityisesti valmistuslaatua ohjelmistotuotteiden näkökulmasta ja tuotelaatua pyrittiin kehittämään erilaisilla menetelmillä.

Suomessa toimii Suomen laatu yhdistys, joka edistää laadukasta johtamista ja kannustaa tulokselliseen kilpailukyvyyn kehittämiseen. Suomen lautupalkinnon myöntää Suomen Laatu yhdistys parhaille ulkoisiin asiantuntija-arviointeihin osallistuneille organisaatioille. Yhdistys on määritellyt erilaisia tunnuspiirteitä, mistä laatu yrityksen erinomaisuuden tunnistaa (Laatukeskus 2010, 4 – 8), näitä tunnuspiirteitä ovat:

*Arvon tuottaminen asiakkaalle.* Organisaatiot tunnistavat ja tietävät, että asiakkaat ovat niiden olemassaolon syy. Asiakas on aina laadun lopullinen arvioija. Asiakas ei aina ole oikeassa, mutta yrityksen toiminnan rahoitus tulee kuitenkin asiakkailta, joten tuotteiden ja niiden takana olevien prosessien tulee kyetä vastaamaan asiakkaan tarpeisiin muuttuvissa tilanteissa.

*Visionäärinen, innostava ja vastuullinen johtajuus.* Johtajat muokkaavat tulevaisuutta ja saavat asiat tapahtumaan toimimalla organisaation arvojen ja eettisyyden esikuvina. Ylin johto on henkilöstöisesti sitoutunut laatu työhön, sitä ei ole delegoitu lautupäälliköille tai erilliselle laatuorganisaatiolle. Tavoitteet ja strategiat johdetaan yrityksen perusarvoista ja johto suorittaa säännöllisesti laatu katselmuksia.

*Prosessijohtaminen.* Päätöksenteko perustuu tosiasioihin ja toimintaa johdetaan strategian mukaisina prosesseina. Intuitio ja mutu-johtaminen ovat taka-alalla. Johtamisjärjestelmän ja päätöksenteon tulee perustua todelliseen ja luotettavaan tietoon. Tietojärjestelmät keräävät tietoa ja yrityksellä on välineet ja prosessit tiedon arvioimiseksi ja hyödyntämiseksi.

*Henkilöstö menestyksen tekijänä.* Organisaatiot arvostavat henkilöstöään ja luovat valtuuttavan kulttuurin, jossa organisaatiokohtaiset ja henkilökohtaiset tavoitteet ovat tasapainossa. Henkilöstö tekee laadun. Henkilöstön merkitys on viety juhlapuheita pidemmälle. Henkilöstöä ei nähdä kustannustekijänä, josta laskusuhdanteen aikana olisi päästävä joustavasti eroon, vaan voimavarana laadun ja asiakastyytyvyyden saavuttamiseksi.

*Luovuuden ja innovoinnin vaaliminen.* Organisaatiot tuottavat kasvavaa arvoa ja suorituskykyä jatkuvalla järjestelmällisellä innovoinnilla ja omaksumalla käyttöön sidosryhmien luovuuden. Laatutyöhön kuuluu jatkuva parantaminen. Mittavankaan uudistuksen jälkeen ei jäädä lepäämään.

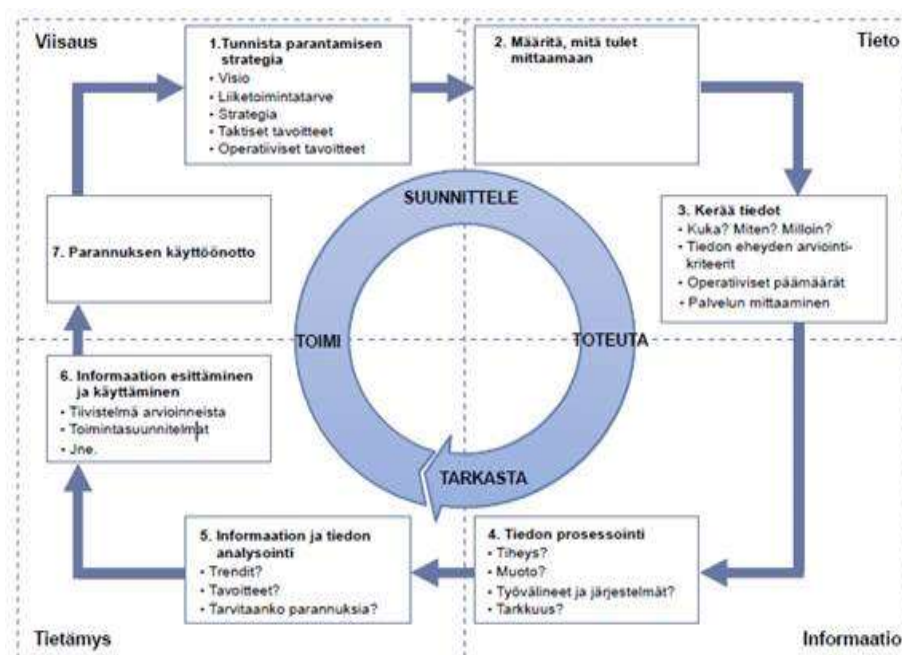
*Kumppanuuksien rakentaminen.* Organisaatiot kehittävät, hakevat ja ylläpitävät luottamukseen perustuvia suhteita erilaisten kumppanien kanssa. Hyvä yhteistyökyky sekä sisällä että suhteessa ulkoihin sidosryhmiin. Yhteistyökumppanien kanssa pyritään aina kaikkia osapuolia hyödyttäviin liikesuhteisiin. Yhteistyötä voidaan tehdä myös kilpailijan kanssa valituilla alueilla.

*Vastuu kestävästä tulevaisuudesta.* Kulttuuriin kuuluvat eettinen ajattelu, selkeät arvot ja vaativammat organisaation käyttäytymisnormit, jotka mahdollistavat taloudellisen ja sosiaalisesti ja ekologisesti kestävä kehityksen. Korkea liikemoraali, turvallisuus- ja terveystieteiden huomioon ottaminen, ympäristöhaittojen syntymisen ja luonnonvarojen tuhlaamisen estämiseen on osa yrityksen toimintaa.

## 2.2 Mikä on laadunhallintajärjestelmä?

Laadunhallintajärjestelmästä (Quality Management System) käytetään useita nimityksiä esimerkiksi toiminnanohjausjärjestelmä, toimintajärjestelmä tai johtamisjärjestelmä. Laadunhallintajärjestelmällä tarkoitetaan yleensä pelkää laadunhallintaa, kun taas toimintajärjestelmässä on usein mukana esimerkiksi ympäristöjärjestelmä. Laadunhallintajärjestelmässä olennaista on se, että siinä on mukana toiminnan parantaminen eli järjestelmässä tulee olla silmukka, jossa kerätty tieto analysoidaan ja sen pohjalta tehdään johtopäätöksiä ja toimintaa parannetaan tulosten perusteella. Jos väitetään että yrityksessä on käytössä laadunhallintajärjestelmä, on se kuvattava ja sen mukaisesti on toimittava. Muuten laadunhallinta on sattumavaraista. (Pesonen 2007, 50-51)

Kuvassa 1 on esitetty ITIL (Information Technology Infrastructure Library) - prosessin mukainen 7 askeleen kehittämisprosessi, jota käytetään mm. ISO/IEC 20000 standardin jatkuvan parantamisen prosessissa.



Kuva 1 Seitsemän askeleen kehittämisprosessi (Wakaru Oy, 2015)

### 2.3 Kehittämistehtävän taustaa

Hyvä laatu merkitsee tuotteiden virheettömyyttä ja alhaisia laatukustannuksia ja niiden seurauksena kustannustehokkuutta. Laatu vaikuttaa yrityksen katteeseen ja kannattavuuteen positiivisesti. Laatu vaikuttaa myös suoraan markkinoilla. Tyytyväiset asiakkaat ovat yleensä uskollisia yritykselle ja lisäävät ostojen määrää ja viestivät positiivisesti myös potentiaalisille muille asiakkaille. Laadun seurauksena yrityksen asema vahvistuu markkinoilla. Pitkällä tähtäyksellä laatu merkitsee yrityksen eloonjäämistä ja työpaikkojen säilymistä (Lecklin 2002, 27).

Mediconsult kehittää ohjelmistotuotteita suomalaisen terveydenhuollon tarpeisiin. Kehittämistä ohjaa Euroopan lainsäädäntö ja direktiivit. Kansallisessa lainsäädännössä on laki terveydenhuollon laitteista ja tarvikkeista (629/2010) ja Council directive concerning medical devices (MDD 93/42/EEC) sekä sen lisäyksi (2007/47/EC). Näitä lakeja ja määräyksiä sovelletaan myös ohjelmistoihin. Kansallisten määräysten noudattamista valvoo Suomessa Valvira.

Terveydenhuollon ohjelmistokehityksessä laatustandardi (ISO 13485 – Medical devices – Quality management systems) on huomioitava yllä mainittujen lakien lisäksi mm. riskienhallintastandardi (ISO 14971), ohjelmiston elinkaaristandardi (IEC 62304:2006) sekä käytettävyyssstandardi (EN 62366). Nämä standardit on pyritty huomioimaan yrityksen laatuorganisaatiossa ja tuotekehityksessä.

#### 2.3.1 Työn lähtökohdat

Tuotekehitystiimi tekee potilastiedon arkiston ja sähköisen lääkemääräyksen tuotetta. Tuotteet myös yhteistestataan, joissain tapauksissa ristiintestataan ja tarkastetaan olennaisten vaatimusten vaatimustenmukaisuus ulkoisen konsulttien toimesta (Kanta 2017). Uusi major-julkaisu tähtää yleensä

uuden ominaisuuden (esimerkiksi laboratoriotulokset, kuvantamisen tiedot tai tahdonilmaisut) Kanta-arkistointia ajatellen.

Tuotekehitystiimiin kuuluu 12 henkilöä. Ohjelmistokehittäjiä on seitsemän, ohjelmistotestaaajia on kolme, tuoteomistaja ja tuotekehityspäällikkö. Tiimi käyttää ohjelmistokehityksessä scrum-menetelmää.

Tuotteiden paremmalla laadulla tavoitellaan kustannussäästöjä ja asiakastytyväisyys paranee vähe-  
nevien ohjelmavirheiden johdosta. Tietyn julkaisun ennustettavuus on myös yksi asia minkä odote-  
taan paranevan laadun paranemisen myötä. Nykyinen versionhallinta ja toiminnanohjausjärjestelmä  
luovat myös haasteen: se mitä ominaisuuksia julkaisussa on todellisuudessa mukana, joudutaan tar-  
kistamaan jokaisessa julkaisussa käsin. Toimintajärjestelmä (JIRA) ja jatkuvan integraation järjestel-  
mät (Jenkins, Git) eivät ole tehtävätasolla yhteydessä keskenään.

### 2.3.2 Laatumittarit

Laatua ei pystytä kehittämään, jos sitä ei voida mitata. Mittaaminen on keskeinen komponentti tässä työssä. Jos laatua ei voida mitata on kehittämistoimenpiteiden tarvetta mahdotonta arvioida. Tuotteen elinkaareen sisältyy erilaisia mitattavia kohteita. Laatumittareita voidaan periaatteessa kohdistaa kaikkiin tuotteen elinkaaren osiin varhaisesta suunnittelusta tuotteen käytöstä poistoon asti. (Pöyhönen 2010, 19.)

Ihmiset suhteuttavat helposti toimintansa mittareihin. Jos on määritelty, ettei katselmointipöytäkirja tarvitse tehdä, kun virheitä on löydetty alle 5, löytyy katselmoinneissa harvoin yli viittä. Tulos ei myöskään saa olla kiinni mittaajasta tai mittausajankohdasta, tulosten tulee olla myös toistettavissa. Mittareista pitää myös ymmärtää, mistä arvон muutokset johtuvat.

Ohjelmistokehitystä ei ole totuttu näkemään mitattavana ilmiönä ja sitä vastustetaan ikään kuin varmuuden vuosi. Mittaaminen luo osalle ihmisistä epävarman olon, ja syntyy asenneongelma. Ihmisten asenteiden muuttamista helpottaa se, että kaikilla on selvillä mittaustulosten hyväksikäyttötapa. Mittaamisen onnistumiseksi on tärkeää, että kaikki prosessiin osallistuvat ymmärtävät, että tarkoituksena on parantaa prosessia eikä esimerkiksi vertailla keskenään prosessiin osallistuvien työtehoa (Haikala ja Märijärvi 2004, 205).

Nykyisestä tuotannonohjausjärjestelmästä (JIRA) voidaan seurata tuotteessa olevien ohjelmavirheiden määrää ennen julkaisua ja julkaisun jälkeen.

Mitattaviksi laatumittareiksi valikoitui:

- Julkaisun aikataulujen toteutuminen
- Julkaistavan version ominaisuuksien määrä
- Tiedossa olevien avointen virheiden kokonaismäärä
- Tiedossa olevat korkeamman luokituksen (major)-virheet

- Kuinka monta ohjelmistovirhettä uusi versio korjaa
- Kuinka paljon versiossa oli katselmoimatonta koodia julkaisuhetkellä
- Montako virhettä havaittiin julkaisun jälkeen 10 päivän aikana
- Montako virhettä havaittiin julkaisun jälkeen 31 päivän aikana
- Montako havaituista virheistä oli ohjelmistovirheitä, jotka vaikuttivat sovelluksen toimintaan
- Koodikatselmoinnissa käytetty aika.

Pöyhänen ja Kylmä (2010, 24) toteavat, että mittaamalla ja analysoimalla pyritään varmistamaan tuotteen vaatimustenmukaisuus, laadunhallintajärjestelmän vaatimuksenmukaisuus ja ylläpitämään laadunhallintajärjestelmän vaikuttavuutta. Kaikkia kerättyjä laatumittareita ei julkaista tässä työssä julkisesti.

### 2.3.3 Kehittämisen- ja tutkimuskohteet

Työn aikana koko tuotekehitysprosessia suunnittelusta, toteutukseen, testaukseen ja lopulta julkaisuun täsmennettiin. Kehittämiskohteena oli tarkastelussa myös julkaisuprosessiin kuuluva aika. Työn aikana tutkittiin jokaisesta julkaisusta, paljonko toteutukseen, testaukseen sekä koodikatselmointiin käytettiin aikaa. Tutkimuksen kohteena oli myös se, saavutettiinkö koodikatselmoinnilla tai viranomaisten yhteistestauksella parempaa laatua tai oliko niillä muita vaikutuksia lopulliseen tuotteeseen.

### 2.3.4 Kehittämistehtävän tavoite ja tarkoitus

Opinnäytetyön tavoitteena oli kehittää tuotekehityksen tuotteiden julkaisuja niin, että tuotteiden laatu paranee ja julkaisu on mahdollisimman pitkälle automatisoitu prosessi. Tavoitteena oli myös tutkia, pystytäänkö ohjelmistorobotiikkaa hyödyntämään julkaisujen ja laadun parantamiseen.

## 2.4 Rajaukset

Mediconsultin laatukäsikirjan pääprosesseja ovat tuotekehitys, ohjelmisto- ja palvelutuotanto, testaus, ohjelmistotuki sekä myynti ja markkinointi.

Tämä työ ei pyri parantamaan ohjelmistotuen, myynnin tai markkinoinnin laatua, vaan keskittyy tuotekehitykseen, ohjelmistotuotantoon ja testaukseen. Ohjelmistotuotannon riskianalyysin vaikutukset laatuun on myös rajattu tästä työstä pois, koska riskianalyysiprosessista saisi kokonaan oman opinnäytetyön.

### 3 TERVEYDENHUOLLON OHJELMISTOKEHITYS

Ohjelmistojen rooli terveydenhuollon laitteissa on kasvanut jatkuvasti. Ohjelmistot ovat nykypäivänä aktiivisesti mukana ohjaamassa eri laitteiden toimintoja, laskemassa tai diagnosoimassa mitattua potilastietoa monimutkaisten algoritmien avulla. Päätökset potilaan hoidosta tai hoitamatta jättämisestä perustuvat lisääntyvässä määrin ohjelmistojen laskemiin tuloksiin (laboratoriotulokset, tehohoidon valvontalaitteet, kuvantamislaitteet). (Pöyhönen 2006, 11.)

Terveydenhuollon laitteita ja tarvikkeita koskevan direktiivin (MDD 93/42/EEC) ja sen nojalla annettujen säädösten ja määräysten mukaan terveydenhuollon laitteet on suunniteltava ja valmistettava siten, että ne eivät suunnitelluissa olosuhteissa ja käyttötarkoituksen mukaisesti käytettynä vaaranna potilaan terveydentilaa ja turvallisuutta eivätkä käyttäjän tai muun henkilön turvallisuutta ja terveyttä. Vaatimukset koskevat myös lääkintälaitteessa olevaa ohjelmistoa tai itsenäisesti toimivaa ohjelmistoa, kuten potilastietojärjestelmää.

Terveydenhuollon laitteiden ja tarvikkeiden valmistajien laadunhallintajärjestelmän vaatimustasoksi ei riitä ISO 9001-standardi. ISO 13485:2016 (Terveydenhuollon laitteet ja tarvikkeet: laadunhallintajärjestelmät) on maailmanlaajuisesti laajimmin käytetty johtamisjärjestelmästandardi terveydenhuollon laitteille ja tarvikkeille. Terveydenhuollon laitteiden ja tarvikkeiden menestyksekkäs suunnittelu ja tuotanto edellyttävät laadun johtamisjärjestelmää perustuen ISO 13485 -standardiin.

Standardi 13485:2016 koskee erityisesti terveydenhuollon laitteita ja tarvikkeita, ja se perustuu ISO 9001 -standardiin ja on linjassa sen kanssa, mutta sen vaatimustaso on korkeampi. ISO 13485:2016 sisältää tarkempia vaatimuksia verrattuna ISO 9001 -standardiin, jolla varmistetaan terveydenhuollon laitteiden erityispiirteiden huomioon ottaminen laadunhallintajärjestelmän rakentamisessa ja ylläpitämisessä. Toisaalta, ISO 9001 -standardissa on piirteitä, jotka puuttuvat ISO 13485 -standardista. Näistä eroavaisuuksista johtuen esim. valmistajalla, joka täyttää ISO 13485 -standardin, ei välttämättä ole ISO 9001:n mukaista laatuja järjestelmää, elleivät eroavaisuudet erikseen täyty myös tämän standardin osalta. Oleellinen ero on, että ISO 13485:n päämäärä on vaatimustenmukaisuuden varmistaminen, kun taas ISO 9001 myös korostaa jatkuvaa parantamista. Valmistajan koko tai tyyppi ei vaikuta vaatimukseen – pienellekin yritykselle on asetettu tismalleen samat vaatimukset kuin suurelle. Valmistajan pitää myös kattaa ulkoistetut toiminnot, eli vastuu niistä heijastuu laadunhallintajärjestelmään. Kaikkien yritysten on kuitenkin mahdollista tehdä rajauksia, mikäli jokin toiminta ei sitä koske – perustelu on kuitenkin dokumentoitava.

ISO 13485 -standardi on tehty käytännön työkaluksi: sen avulla erityisesti terveydenhuollon laitteiden ja tarvikkeiden valmistajat voivat varmistaa lainsäädännön turvallisuusvaatimusten täyttymisen ja asiakasvaatimukseen vastaamisen tuotteen elinkaaren eri vaiheissa. ISO 13485 -standardi ei tarjoa suoria ratkaisuja suunnitteluun. Se kuitenkin auttaa luomaan kehyksen, joka mahdollistaa erilaisten ratkaisumallien käytön riippuen toisaalta valmistajan koosta, tuotteen monimutkaisuudesta ja poten-

tiaalisista riskeistä sekä toisaalta laatuavoitteista, jotka valmistaja on itse asettanut omille toimintoihilleen ja laitteilleen. (Inspecta Oy. Terveystuotteen ja tarvikkeiden johtamisjärjestelmän sertifiointi)

Käytännössä ohjelmistojen turvallisuutta ja vaatimuksenmukaisuutta ei voida osoittaa valmiista ohjelmistosta, joten ohjelmistojen turvallisuuden arviointi ulotetaan valmiista ohjelmistosta sitä tuottavan prosessin arviointiin (suunnittelu, vaiheistus, validointi sekä riskienhallinta). Tämä on huomiotava silloin, kun ohjelmistoille asetetaan vaatimuksia, joiden täyttyminen osoitetaan suunnitteludokumentoinnin avulla. (Pöyhönen 2006, 15.)

Usein perinteinen ohjelmistotuotannon vesiputousmalli ei sovellu sellaisenaan terveydenhuollon ohjelmistokehitykseen, minkä takia tarvitaan yrityskohtaista räätälöityä mallia. Perinteinen vesiputousmalli on vaiheellinen ohjelmistotuotantoprosessi, jossa suunnittelu- ja toteutusprosessi etenee vaihe vaiheelta alaspäin, kuin vesiputouksessa (määrittely, suunnittelu, toteutus, integraatio, testaus, asennus ja ylläpito). Räätälöity malli kohdistuu pääasiassa lakisääteisten vaatimusten lisäämisestä suunnittelun lähtötiedoiksi, riskienhallinnan toimenpiteistä sekä jäljitettävän suunnitteludokumentoinnin kehittämistarpeista.

### 3.1 Lääkintälaitteen ohjelmistotuotanto

Usein terveydenhuollon ohjelmistot käsitetään lääkitelaitteeksi, vaikka sovelluksessa ei olisi rajapintoja tai antureita varsinaiseen potilaaseen. Pöyhösen (2006, 16 - 17) mukaan lääkitelaitteen ohjelmistotuotannon elinkaari sisältää neljä merkittävää eroa perinteiseen vesiputousmalliin verrattuna:

1. Valmistajan tulee kyetä osoittamaan ennen markkinoille saattamista, että ohjelmisto täyttää sille asetetut lakisääteiset vaatimukset.
2. Ohjelmistojen suunnittelu on toteutettava, siten että koko suunnittelun elinkaari katetaan riskienhallinnan toimenpiteillä. Riskianalyysi alkaa suunnittelun alkaessa ja loppuu, kun suunniteltu ohjelmisto siirretään tuotantoon.
3. Riskianalyysin on osoitettava vaatimusmäärittelystä ne vaatimukset, joilla ehkäistään haitallisten tapahtumien toteutumista.
4. Ohjelmiston vaatimustenmukaisuus arvioidaan ohjelmiston suunnitteludokumenteista. Arviointi kohdistuu suunnitteludokumentteihin ja ohjelmistoa tuottavaan suunnitteluprosessiin.

### 3.2 Yrityksen laatujärjestelmä

Toimintajärjestelmän päämääränä on toimia jatkuvan parantamisen työkaluna Mediconsultin toiminnan kannalta keskeisissä toiminnoissa. Käsikirjassa kuvataan toiminnan pääpiirteet sanallisesti sekä annetaan vastaus tai viittaus siihen, miten sovellettujen standardien tai direktiivien (93/42/EEC ja ISO 13485) asettamiin vaatimuksiin käytännössä vastataan. Toimintajärjestelmä kattaa koko Mediconsult Oy:n toiminnan ja kaikki sovellettujen vaatimuskriteerien vaatimuskohdat. Mitään ei ole rajattu pois.



Mediconsult käyttää riskien arvioinnissa riskianalyysityökalua, jonka avulla varmistetaan, että tuotteiden käytön riskit on huomioitu, niihin on varauduttu ja tehty riskien minimoimisen edellyttämät toimenpiteet. Tuotteiden riskien arvioinnin dokumentaatio kerätään osaksi valvottua dokumentaatiota ja tallenteita. Mediconsultin laatukäsikirja painottuu pääprosessien mallintamiseen. Pääprosesseja ovat tuotekehitys, ohjelmisto- ja palvelutuotanto, testaus, ohjelmistotuki sekä myynti ja markkinointi. Laatukäsikirja tähtää jatkuvan parantamisen työkaluksi yrityksen toiminnan kannalta keskeisissä toiminnoissa. Toimintajärjestelmä kattaa koko Mediconsult Oy:n toiminnan ja kaikki sovellettujen vaatimuskriteerien vaatimuskohdat.

Sisäisiä mitattavissa olevia laatutavoitteita (Mediconsult 2017, 12):

- Osaaminen ja pätevydet

Koko henkilöstön osaaminen ja pätevydet kartoitetaan kerran vuodessa tehtävällä Mediconsult Got Talent -kyselyillä. Osa kyselyistä on koko henkilöstölle: substanssiosaaminen, vuorovaikutustaidot, Mediatri, tuoteosaaminen sekä sertifikaatit. Jokaiselle yksikölle on oma yksikkökohtainen kysely: tuotekehitys, tekninen tiimi, ohjelmistotuki sekä myynti ja asiakkuudet. Osaamisen kartuttaminen suunnitellaan toteutustasolla koulutussuunnitelmassa. Pätevyksien hankkimista ja koulutusten toteutumista seurataan vuositasolla. Henkilöstö kirjaa itse oman, merkittävästi työtä edistävän lisäkoulutuksen koulutustietoihin. Intran henkilörekisterissä on erikseen kohta, jossa henkilökunta merkitsee koulutuksensa, käytyt muut kurssit ja koulutukset, saadut sertifioinnit ja saadut pätevydet.

- Toimintaprosessit

Yrityksen toimintaa ohjataan sisäisesti määritellyillä prosesseilla tukitoiminnoissa ja myynnissä sekä ohjelmistokehityksessä/-tuotannossa. Tikettien määrää seurataan jatkuvasti ja raportoidaan vuositasolla. Ohjelmistotuen ja palveluiden sekä tukipyyntöjen osalta raportoinnin johtoryhmälle tekee puolivuositain ja vuosittain ohjelmistotuen vastuullinen vetäjä. Ohjelmistokehityksen osalta tikettikehityksen esittelee puolivuositain ja vuositasolla kehitystiimien vastuulliset vetäjät. Myynnin osalta tikettikehityksen esittelee puolivuositain ja vuositasolla myyntijohtaja. Vähintään puolivuosi- ja vuositasolla seurataan ja päätetään jatkotoimenpiteistä ko. prosesseissa johtoryhmässä. Seurattavat parametrit ovat tikettimäärät sekä niiden läpimenoajat eri luokissa.

Ulkoisia laatutavoitteita ovat hyvä yrityskuva, asiakastyytyväisyys ja toiminnan kehitys. Muita seurannan arviointiin sopivia menetelmiä ja mittareita arvioidaan johtoryhmässä. Yrityskuvaa ja asiakastyytyväisyyttä selvitetään asiakaskyselyillä, joita lähetetään kaikille asiakasryhmille eri asiakassegmenteissä, kuten myös eri ammattiryhmille.

### 3.2.1 Tuotteen toteuttaminen tuotekehityksessä

Mediconsultin tuotekehitys jakautuu useampaan tuotelinjaan, jotka käyttävät muutamaa erilaista prosessimallia kehitystyössä. Suunnitteluksi katsottavat tuotekehitystapaukset ovat kokonaan uusien tuotemoduulien, uusien käyttötarkoitusten tai merkittävällä tavalla uudenaikaisiksi muutettavien tuotteiden luomiseen liittyviä. Merkittäviksi muutoksiksi katsotaan tilanteet, joissa ohjelmistotuotteen käyttötarkoitus, valmistumisprosessi, keskeiset alihankkijat tai jokin muu tuotteen turvallisuuteen ja toimivuuteen vaikuttava osatekijä muuttuu merkittävästi.

Tuotekehitys on jaettu eri projekteihin. Projekti hallitsee itse tikettien kehittämistä. Korkean tason priorisointi ja tavoitteet asetetaan koko tuotekehityksen yhteisissä roadmap palaverissa. Tarve ohjelmistokehitykselle havaitaan usein asiakasrajapinnassa, mutta se voidaan todeta myös kehittäjien toimesta. Asiakasrajapinnassa havaituille toiveille luodaan oman tyyppinen tiketti, joka linkitetään tuotekehityksen tikettiin, kun se luodaan.

Suuremmat kehitystehtävät joudutaan priorisoimaan eli ne lisätään projektin työlistalle, josta töitä poimitaan toteutettavaksi, kun se on resurssien puolesta mahdollista. Listalta toteutettavaksi poimitut asiat lisätään projektin omaan työjonoon. Pienemmät kehitystyöt, kuten virheiden korjaukset voidaan lisätä suoraan ohjelmistokehitystiketeiksi ilman priorisointia.

### 3.3 Tuotekehityksen käytössä oleva prosessimalli

Tässä kappaleessa on kuvattu lyhyesti Mediconsultin tuotekehityksessä olevaa kehityksen aikaista prosessimallia.

#### 3.3.1 Scrum

Scrum kuuluu ketteriin kehitysmenetelmiin, jotka perustuvat Agile Manifesto -julistuksen arvoihin ja periaatteisiin. Tämä julistus on ketterän sovelluskehityksen perusta ja se sisältää neljä arvoa sekä kaksitoista periaatetta, joita ketterät menetelmät noudattavat.

Arvot on määritelty Agile Manifeston (Agile Manifesto, 2001; Agile Alliance, 2017) suomenkielisillä sivuilla seuraavasti:

**Ketterän ohjelmistokehityksen julistus**

Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

**Yksilöitä ja kanssakäymistä** enemmän kuin menetelmiä ja työkaluja

**Toimivaa ohjelmistoa** enemmän kuin kattavaa dokumentaatiota

**Asiakasyhteistyötä** enemmän kuin sopimusneuvotteluja

**Vastaamista muutokseen** enemmän kuin pitäytymistä suunnitelmassa

Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.

Scrum on suunniteltu tukemaan monimutkaisten tuotteiden kehittämistä. Scrum-menetelmässä on kehitystiimi rooleineen, tapahtumat, tuotokset ja säännöt. Jokainen näistä elementeistä palvelee tiettyä tarkoitusta ja on tärkeä osa scrumin onnistumisesta. Scrumin säännöt sitovat yhteen roolit, tapahtumat ja tuotokset ja ohjaavat niiden välistä vuorovaikutusta. (Schwaber ja Sutherland 2011.)

Scrumtiimi (Scrum team) koostuu tuoteomistajasta, kehitystiimistä ja scrum masterista. Tiimi on monitaitoinen ja itseohjautuva. Itseohjautuva tiimi päättää itse sprintin työjärjestyksen ja sen, kuinka parhaiten tekee työnsä. Monitaitoisella tiimillä on työn tekemiseen vaadittava osaaminen. Scrumtiimi kehittää ja tuottaa tuotteita. (Schwaber ja Sutherland 2011.)

Sprintti on ajallisesti määritelty jakso, jonka aikana tuotetaan valmis, määritelmän täyttävä ja käytökelpoinen tuote. Sprintti koostuu sprintin suunnittelupalaverista (planning), päiväpalaverista (daily), kehitystyöstä, sprinttikatselmuksesta (review) ja retrospektiivistä (retrospective). (Schwaber ja Sutherland 2011.)

Tuoteomistaja (Product Owner) on henkilö, joka vastaa tuotteen kehitettävistä ominaisuuksista. Tuotteen omistajan tehtävänä on tehdä kaikki päätökset tuotteen ominaisuuksista ja toiminnallisuuksiin vaikuttavista seikoista sekä kehitysjonon sisällössä ja sen prioriteeteissa. (Schwaber ja Sutherland 2011.)

Kehitystiimissä on ammattilaisia, jotka muuttavat tuotteen kehitysjonon sisällön potentiaalisesti julkaisukelpoiseksi, valmiiksi tuotteeksi jokaisessa sprintissä. Kehitystiimin jäsenet osallistuvat tuotteen kehitykseen. Kehitystiimin optimaalinen koko on riittävän pieni. (Schwaber ja Sutherland 2011.)

Scrum-mestari (Scrum Master) vastaa siitä, että kaikki ymmärtävät ja käyttävät Scrumia. Scrum-mestari palvelee kehitystiimiä valmentamalla sitä itseohjautuvuuteen ja moniosaamiseen, opetta-

malla ja johtamalla kehitystiimiä luomaan korkean lisäarvon tuotteita, poistamalla esteitä kehitystiimin etenemisen tieltä, fasilitoi pyydettyä Scrumin tapahtumia sekä valmentaa kehitystiimiä, jossa Scrumia ei ole otettu käyttöön ja ymmärretty. Scrum-mestari auttaa myös scrumitiimin ulkopuolisia ymmärtämään hyödyllisiä toimintatapoja toimia scrumitiimin kanssa. (Schwaber ja Sutherland 2011.)

Sprint review tai sprinttikatselmus pidetään sprintin lopussa, jossa tarkastellaan kehitetty tuoteversio. Tuoteomistaja tunnistaa, mikä osa tuotteesta on valmista ja mikä mahdollisesti ei. Scrumitiimi ja sidosryhmät voivat keskustella siitä, mitä sprintissä kehitettiin. Sprinttikatselmuksessa voidaan esittää tuoteversiosta demo, jonka tavoitteena on saada palautetta, edistää keskustelua ja luoda pohjaa mahdolliselle seuraavalle sprintille. (Schwaber ja Sutherland 2011.)

Päivittäinen Daily-palaveri pidetään joka päivä ja sen kesto on 15 minuuttia. Palaveri synkronoi kehitystyön toiminnot ja laatii suunnitelman seuraavan 24 tunnin ajalle. Tämä tapahtuu tarkastelemalla työtä edellisen Daily-palaverin jälkeen ja ennakoimalla työtä, joka voitaisiin tehdä ennen seuraavaa Daily-palaveria. Tätä käytetään seuraamaan sprintin edistymistä ja tarkastukseen, että tavoite saavutettaisiin. Päivittäinen kokous parantaa viestintää ja poistaa muita kokouksia, tunnistaa kehittämisen esteet ja korostaa nopeaa päätöksentekoa sekä parantaa kehitystiimin tietotaidon tasoa.

### 3.4 Yrityksen tuotekehityksen ohjelmistot

Tähän kappaleeseen on kasattu Mediconsultin tuotekehityksessä käytettäviä ohjelmistoja, joilla on merkitystä tuotteiden laatuun.

#### 3.4.1 Atlassian Jira

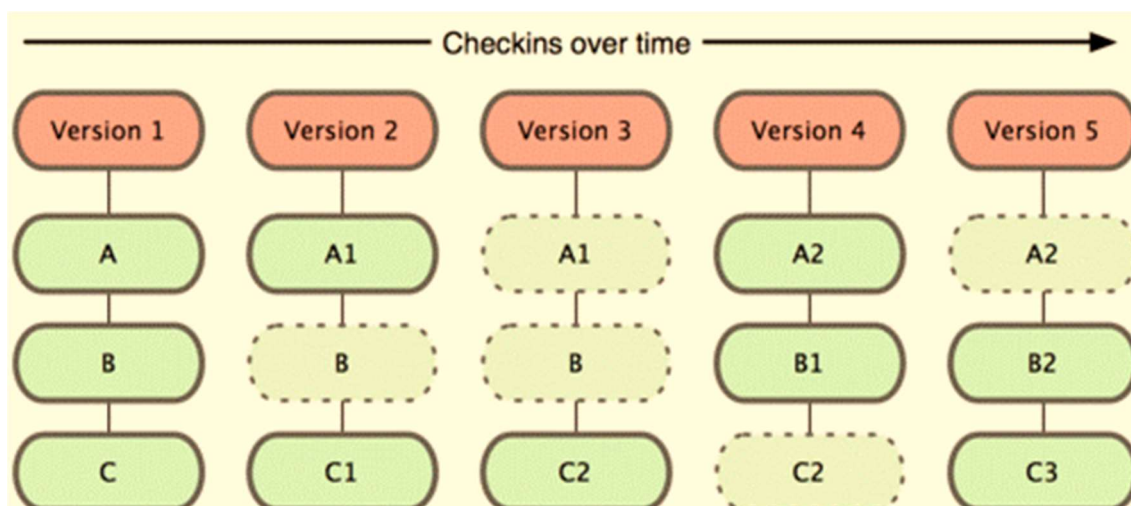
JIRA on tehtävienhallintaohjelmisto projektimuotoiseen työskentelyyn. Projektimuotoisessa työskentelyssä projektia ei yleensä voida tehdä kerralla valmiiksi, vaan se on syytä jakaa osatehtäviin, joiden suorittaminen saa aikaan lopullisen tuloksen. Tehtävienhallinnan roolina on huolehtia, että osatehtävät tulevat suoritetuksi ja kokonaisuus valmistuu. Tehtävienhallinnasta on hyötyä myös silloin, kun työskentely ei ole projektimuotoista. Työtehtävät saattavat olla irrallisia, mutta silti tarpeelliset suorittaa. Töiden kirjaaminen tehtävienhallintaan auttaa seuraamaan, että ne tulevat valmiiksi eivätkä unohdu. Tehtäviä voidaan linkittää toisiinsa, esimerkiksi niin että tehtävä X pitää olla tehtynä ennen tehtävä Y:tä. (Eduix 2017)

JIRA tehtävienhallintaohjelmistoa voidaan käyttää esimerkiksi projektien hallintaan ja erilaisten työmääräysten, virheiden ja tukipyyntöjen raportointiin. JIRAA voidaan käyttää selaimella, jolloin työskentely on aikaan tai paikkaan sitomatonta, tai integroida se perinteisesti olemassa olevaan ohjelmakantaan. JIRAan on saatavilla useita erilaisia lisäosia. (Eduix 2017)

JIRAn perustana ovat projektit, tehtävät ja työnkulut. JIRalla voidaan luoda tehtäviä, lähettää ne tiedoksi vastuullisille henkilöille, kirjata muistiin tehtäviin liittyviä tietoja ja siirtää tehtäviä työvaiheesta toiseen. Ohjelmistoprojekteissa JIRaa käyttämällä on helppo pitää lukua havaituista virheistä sekä seurata niiden käsittelyn etenemistä. (Eduix 2017)

### 3.4.2 Git

Terveysthuollon ohjelmistojen tuotekehityksessä tarvitaan versionhallintaa, koska jo pelkästään MDD direktiivi vaatii, että muutokset pitää pystyä jäljittämään. Git-versionhallinta on järjestelmä, joka ajan kuluessa tallentaa muutoksia tiedostoon tai joukkoon tiedostoja, jotta voidaan palata tiettyihin versioihin myöhemmin. Kun Gittiin tehdään pysyvä muutos (commit), Git ottaa periaatteessa kuvan siitä, miltä tiedostosi näyttävät kyseisellä hetkellä, ja varastoi viitteen tähän tilannekuvaan. Ollakseen tehokas, jos tiedostoa ei ole muutettu, Git ei varastoi sitä uudestaan - vaan linkittää sen edelliseen identtiseen tiedostoon, jonka se on jo varastoinut (Chacon ja Straub 2017).



Kuva 2. Git varastoi dataa tilannekuvina (Chacon ja Straub 2017).

Git on hajautettu versionhallintajärjestelmä mikä tarkoittaa sitä, että jokaisella käyttäjällä on käytössä paikallinen kopio versionhallinnasta. Käyttäjät tekevät omaan paikalliseen kopioonsa muutoksia ja lopulta liittävät (merge) oman kehityshaaransa (branch) projektin pääasialliseen versionhallintajärjestelmään.

Gitissä eri kehityshaarojen käyttäminen (esimerkiksi luominen, vertailu ja yhdistäminen) on kohtuullisen helppoa ja ideana on, että isommista muutoksista luodaan oma kehityshaaransa. Kehityshaaroja voidaan muokata toisistaan riippumatta, ja lopulta esimerkiksi vakaasta kehityshaarasta harhautetussa epävakaaassa kehityshaarassa tehty työ voidaan liittää takaisin vakaaseen haaraan työn valmistuttua.

Kehitystyö tapahtuu käytännössä viemällä muutoksia versionhallintaan eli tekemällä ns. committeja (englanniksi commit, vakiintunutta suomennosta ei ole). Gitissä jokaiselle commitille luodaan yksilöllinen SHA-1-tunniste, jolla siihen voidaan viitata. Tunnisteen avulla varmistetaan myös se, ettei tietokantaa voida vahingossa tai tahallaan korruptoida (Linux 2017).

### 3.4.3 Atlassian Crucible & FishEye

Crucible mahdollistaa versionhallinnan koodikatselmoinnin tai minkä tahansa asian katselmoinnin. Crucible työkalua käytetään myös riskianalyysin tekemisessä ja riskien arvioinnissa. Työkalu mahdollistaa lähettää usealle henkilölle katselmointiin ohjelmistokoodia tai riskianalyysin tavoitteita. Katselmoitavaa asiaa voidaan kommentoida ja antaa tavoiteaika mihin mennessä katselmointi pitää tehdä.

The screenshot shows the Atlassian Crucible interface for a code review. The title is "KANTA-1952: Maitohampaan korvaaminen hampaalla voisi merkitä maitohampaan...". The interface is divided into several sections:

- Details:** A table listing participants, their roles, time spent, and comments.
 

Participant	Role	Time Spent	Comments	Latest Comment
[Avatar]	Author	21m	2	Ei käytännössä. Mediatri muodostaa vain tietokantarivit a...
[Avatar]	Reviewer - Complete	13m		
[Avatar]	Reviewer - Complete	13m		
[Avatar]	Reviewer - Complete	28m	1	Onko näistä koodistoista eri versioita käytössä että kons...
<b>Total</b>		<b>1h 14m</b>	<b>3</b>	
- Objectives:** A section containing text describing the review goals and context, such as "Jos lapsipolilla maitohampaan korvaa tavallisella hampaalla, voisi adapteri arkistoida korvatus maitohampaan entyn SH2-tiedolla Koko hampaan puutos. Tällä hetkellä adapteri vain pudottaa maitohampaan entryn pois kokonaistarkastus-asialinjalla (+ tuo rautahampaan entryn uutena), jos samalle kohdalle on merkitty rautahammas. Ellei tehdä kokonaistarkastusta, maitohampaan viimeisin entry jää "elämään" vanhalta SH2-koodillaan HAM(K)asiakirjoista koostettuun-hammastietukseen - ja elämään jäänyt SH2-koodi on hyvin todennäköisesti virheellinen, ellei sitä ole ekapoliittisesti SH2-koodilla merkitty puuttavaksi."

Kuva 3. Crucible koodikatselmointi

FishEye on selaimessa toimiva lähdekoodin selain. Se tarjoaa käyttäjille kehittyneen selaamisen ja etsimisen Git versionhallinnasta. Crucible ja FishEye integroituvat suoraan JIRA-tehtävienhallintasovellukseen.

### 3.4.4 Jenkins

Sovellusta käytetään jatkuvassa integraatiossa, jolla voidaan tarkkailla esimerkiksi versionhallintaan koodin vientiä tai toistuvasta suoritettavia tehtäviä. Koodin vienti pystyy käynnistämään automaattisesti käännösprosessin (build). Jenkins on ilmainen ja lisensoitu MIT-lisenssillä (Laitinen 2011, 18). Laitinen kertoo konfiguroinnin ja ajojen luonnin hoituvan graafisen käyttöliittymän avulla. Jenkinsissä voi luoda ja nimetä erilaisia näkymiä. Näkymät on tarkoitettu Jenkins-ajojen kategorisointiin ja hallintaan. Jenkinsajoja voi vapaasti siirrellä näkymältä toisille ja tietty ajo voi näkyä usealla eri näkymällä (Laitinen 2011, 34). Sovellukseen voidaan määritellä erilaisia ominaisuuksia. Tällaisia ovat esimerkiksi yhteistoiminta versionhallinnan toimenpiteiden kanssa, ajojen laukaiseminen tai ajastus tietyin ehdoin sekä parametrit voi välittää toisesta ajosta eteenpäin. Ajoihin voidaan liittää ehdollista logiikka, jolla voidaan tarpeen mukaan säädellä ominaisuuksia.

Package	Kesto	Epäonnistui	Ohitettiin	Pass	Yhteensä
test.mediconsult...	1 min 33 sec	0	1	35	36
test.mediconsult...	6,8 sec	0	2	2	2
test.mediconsult...	1 min 32 sec	0	0	35	35
test.mediconsult...	5 min 7 sec	0	6	571	577
test.mediconsult...	24 sec	0	127	127	127
test.mediconsult...	1 min 35 sec	0	3	97	100
test.mediconsult...	17 sec	0	41	41	41
test.mediconsult...	15 min	0	1	276	277
test.mediconsult...	5 min 41 sec	0	0	147	147
test.mediconsult...	52 sec	0	0	35	35
test.mediconsult...	41 sec	0	1	95	96
test.mediconsult...	0,81 sec	0	0	1	1
test.mediconsult...	29 sec	0	1	33	34
test.mediconsult...	20 min	0	16	397	413

Kuva 4. Jenkins yksikkötestien nightly-näkymä

Mustaniemi (2013, 22) esittelee yleisen Jenkinsin käyttötavan, nightly buildin. Nightly build-termiä käytetään, kun Jenkins-ajo ajastetaan käynnistymään yöllä hakemaan uusimmat lähdekoodit versiohallinnasta ja kokoamalla ne paketiksi, jota vasten suoritetaan testejä.

### 3.4.5 Slack

Slack on hajautettujen ryhmien ja organisaatioiden sisäiseen viestintään suunnattu pikaviestintäsovellus. Slack-ryhmän sisällä voi keskustella sekä yksityisesti yksittäisen käyttäjän kanssa, että ryhmäviestiä usean käyttäjän kanssa tai perustaa kanavan, jolla keskustelu on tarkoitus suunnata esimerkiksi tietyn aihealueen pariin. Slackiin on mahdollista jakaa kuvia tai liitetiedostoja.

Isossa toimistossa Slack vähentää turhaa kävelemistä työpisteiden välillä, ja etätyötä tehdessä työkaverit ovat kätevästi muutaman napinpainalluksen päässä. Henkilökunta voi liittyä haluamalleen kanavalle ja seurata sekä kommentoida kanavalla käytävää keskustelua. Esimies tavoittaa oman tiimensä välittömästi. Kanaville voi myös tehdä hälytyksiä, jolloin kanavalle liittyneet ihmiset saavat ilmoituksen näytölleen tai kohdistaa viestin vain yhdelle ihmiselle.

Slackin on saatavilla eri liitännäisiä esimerkiksi Jenkinsiin, jolloin Slackiin saadaan ilmoituksia esimerkiksi, jos ohjelmiston käännös on epäonnistunut. Slackiin voi tehdä myös omia sovelluksia, joten se toimii eräänlaisena käyttöliittymänä käyttäjien ja tietojärjestelmien kesken.

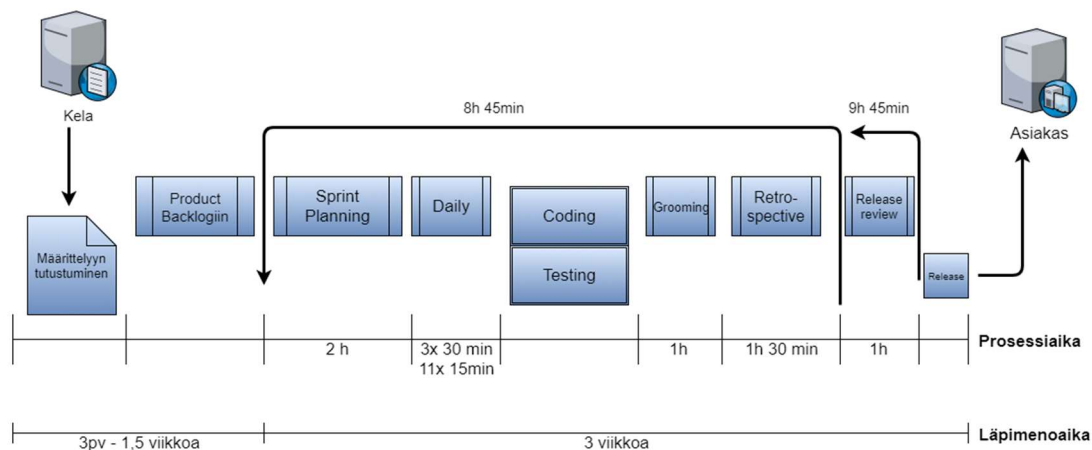
### 3.4.6 Testauksen työkalut

Testaus käyttää työpöytäsovellusten testauksen automatisointiin Test Complete -sovellusta. Testauksen dokumentointiin käytetään JIRA-järjestelmää sekä TestLink -sovellusta. Selainpohjaisten tuotteiden testaukseen käytetään Watir, Ruby sekä Protractor-sovelluksia.

## 4 OHJELMISTOTUOTTEEN VALMISTUSLAATU

### 4.1 Valmistuksen arvovirtakuvaus (Value Stream Mapping)

Työssä kuvattiin tiimin tuotekehityksestä arvovirtakuvauksen (value stream mapping). Arvovirtaan kuvataan kaikki toimenpiteet, jotta palvelu voidaan toimittaa asiakkaalle. Arvovirtakuvausta käytetään prosessin esteiden tunnistamiseen ja priorisointiin.



Kuva 5. Valmistuksen arvovirtakuvaus

Scrum tuotekehitykseen kuuluu erilaisia prosessivaiheita 3 viikon työjakson (Sprint) aikana neljä. Sprint planning, Daily scrum, Sprint Grooming sekä Sprint Retrospective. Sprint planning ja retrospective on vakiokestoisia (2h ja 1h 30min). Kolmena maanantaina Daily scrum on 30 minuutin mittainen ja kaikkina muina työpäivinä 15 minuuttia. Sprint Grooming pidetään juuri ennen sprintin vaihtumista ja se on tunnin mittainen. Prosessiajaksi saadaan tällöin 8h 45min, kun scrum-prosessin (Sprintin kesto) on vakio 15 työpäivää. Sprinttejä toistetaan niin monta, kunnes täysin valmis tuote voidaan julkaista.

Arvovirtakuvausta voidaan hyödyntää uusien työntekijöiden perehdytyksessä. Arvovirtakuvauksesta voidaan arvioida suorituskykyyn ja laatuun vaikuttavia asioita.

#### 4.1.1 Valmistuksen prosessin suorituskyky

Laatuun liittyy olennaisesti suorituskyky. Prosessin suorituskyvyn parantaminen on yleisin suorituskyvyn kehittämismotiivi. Prosessissa ilmenee yleensä virheitä, joita pyritään vähentämään, kohottamaan samalla suorituskykyindeksiä sekä pienentämään laatukustannuksia. Suorituskyky syntyy prosessiin osallistuvien henkilöiden työstä. Suorituskykyä parannettaessa puhtaasti teknisillä mittareilla ilmaistun tulosparannuksen lisäksi on huomioitava, että henkilöstö kokee muutokset mielekkääksi ja on valmis toteuttamaan ne itse. Tämä huomioitiin tätä työtä tehdessä.



Asiantuntijatyössä pelkkä läpimenoajan stabilointi on asiakkaan näkökulmasta niin suuri suorituskyvyn parannus, että muuta ei vähään aikaan osata edes vaatia. (Torkkola, 2016). Scrum-prosessi pyrkii jatkuvaan parantamiseen ja suorituskyvyn ja tehokkuuden lisäämiseen. Kolmen viikon työn (Sprint) jälkeen pidämme aina retrospektiivin (Sprint Retrospective), jossa kaikki pohtivat mikä meni hyvin, huonosti ja missä voisimme parantaa. Retrospektiivi pidetään Scrum-kehitysmenetelmän viimeisen suorituskykyyn tähtäävän periaatteen takia:

*Tiimin tulee selvittää säännöllisin väliajoin, miten se voisi tulla aikaisempaa tehokkaammaksi ja hienosäätää käytettävää työtapaa ja prosessia sen mukaisesti.*

Retrospektiiviä ei kuitenkaan pidetä Sprint Demon yhteydessä, vaikka sitä suositellaan useassa kirjallisuudessa.

The advantage of scheduling these two rituals on the same day is that the interruption in productivity associated with the rituals is minimized. The disadvantage for the team is that these days produce primarily scrum artifacts, as opposed to tangible product development. That tradeoff needs to be considered, and different teams will have different preferences. (Green 2016, 24)

Tätä ei noudateta, koska retrospektiivissa on havaittu tulevan tärkeitä asioita, että siihen varataan riittävästi aikaa eikä sitä yhdistetä toiseen palaveriin. Retrospektiivin kesto on 1.5 tuntia. Rakentavaa keskustelua syntyy aina ja tietoa jaetaan kaikkien kesken.

Suorituskyvyn parantamista voidaan tehdä (Torkkola, 2016):

- Lyhentää käytettävää aikaa, tehdään nopeammin.
- Lisää resursseja, joka lisää kustannuksia.
- Rajoita keskeneräisen työn määrää.
- Vähennä prosessin vaihtelua:
  - o Stabiiloi, muuta ennustettavaksi
  - o Vähennä virheitä

Tuotekehityksessä kuitenkin ajan lyhentäminen ei automaattisesti tarkoita nopeampaa, ellei huolehdi siitä, että pystytään tekemään samat asiat. Kokonaisaika lyhenee vain, jos tehokkuus kasvaa, mutta tehokkuus ei kasva suoraan aikaa lyhentämällä.

Keskeneräisen työn määrää voidaan rajoittaa Sprint Planning avulla, ettei Product Backlogista oteta uutta tehtävää tiimille. Prosessi on myös stabiili ja ennustettava, vakiomittainen. Työssä valittiin läpimenoajaksi 60 työpäivää, koska se on yleensä aika, minkä välein julkaistaan uusia ominaisuuksia ja sen aikana on virheet löydetty ja korjattu. Varsinainen julkaisukatselmointi lisää prosessin lopussa aikaa vain 1 tunnin.

Mediconsultilla nykyinen prosessiaika scrum prosessimallissa vie yhdeltä työntekijältä 60 työpäivän aikana 2160 minuuttia työaika eli 36 tuntia. Tiimin työntekijämäärä on 12 henkeä, joten todellinen työaika on 36 h x 12 henkilöä = 432 tuntia ns. prosessiaikaa.

## 4.2 Tuotelaatu

Tuotelaatu korostaa suunnittelun osuutta tuotteen laadun määräyksessä. Tuotelaadulla tarkoitetaan niitä ominaisuuksia, joita tuotteeseen rakennetaan sen käyttötarkoitusta silmällä pitäen. Suunnittelu-keskeinen laatu on asiakkaan kannalta lupaus laadusta ja siten vaikuttaa siihen, millaisen hinnan asiakas on valmis tuotteesta maksamaan.

Scrum-prosessissa varsinaista tuotteen suunnittelua tehdään pitkin sprinttiä. Prosessissa on kuitenkin Sprint Planning kokous, jonka kesto Mediconsultilla on 2 tuntia. Työssä laskettiin taulukkoon 1 kaksi erilaista vaihtoehtoa neljän sprintin (60 työpäivän) ajalta, miten eri kestoiset kokoukset vaikuttaisivat prosessiaikaan ja periaatteessa sitä kautta suorituskykyyn eli lyhennetään käytettävää aikaa.

TAULUKKO 1. Suunnitteluun ja kokonaisprosessiin käytettävä aika

Nykyinen prosessiaika			Planning lyhennetty			Planning, Grooming ja Retroa lyhennetty		
N		Prosessiaika (min)	1		Prosessiaika (min)	2		Prosessiaika (min)
1. sprint (15pv)	Sprint Planning	120	1. sprint (15pv)	Sprint Planning	90	1. sprint (15pv)	Sprint Planning	90
	Maanantain Daily Scrumit	90		Maanantain Daily Scrumit	90		Maanantain Daily Scrumit	90
	Ti-Pe Daily Scrum	165		Ti-Pe Daily Scrum	165		Ti-Pe Daily Scrum	165
	Grooming	60		Grooming	60		Grooming	45
	Retrospective	90		Retrospective	90		Retrospective	60
2. sprint (30pv)	Sprint Planning	120	2. sprint (30pv)	Sprint Planning	90	2. sprint (30pv)	Sprint Planning	90
	Maanantain Daily Scrumit	90		Maanantain Daily Scrumit	90		Maanantain Daily Scrumit	90
	Ti-Pe Daily Scrum	165		Ti-Pe Daily Scrum	165		Ti-Pe Daily Scrum	165
	Grooming	60		Grooming	60		Grooming	45
	Retrospective	90		Retrospective	90		Retrospective	60
3. sprint (45pv)	Sprint Planning	120	3. sprint (45pv)	Sprint Planning	90	3. sprint (45pv)	Sprint Planning	90
	Maanantain Daily Scrumit	90		Maanantain Daily Scrumit	90		Maanantain Daily Scrumit	90
	Ti-Pe Daily Scrum	165		Ti-Pe Daily Scrum	165		Ti-Pe Daily Scrum	165
	Grooming	60		Grooming	60		Grooming	45
	Retrospective	90		Retrospective	90		Retrospective	60
4. sprint (60pv)	Sprint Planning	120	4. sprint (60pv)	Sprint Planning	90	4. sprint (60pv)	Sprint Planning	90
	Maanantain Daily Scrumit	90		Maanantain Daily Scrumit	90		Maanantain Daily Scrumit	90
	Ti-Pe Daily Scrum	165		Ti-Pe Daily Scrum	165		Ti-Pe Daily Scrum	165
	Grooming	60		Grooming	60		Grooming	45
	Retrospective	90		Retrospective	90		Retrospective	60
Yhteensä		2100	Yhteensä		1980	Yhteensä		1800
Julkaisu	Release Review	60	Julkaisu	Release Review	60	Julkaisu	Release Review	60
	Release			Release			Release	
Prosessiaika (min)		2160	Prosessiaika (min)		2040	Prosessiaika (min)		1860
(h)		36	(h)		34	(h)		31
Tehostus-%		0	Tehostus-% vs. N		5,88	Tehostus-% vs. 1		9,68
						Tehostus vs. N		16,13

Vaihtoehdossa 1 laskettiin, jos jokaisen sprintin suunnittelukokousta lyhennettäisiin 30 minuuttia, yhden työntekijän kokonaisaika säästyisi 2 tuntia eli 5.88%, koko tiimin työaika säästyisi tuolloin 24 tuntia.

Vaihtoehdossa 2 työssä kokeiltiin radikaalimpaa ratkaisua, missä suunnittelukokouksen aikaa vähennettiin 30 minuuttia, grooming 15 minuuttia sekä retrospektiiviä 15 minuuttia. Yhden henkilön kokonaisaika vaihtoehdossa 2 säästyisi 5 tuntia, eli koko tiimin työaika säästyisi 60 tuntia.

Laskennat tehtiin elo-syyskuun aikana, mutta niitä ei jalkautettu tiimin toimintaan, koska työssä epäiltiin prosessimuutoksen hyödyllisyyttä. Kokousten lyhentäminen ei vaikuta suorituskykyyn tai se voi haitata siihen negatiivisesti. Ohjelmistotuotannossa informaatio on keskeistä ja kokouksissa jaetaan informaatiota. Jos viestintä ei toimi ja tiimillä on eri käsitys tai puutteellinen käsitys asioista, siitä seuraa tehottomuutta, vaikka aikaa olisi yhdessä kohdassa säästetty.

### 4.3 Valmistuslaatu

*Valmistuslaatu* keskittyy valmistusprosessiin ja varmistaa tuotteiden valmistuksen määritysten mukaan. Ohjelmistotuotannossa valmistuslaaduksi voidaan käsittää tehtävän varsinainen koodausvaihe ja muut ovat tuotelaatua. Prosessia kehittämällä virheet pyritään ennakoimaan ja välttämään. Suunnittelun jälkeen alkaa usein tarkempi suunnittelu tai varsinainen koodaaminen. Työn aikana otettiin kesäkuussa käyttöön JIRA-järjestelmään uusi uimarata Testing-sarake ja Quality Assurance (QA)-tila, muutos pyrkii parantamaan laadunvarmistusta ja tekemään testausprosessin näkyvämmäksi. Aiemmin testaus suoritettiin In Review -tilassa, joka käsitti sekä koodikatselmoinnin että testauksen. Tällä pystytään myös erottamaan tarkemmin koodin katselmoiintin kuluva aika ja testausaika toisistaan. Erottelulla pystytään ennakoimaan paremmin testauksessa kuluva aika, koodikatselmoinnissa käytettyä aikaa ja sitä kautta uuden version julkaisupäivää.

Unmapped Statuses	To Do	In Progress	Review	Testing	Done
Drag statuses below to a column to map them  Statuses not containing issues BACKLOG No issues IN PLANING No issues IN QA No issues PLANNING No issues SELECTED FOR DEVELO... No issues WAITING CUSTOM... No issues	Drag to rearrange, or delete  TO DO 1 issue OPEN 557 issues REOPENED 1 issue	Drag to rearrange, or delete  IN PROGRESS 22 issues	Drag to rearrange, or delete  IN REVIEW 15 issues	Drag to rearrange, or delete  QA 6 issues	Drag to rearrange, or delete  RESOLVED 499 issues DONE No issues CLOSED 1828 issues

Kuva 6. Valmistusprosessin eri vaiheet ja uusi Testing-vaihe.

Tuotteen suunnittelu ja valmistus etenevät vasemmalta oikealle: To Do – In Progress – Review – Testing – Done. Jos testauksessa havaitaan ohjelmistovirhe, palautetaan tehtävä In Progress -tilaan ja se etenee taas siitä eteenpäin oikealle. Review-tilaa käytetään silloin, kun työstettävä asia koodikatselmoidaan kehittäjien kesken. Valmistuslaatua valvotaan koodin crucible-ristiinkatselmoinnilla, jolloin saadaan useamman ihmisen näkemys siitä mitä on tehty. Tavoitteena on katselmoida kaikki

muutokset ja uudet ominaisuudet mitä tehdään. Testauksen voi aloittaa koodikatselmoinnin kanssa samanaikaisesti.

Sprintin daily-palavereihin otettiin syyskuussa käytännöksi, että tehtävät heijastetaan kaikkien nähtävillä videotykillä ja rajataan (Quick filters) henkilöön, joka on äänessä dailyn aikana. Muutoksella pyrittiin helpottamaan henkilöiden työjonon näkyvyyttä tiimin kesken, jotta kaikki ei ole vain muistin varassa näin ison tiimin kesken.

Kuva 7. JIRA-näkymä Sprintin tehtävistä ja Quick filters.

Tämän työn aikana siirryttiin käyttämään uutta versionhallintajärjestelmää (Git). Tiimi tunnisti retrospektiivissä lokakuussa, että valmistusprosessi on epäselvä. Tästä tehtiin liitteen 1 mukainen prosessikuvaus.

#### 4.4 Asiakaslaatu

*Asiakaslaatu.* Asiakkaiden tarpeet tyydyttävä laatu on hyvää laatua. Tutkimuksen aikana viidessä eri tuote versiossa huhtikuusta - marraskuuhun 2017 asiakas raportoi ainoastaan kesäkuussa yhden ja marraskuussa kaksi ohjelmistovirhettä. Yhteensä eri ominaisuuksia julkaistiin tällä aikavälillä 190 kappaletta.

Asiakaslaatuun vaikuttaa myös aikataulujen paikkansapitävyys. Erityisesti Kanta-adapterin versio 4.2.0 aiheutti hyvän asiakaslaadun kannalta poikkeaman. Tämän version ongelmia on käsitelty myöhemmin tässä työssä. Asiakastytyväisyyskyselyä ei suoritettu tämän työn aikana.

## 5 TUTKIMUSKOHTEET JULKAISUISSA

Työn aikana julkaisimme tuotteestamme 5 uutta versiota, joista 2 versiota oli pääversioita. Jokaisesta julkaisusta tutkittiin, jokaisen tehtävän kohdalta seuraavat asiat:

- montako päivää tehtävän suorittamiseen kului siitä, kun se otettiin tehtäväksi ja se valmistui
- kuinka monta päivää tehtävä oli työjonossa ennen kuin se otettiin työn alle
- kuinka monta päivää tehtävä oli tehtävänä ohjelmistosuunnittelijalla
- kuinka monta päivää tehtävä oli tehtävänä testauksessa
- montako minuuttia kului koodin katselmoinnissa aikaa.

Versioiden sisältö ja laajuus vaihtelivat pääversiosta pikaisesti tehtävään hotfix-korjaukseen. Jokaisen version sisältö tutkittiin erikseen tehtävä kerrallaan.

### 5.1 Tuotteen julkaisu

Tuote pyritään julkaisemaan ennalta sovittuna päivänä ja järjestämään julkaisukatselmointi (Release review). Julkaisupäivä sovitaan vasta sitten, kun tuote alkaa olla lähes julkaisuvalmis. Varsinainen asennuspaketti eli tuote koostetaan Jenkins nightly build-prosessilla, jolloin ajetaan automaattiset yksikkötestit. Tämän jälkeen tuote asennetaan tuotekehityksen testausympäristöön ja testataan vielä päätoiminnot (Smoke-test) manuaalisesti.

Julkaisukatselmointiin kutsutaan erilaisia sidosryhmän edustajia esimerkiksi teknisestä tiimistä, ohjelmistotuesta, käyttöönottoprojektien henkilöitä sekä lääketieteellinen johtaja. Tuotteen omistaja esittelee valmiin tuotteen ja sen tärkeimmät ominaisuudet sekä tiedossa olevat virheet ja mahdollisen riskianalyysin päätulokset ja käyttöönoton edellytykset. Julkaisukatselmoinnissa päätetään, julkaistaanko tuote asiakkaiden saataville. Sidosryhmien edustajat voivat esittää kysymyksiä julkaistavasta tuotteesta. Julkaisukatselmoinnista tehdään pöytäkirja, jossa on

- Julkaisun yleiskuvaus sisällöstä
- Konfiguraatioon tarvittavat muutokset
- Rajapintojen muutokset
- Muutokset yhteiskäyttöisiin komponenttikirjastoihin
- Riippuvuudet muihin tuotteisiin tai ohjelmistoversioihin
- Koodikatselmoinnin tulokset
- Riskianalyysin katselmointi
- Testauksen katselmointi
- Tiedossa olevien ohjelmavirheiden lista prioriteetteineen.
- Dokumentaation katselmointi
- Julkaisupäätös
- Muutosluettelo.

Julkaisukatselmoinnin jälkeen tuotteesta on saatavilla tarvittavat asennuspaketit ja se viedään asiakkaille sovittujen aikataulujen mukaan. Tuotekehitys ei ole enää tämän jälkeen mukana uuden version asiakkaiden käyttöönotoissa, lukuun ottamatta poikkeustilanteita. Uuden version päivitys aiheuttaa asiakkailta lyhyen käyttökaton sähköisen reseptin toiminnassa, joten päivityksistä joudutaan sopimaan erikseen.

## 5.2 Kanta-adapteri versio 4.2.0

Versio oli tarkoitus julkaista 28.4.2017, mutta se voitiin julkaista vasta 5.5.2017 eli 7 päivää myöhässä, välissä oli myös arkipyhä. Tässä julkaisussa työssä kokeiltiin, miten versioon vaikuttaa, jos ohjelmistosuunnittelijoille annetaan tiukka aikaraja (28.4.2017), milloin työt pitää olla tehtynä. Version testauksessa löytyi kahta päivää julkaisua ennen sähköisestä lääkemääräyksestä ohjelmistovirhe, joka esti käyttöä, joten julkaisuaikataulu siirtyi. Samaa virhettä jouduttiin korjaamaan kahteen kertaan.

Seuraava versio 4.2.1 päätettiin julkaista 6.6.2017, koska versioon oli tulossa paljon asiakkaan toivomia ominaisuuksia ja niitä ei keretty tekemään tähän versioon. Tästä 4.2.1 versiosta tuli kuitenkin kiireellinen korjausversio 4.2.0:n tuotantokäytössä esiintyvien ongelmien takia.

Versio 4.2.0 päivitettiin asiakkaille tuotantoon 2.6.2017 eli 28 päivää julkaisun jälkeen. Annettu tiukka aikaraja ei olisi ollut tämän perusteella tarpeellinen. Pidempi aikaraja ei olisi myöskään vaikuttanut siihen, etteikö korjausversiota olisi jouduttu julkaisemaan.

TAULUKKO 1. Version 4.2.0 yhteenveto

	lukumäärä
Uusien ominaisuuksien lukumäärä	52
Tiedossa olevia virheitä julkaisussa	42
Korkeamman luokan virheitä	4
Katselmoimatonta koodia julkaisuhetkellä	7
Virhekorjauksia mukana versiossa	19
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 10 pv)	5
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 11-31 pv)	7

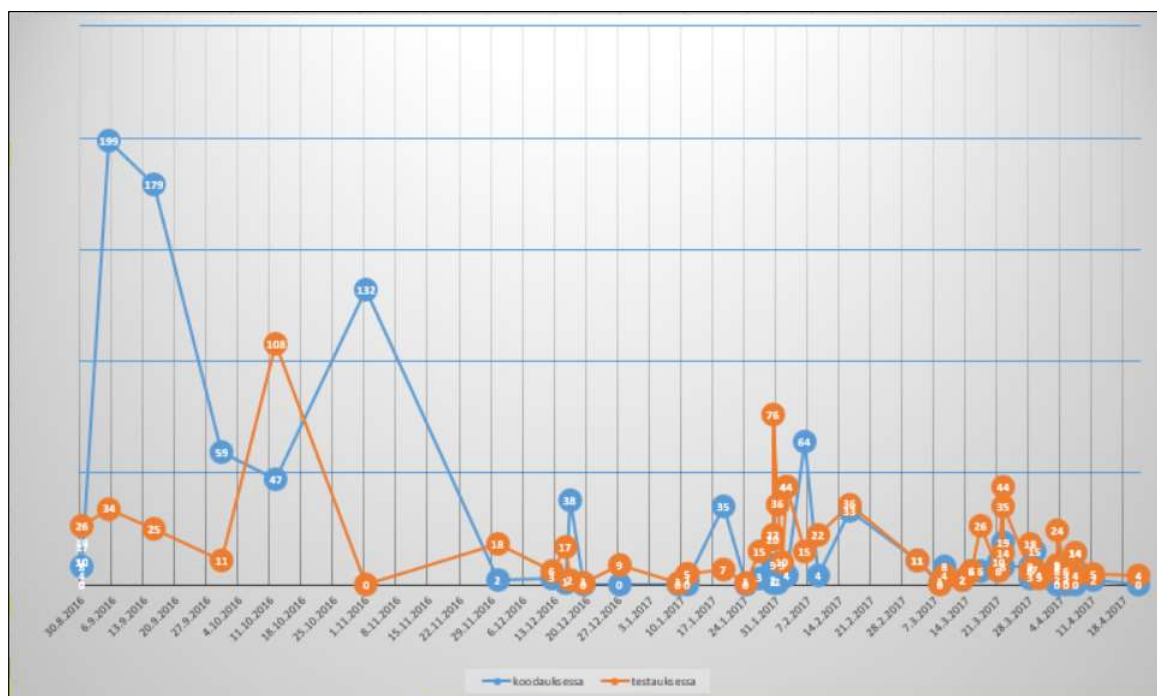
Versiossa piti ottaa käyttöön Kelan toteuttama 16kk sähköisen lääkemääräyksen haun aikarajaus (recentMedicationInformation), mutta Kelan asiakastestiympäristö ei toiminut oikein, jolloin jouduttiin palauttamaan hakuun vanhan toiminnan. Kelan asiakastestauksen ohjelmistovirheestä raportoitiin 26.4.2017. Virheen korjauksesta kysyttiin uudelleen 2.6.2017. Kela antoi vastauksen vasta 4.9.2017, jolloin ilmoitettiin, etteivät muuta toteutusta, vaan tulevassa lääkitysprojektissa on tavoitteena saada parempi näkyminen. Lääkitysprojektin aikataulua ei ole tiedossa (30.11.2017).

Versio 4.2.0 mahdollistaa sähköisen lääkemääräyksen kieltojen käyttöliittymän käyttöönoton, korjauksia rokotusrakenteeseen, kotikuntatiedon asetuksen oletuksena, parannuksia lokitustietoihin,

tiukennettiin toimikorttien tietoturva sekä useita pienempiä korjauksia. Versiossa oleva sähköisten lääkemääräysten kiellot vaativat Kelan yhteistestauksen. Yhteistestaus käsitti 17 testiketjua, joista osa testiketjuista testasi sähköisen lääkemääräyksen version taaksepäin, eteenpäin ja uuden määrittelyversion ominaisuuksia. Sähköisistä lääkemääräyksien kielloista ei ole löytynyt virheitä, mutta ominaisuutta ei ole vielä otettu käyttöön asiakkailta, joten luotettavaa tietoa siitä oliko yhteistestauksella vaikutusta laatuun ei ole toistaiseksi saatavilla.

TAULUKKO 2. Version 4.2.0 yhteenveto tehtävien suoritusajoista

	Keskiarvo	Mediaani
Tehtävän luonnista siihen, kun tehtävä oli valmis (päivää)	68	41
Tehtävän luonnista siihen, kun tehtävä otettiin työn alle (päivää)	35	16
Kuinka kauan tehtävää ohjelmoitiin (päivää)	19	4
Kuinka kauan tehtävä oli testauksessa (päivää)	17	10
Kuinka kauan tehtävä oli koodikatselmoinnissa (minuuttia)	77	24



Kuva 8. Version 4.2.0 työn alle ottopäivä vs. koodausaika ja testausaika

Kuvaajasta voidaan päätellä, että tässä versiossa elokuusta 2016 joulukuun alkuun on keskimääräinen koodausaika ja testausaika ollut reilusti keskiarvoa suurempi. Tämä ei kuitenkaan kerro suorituskyvystä, koska samaan aikaan tehtiin muitakin kuin versiota 4.2.0 ja kuvaaja tarkastelee vain tätä versiota.

### 5.2.1 Version käyttöönotto asiakkaalla

Version 4.2.0 käyttöönotto tehtiin 2.6.2017 pienemmille asiakkaille, versiosta ei raportoitu ongelmia. Version käyttöönotto edellytti tietokannan rakenteen päivitystä ja konfiguraatiomuutoksia. Uusi versio päivitettiin 6.6.2017 isolle perusterveydenhuollon asiakkaalle (A). Asiakas raportoi seuraavana

päivänä ongelman, mikä ilmeni terveydenhuollon varakorttien käytössä. Varakortilla ei pystynyt tekemään sähköistä lääkemääräystä. Versiossa oli tiukennettu tietoturvaan liittyen ammattikorttien käsittelyä. Ominaisuutta ei oltu testattu varakorteilla. Varakortteja ei myönnetä testaukseen, joten ominaisuutta ei edes pystytty testaamaan, eikä Mediconsultin terveydenhuollon ammattilaisilla ollut varakortteja. Mediconsultin omassa testauksessa 7.6.2017 löytyi myös virhe julkaisun jälkeen, joka vaikutti reseptien ulkomaanjäljennöksen tulostamiseen.

Asiakkaalle (A) päivitettiin 7.6.2017 takaisin edellinen versio, jossa ongelmaa varakorttien kanssa ei ollut. Versiosta lähetettiin muille asiakkaille tiedote, jossa kerrottiin löydetyistä virheistä ja korjausaikataulusta. Ongelmat varakorttien käsittelyssä ja ulkomaanjäljennöksessä saatiin pikakorjattua versioon 4.2.1.

### 5.3 Versio 4.2.1

Versio päätettiin tehdä heti pääversion 4.2.0 jälkeen, koska osa asiakkaista odotti erästä ominaisuutta tuotantokäyttöön. Versio julkaistiin suunniteltuna päivänä 6.6.2017.

TAULUKKO 3. Version 4.2.1 yhteenveto

	lukumäärä
Uusien ominaisuuksien lukumäärä	12
Tiedossa olevia virheitä julkaisussa	44
Korkeamman luokan virheitä	4
Katselmoimatonta koodia julkaisuhetkellä	5
Virhekorjauksia mukana versiossa	7
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 10 pv)	5
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 11-31 pv)	7

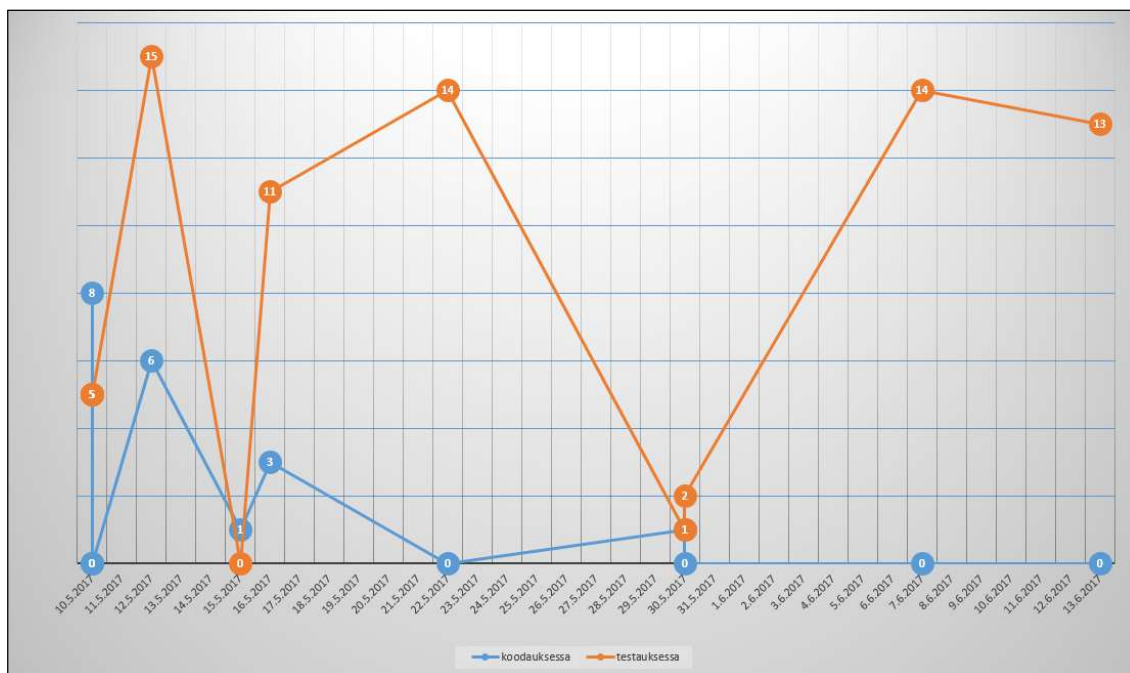
Versiossa oli korjattu myös 4.2.0 löytynyt ulkomaanjäljennöksen ohjelmistovirhe. Versio paketoitiin uudelleen 21.6.2017 ja päivitettiin yhdelle asiakkaalle 21.6.2017. Asiakas raportoi 26.6.2017 ongelman sähköisen reseptin kirjoituksessa. Ongelma paikallistettiin siihen, että asiakas käyttää kahta toimikortinlukijaa ja toimikorttia, joista toinen on terveydenhuollon toimikortti ja toinen organisaatiokortti. Ongelma korjattiin ja julkaistiin samana (26.6.2017) päivänä versioon 4.2.2.

TAULUKKO 4. Version 4.2.1 yhteenveto tehtävien suoritusajoista

	Keskiarvo	Mediaani
Tehtävän luonnista siihen, kun tehtävä oli valmis (päivää)	25	12
Tehtävän luonnista siihen, kun tehtävä otettiin työn alle (päivää)	10	5
Kuinka kauan tehtävää ohjelmoitiin (päivää)	1,6	0
Kuinka kauan tehtävä oli testauksessa (päivää)	8	5
Kuinka kauan tehtävä oli koodikatselmoinnissa (minuuttia)	33	17



Version julkaisutestauksen testitapauksia täydennettiin löydettyjen virheiden perusteella, jotta seuraavassa julkaisussa osattaisiin testata myös ne asiat, mitkä olivat nyt jääneet testaamatta.



Kuva 9. Version 4.2.1 työn alle ottopäivä vs. koodausaika ja testausaika

Kuvaajasta voidaan päätellä, että julkaisussa oli nopeasti tehtäviä korjauksia, jotka vievät paljon testausaika tai testauksessa oli ruuhkaa sekä resurssivajetta.

#### 5.4 Versio 4.2.2

Versio 4.2.2 julkaistiin 26.6.2017 ja oli hotfix versiosta 4.2.1. Versio piti sisällään ainoastaan kaksi uutta korjausta 4.2.1 versioon. Toinen korjauksista oli automaattisen virheenjäljitysohjelman (find-bugs) ilmoittama muutostarve ja toinen asiakkaan raportoima ongelma 4.2.1 versiossa terveydenhuollon varakorttien kanssa.

TAULUKKO 5. Version 4.2.2 yhteenveto, suluissa versio 4.2.1

	lukumäärä
Uusien ominaisuuksien lukumäärä	2 (12)
Tiedossa olevia virheitä julkaisussa	44
Korkeamman luokan virheitä	4
Katselmoimatonta koodia julkaisuhetkellä	1
Virhekorjauksia mukana versiossa	2
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 10 pv)	6 (5)
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 11-31 pv)	8 (7)

#### 5.4.1 Version käyttöönotto asiakkaalla

Versio päivitettiin kaikille asiakkaille 29.6.2017. Versio edellytti tietojärjestelmän tietokannan konfigurointia. Päivityksen yhteydessä tehtiin ensin konfigurointivirhe, jolloin tietojärjestelmä ei toiminut oikein.

#### 5.5 Versio 4.2.3

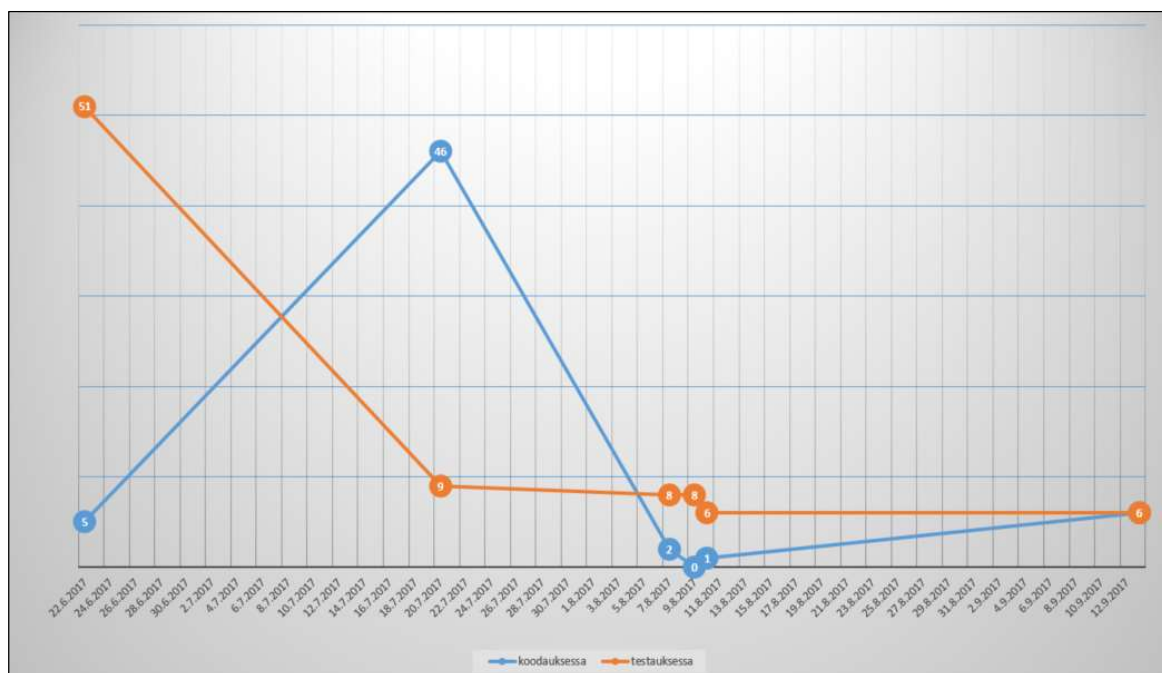
Tuotannosta oli raportoitu kesäkuussa virhe, joka esiintyi satunnaisesti muutamalla asiakkailla, joilla oli hitaat tietoliikenneyhteydet. Versioon 4.2.3 korjattiin tämä ja ongelma. Versio julkaistiin 14.9.2017, joka oli 6 päivää myöhässä suunnitellusta aikataulusta.

TAULUKKO 6. Version 4.2.3 yhteenveto

	lukumäärä
Uusien ominaisuuksien lukumäärä	6
Tiedossa olevia virheitä julkaisussa	31
Korkeamman luokan virheitä	2
Katselmoimatonta koodia julkaisuhetkellä	0
Virhekorjauksia mukana versiossa	2
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 10 pv)	2
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 11-31 pv)	1

TAULUKKO 7. Version 4.2.3 yhteenveto tehtävien suoritusajoista

	Keskiarvo	Mediaani
Tehtävän luonnista siihen, kun tehtävä oli valmis (päivää)	54	58
Tehtävän luonnista siihen, kun tehtävä otettiin työn alle (päivää)	31	32
Kuinka kauan tehtävää ohjelmoitiin (päivää)	10	4
Kuinka kauan tehtävä oli testauksessa (päivää)	15	8
Kuinka kauan tehtävä oli koodikatselmoinnissa (minuuttia)	78	57



Kuva 10. Version 4.2.3 työn alle ottopäivä vs. koodausaika ja testausaika

Kuvaajasta voidaan päätellä, että versiossa oli kaksi ominaisuutta, joista toinen vei paljon koodausaika ja toinen testausaika.

### 5.5.1 Version käyttöönotto asiakkaalla

Versio päivitettiin kaikille asiakkaille 26.-27.9.2017 ja se on toiminut moitteettomasti.

### 5.6 Versio 4.3.0

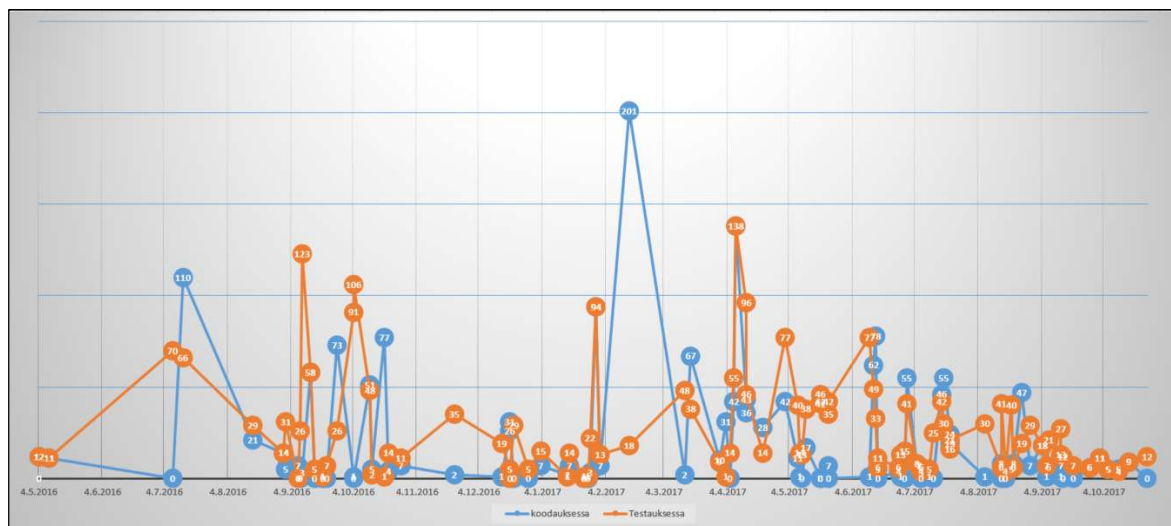
Versio piti sisällään isoja kokonaisuuksia, joita oli työstetty ja testattu useamman kuukauden aikana. Versio julkaistiin 1.11.2017. Versiossa oleva suun terveydenhuollon arkistointi vaati Kelan yhteistestauksen sekä asiakkaan tekemän ristiin testauksen.

#### TAULUKKO 8. Version 4.3.0 yhteenveto

	lukumäärä
Uusien ominaisuuksien lukumäärä	118
Tiedossa olevia virheitä julkaisussa	31
Korkeamman luokan virheitä	0
Katselmoimatonta koodia julkaisuhetkellä	4
Virhekorjauksia mukana versiossa	20
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 10 pv)	2
Löydetyt uudet virheet julkaisun jälkeen (ensimmäiset 11-31 pv)	8

TAULUKKO 9. Version 4.3.0 yhteenveto tehtävien suoritusajoista

	Keskiarvo	Mediaani
Tehtävän luonnista siihen, kun tehtävä oli valmis (päivää)	136	137
Tehtävän luonnista siihen, kun tehtävä otettiin työn alle (päivää)	96	16
Kuinka kauan tehtävää ohjelmoitiin (päivää)	18	6
Kuinka kauan tehtävä oli testauksessa (päivää)	26	15



Kuva 11. Version 4.3.0 työn alle ottopäivä vs. koodausaika ja testausaika

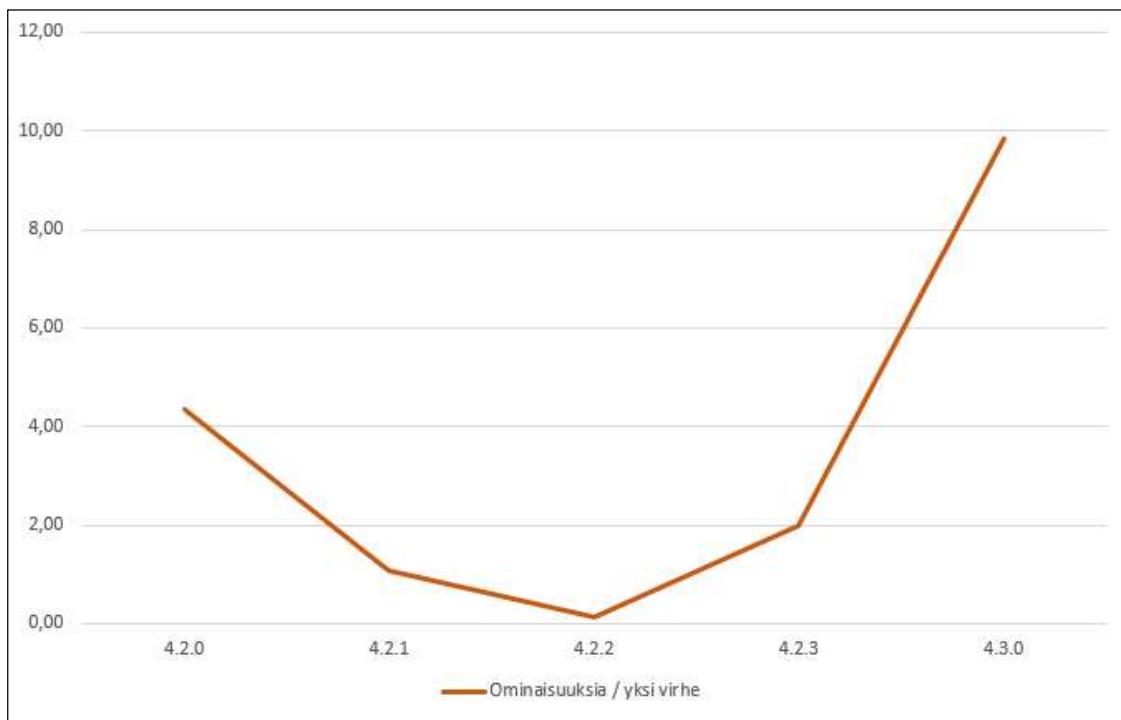
## 5.7 Yhteenveto eri mittareista ja laadusta julkaisuissa

TAULUKKO 10. Yhteenveto tuotteen julkaisuista

Päivä	Versio	Versiossa julkaistut ominaisuudet	Havaitut virheet julkaisun jälkeen (1kk aikana)	Katselmoimattomia koodia julkaisuuskellä	Ominaisuuksia / yksi virhe	Paljonko aikaa kului, kun tehtävä otettiin työn alle	Paljonko keskimäärin kesti tekeminen	Paljonko keskimäärin kesti testaaminen
5.5.2017	4.2.0	52	12	7	4,33	16	4	10
6.6.2017	4.2.1	12	11	5	1,09	5	1	5
26.6.2017	4.2.2	2	14	1	0,14	0	6	13
14.9.2017	4.2.3	6	3	0	2,00	32	4	8
1.11.2017	4.3.0	118	12	4	9,83	16	6	15
Keskiarvo						13,8	4,2	10,2

Todella karkean arvion laadusta saadaan, kun jaetaan ominaisuuksien lukumäärä virheiden lukumäärällä, jolloin saadaan suhdeluku, kuinka monta ominaisuutta tehtiin ennen kuin tehtiin yksi virhe. Versio 4.2.2 on kuitenkin poikkeus, koska versio tehtiin edellisen version virheiden takia, joten näiden kahden versioiden suhdeluvut voidaan summata, jolloin suhdeluvuksi saadaan  $1,09 + 0,14 = 1,23$ .

Tutkittuja työaikoja voidaan hyödyntää töiden ja julkaisujen suunnittelussa, kun tiedetään keskiarvot ja mediaanit. Tämän työn ja näiden versioiden perusteella menee 14 päivää, että työ voidaan ottaa työjonosta toteutettavaksi. Sprintin pituus (15pv) vaikuttanee tähän aikaan, koska yleensä sprinttiin ei lisätä kesken työjakson töitä. Toteutustyö testauksineen vie noin 15 päivää.



Kuva 12. Ominaisuuksien ja yhden virheen suhdeluku

Kaaviosta nähdään, että ominaisuuksien ja virheiden suhdeluku on parantunut eri versioiden myötä eli laadun parantamisen toimenpiteet ovat toimineet. Jos tarkastellaan eri versioista katselmoimattoman koodin määrää se korreloi suoraan löydettyjen virheiden kanssa sekä ominaisuuksien ja virheen suhdelukuun.



Kuva 13. Katselmoimatonta koodia prosentteina versioittain

## 6 KEHITTÄMISKOHTEITA LAADUN PARANTAMISEEN TERVEYDENHUOLLON OHJELMISTOKEHITYKSESSÄ

### 6.1 Koodin katselmointikäytännöt

Käytössä oleva julkaisuprosessi sallii sen, että katselmoimatonta koodia on julkaisuhetkellä, koska katselmoinnissa löytyy harvoin todellista korjattavaa, vaan muutokset liittyvät esimerkiksi ohjelmakoodin esitystapaan. Versiossa 4.2.2 oli katselmoimatonta koodia 50%, koska ominaisuuksia oli vain kaksi. Toinen katselmoitavista ominaisuuksista oli omissa testeissä löydettyjä pieniä korjauksia, joiden arvioitiin, etteivät ne estä julkaisua.

Tarkastellessa laatua edellisessä kappaleessa kuvassa 11 ja kuvassa 12 esitettyjen mittareiden avulla: katselmoimattoman koodin määrä vaikuttaa laatuun. Julkaisuprosesia pitäisi muuttaa niin, ettei julkaisua tehdä, jos versiossa on yhtään koodia katselmoimatta.

### 6.2 Uudet ominaisuudet potilastiedon arkistossa ja sähköisessä reseptissä

Kansalliset muutokset mitä toteutettiin, vaikuttivat version sisältöön ja julkaisun jälkeen löytyneisiin virheisiin. Ominaisuuksia ei kannata ottaa versioon mukaan, jos niiden testausta ei ole pystytty valimoimaan kansallisia määrittäyksiä ja toteutusta vasten tai toteutus on epäselvä.

Ennen kuin tuotekehitys lähtee toteuttamaan uusia ominaisuuksia kannattaa viranomaisilta varmistaa, että ominaisuus on testattu heidän toimesta ja heidän omissa testausympäristöissä. Nyt jouduimme poistamaan versiosta määrittysten mukaan tehtyjä ominaisuuksia, mitkä eivät toimineet viranomaisten ympäristöissä. Scrum filosofian mukaan arvostetaan enemmän asiakasyhteistyötä kuin sopimusneuvotteluja, niin tässä tapauksessa neuvottelu ja määrittysten kyseenalaistaminen olisi ollut paikallaan viranomaisten kanssa. Tosin viranomaisten kanssa kysymysten vastausajat ovat usein kohtuuttoman pitkiä.

### 6.3 Asennusohjeiden ja konfiguraatiomuutosten dokumentointi

Scrum-filosofia arvostaa enemmän toimivaa ohjelmistoa, kuin kattavaa dokumentaatiota asennusohjeet ja konfiguraatiomuutokset on dokumentoitava tarkemmin. Konfiguraatiossa oli tapahtunut virhe, joka esti asiakkaan käyttöönottoa. Tämä johtui siitä, että konfiguraatiomuutos oli kirjoitettu pienellä alkukirjaimella, kun se olisi pitänyt kirjoittaa Isolla alkukirjaimella. Julkaisussa kannattane ottaa käyttöön tiimin sisäinen dokumentaation katselmointi, ennen virallista julkaisukatselmointipalaveria.

### 6.4 Versiojulkaisussa tehtävät testitapaukset ja testausympäristöt

Version julkaisutestauksen testitapauksia kannattaa täydentää löydettyjen virheiden perusteella. Julkaisutestauksen testitapaukset kannattaisi automatisoida käännösprosessin yhteyteen. Manuaalisia testitapauksia on tällä hetkellä 33 kappaletta, jotka koskevat 5 tietojärjestelmän osiota: Suostustenhallinta, Arkistointi, Reseptit, Tahdonilmaisut sekä Lääkemääräyksen luovutuskiellot. Osioita tulee

tulevaisuudessa enemmän, joten testausautomaation käyttäminen olisi tehokasta ja poistaisi mahdollisia inhimillisiä virheitä.

Yhden version testauksen aikana havaittiin, että Mediconsultin testausympäristössä oleva sertifi-kaatti oli vanhentunut (voimassa 5 vuotta). Tämä ei ollut tuotekehitystiimin vastuulla, mutta se myöhästytti julkaisua muutamalla päivällä, koska testaamaan ei päästy ja sitä myötä asiakkaalle annetut lupaukset siirtyivät. Tekniikan tiimille on vastuutettava prosessi, ettei tällainen jatkossa hidasta testausta.

## 6.5 Retrospektiivin tulokset

Sprint retrospektiivejä pidettiin tämän työn aikana 7 kappaletta, joista parannusehdotuksia toimitaan saatiin 15 kappaletta. Kaikkia retrospektiivin parannusehdotuksia ei kuitenkaan viedä eteenpäin käytäntöön eikä niitä seurata. Kehitysehdotuksille kannattaa jatkossa vastuuttaa henkilö, joka vie ja seuraa päätettyä asiaa eteenpäin. Kaikkia kehitysehdotuksia ei kuitenkaan voi viedä sellaisenaan eteenpäin, vaan vaativat tarkempaa analysointia.

## 6.6 Prosessiin kuluva aika

Työssä tutkittiin myös prosessiin kuluva kokonaisaika ja eri tehtävien keskiarvoaikoja. Ne ovat liian pitkät ja palaverit eivät ole enää tehokkaita, eivätkä pysy aina asiassa. Tämä johtunee osittain siitä, että tiimin koko on liian iso Scrum-tuotekehitystiimiksi. Tehtävästä toiseen siirtyminen (task switching) aiheuttaa laadun heikkenemistä. Tiimi on päätetty hajottaa joulukuussa kahdeksi eri tiimiksi, jossa on omat Scrum-masterit ja oma kehitysjono, jolloin tiimeissä on vähemmän kesken-eräistä työtä.

## 6.7 Julkaisu ja julkaisukuvauksen automatisointi

Aikaisemmin uusi versio otettiin Jenkinsin Nightly Build -prosessista, joka käynnistyi joka yö. Jos automaattisessa käännös- ja testausprosessissa tuli virheitä, ne korjattiin seuraavana päivänä ja odotettiin taas seuraavan yön käännöstä. Uuden version tekeminen edellytti myös, että päivitettiin versio numero manuaalisesti ennen yöllä tapahtuvaa käännöstä. Jos tämä unohdettiin tehdä, pahimmillaan julkaisu myöhästyi yksinkertaisen virheen takia yhdestä kolmeen päivää.

Tätä julkaisukäytäntöä muutettiin niin, että Jenkinsiin tehtiin uusi työprojekti (Job), jolla saadaan muodostettua uusi versio halutuista versionhallinnan tiedoista. Työprojektin käynnistäminen kysyy mistä versionhallinnan tiedoista julkaisu tehdään ja mikä versionumeroksi tulee.

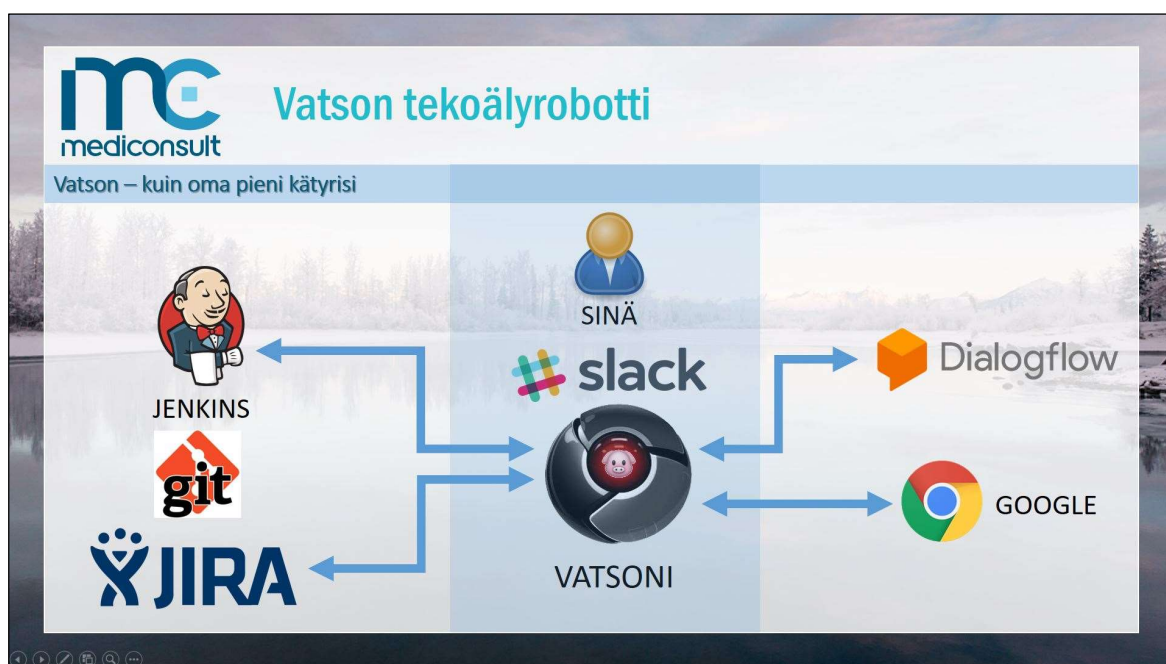
Julkaisukuvaus kannattaa automatisoida mahdollisimman pitkälle ohjelmistorobotille ja yhdistää se uudistettiin julkaisukäytäntöön niin, että viimeinen vaihe on missä ohjelmistoroboti käy version läpi.

## 6.8 Tekoälyn ja ohjelmistorobotin hyödyntäminen

Työn aikana havaittiin, että eri järjestelmät eivät siirrä tietoa keskenään, vaan toimivat täysin toisistaan riippumattomasti. Version julkaisukuvauksen tuottaa JIRA-järjestelmä, mutta sillä ei ole varsinaisesta Jenkins-julkaisusta mitään tietoa. JIRA-tehtävien tilojen ylläpito voidaan osittain automatisoida Jenkins-työhön, mutta tätä ei haluta muuttaa.

Työn aikana pohdittiin yrityksen käytössä olevia työkaluja ja tieto julkaisussa olevista ongelmista pitäisi saada tiimin tietoon. Tähän käyttötarkoitukseen työssä tehtiin Slack-botti "Watson" ja yhdistettiin DialogFlow tekoälymoottoriin, joka integroitiin JIRA järjestelmään REST-rajapintojen avulla. Koska ohjelmistokehitys ei ole ehkä tottunut näkemään ohjelmiston laatua mitattavana ilmiönä, vastarinta tehdyistä analyyseistä on tavallaan turhaa, kun vastassa on tekoälyrobotti eikä sille pysty olemaan vihainen tai kiukutella.

Ohjelmistorobotti on väsymätön ja tehokas laadun tarkastaja. Ohjelmistorobotti ei kuitenkaan ole täysin luotettava, vaan suuntaa antava johtuen ohjelmistojen laadun ja tehtävien hankalasta määrittämisestä ja robotti päättelee aina asiat tiukkojen sääntöjen mukaan.



Kuva 14. Tekoälyrobotin integraatiosovellukset

Robotin pitää olla ajasta ja paikasta irrallinen, joten työssä päädyttiin valitsemaan yrityksen käytössä olevan Slack-pikaviestintäsovelluksen, joka toimii missä tahansa päätelaitteessa jatkuvasti. Haluttiin, että robotti ymmärtää tuotettua tekstiä, joten se yhdistettiin oppivaan tekoälyyn (DialogFlow). Robotin ohjelmiston vuokaavio on kuvattu yleisellä tasolla liitteessä 2. DialogFlowlle opetetaan komentoja Intenttien avulla ja se palauttaa JSON-tietomallin, siitä mitä tekoäly päätteli kysyttävän (liite 3). Tämä on myös tietoturvallinen ratkaisu, koska yrityksen tietoverkon ulkopuolelle ei siirretä mitään tuotekehitykseen liittyvää tietoa, ainoastaan henkilön kysymät kysymykset, joilla rikolliset eivät tee mitään, vaikka he pääsisivät Slack-tietoihin tai DialogFlow-tekoälyyn käsiksi.

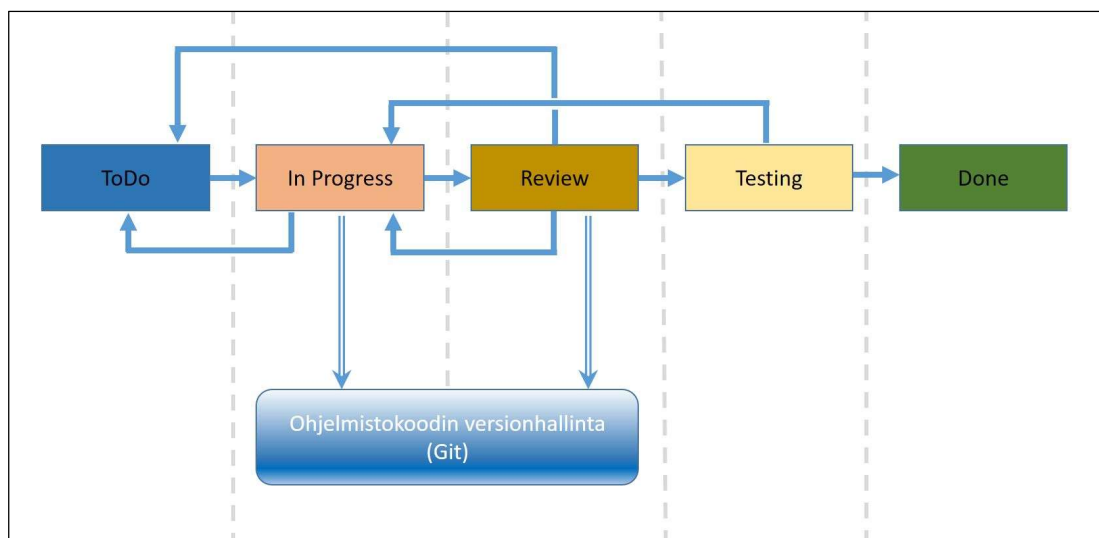


Tekoälylle opetettiin, että "get release 4.3.0 data" tarkoittaa, että käyttäjä haluaa robotin käynnistävän "haeversiotiedot" toiminnon (action). Liitteen 3 esimerkki tietomalli näyttää, että robotilta kysyttiin "Hi watson i want to know release 4.2.1 data" ja se päätteli, että 68% (score) todennäköisyydellä käyttäjä haluaa komennon olevan "haeversiotiedot" parametreilla "4.2" ja "1". Tekoälyyn lisättiin myös muita ei laatuun liittyviä käsitteitä ja älykkyyttä (smalltalk, vitsejä, lounaslistat), jotta se olisi mahdollisimman ihmismäinen (Liite 5).

Tämän työn aikana robotille ohjelmoitiin laadun parantamiseen seuraava komento:

Get release X.Y.Z data (Liite 3).

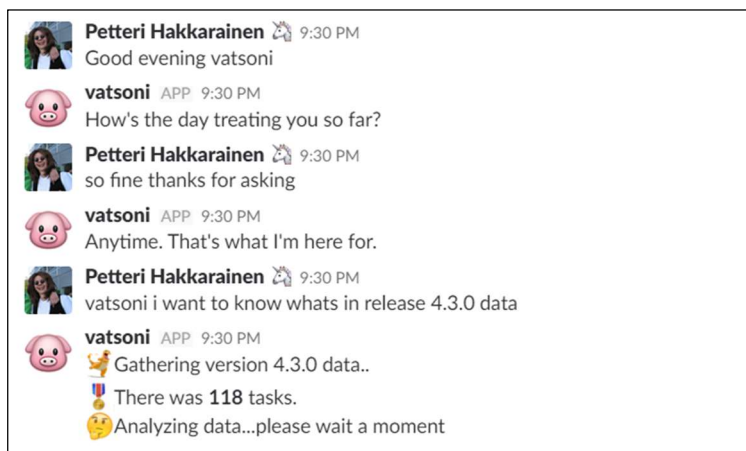
Tämä käynnistää version kokonaisanalysoinnin (Liite 2 - Do Some Stuff). Robotti analysoi version julkaisun kaikki tehtävät ja tarkistaa jokaisesta vaiheesta aloitusajan, montako kertaa tehtävä aloitettiin alusta (In Progress), tehtiinkö tehtävälle koodikatselmointi, annettiinko tehtävälle ratkaisukuvaus ja vietiinkö tehtävästä versionhallintaan ohjelmistokoodia sekä vietiinkö versionhallintaan ohjelmistokoodia ratkaisupäivän jälkeen. Nämä tiedot mahdollistavat saman analyysin, mitä tässä työssä tehtiin käsityönä, huomattavasti laajennettuna ja erittäin nopeasti. Komennon voi käynnistää heti julkaisun tekemisen jälkeen (kappale 6.7) tai se voidaan automatisoida Jenkins-käännösprosessin yhteyteen.



Kuva 15. JIRA tehtävien eri vaiheet ja sallitut siirtymät

Työssä valittiin yhdeksi analysoitavaksi asiaksi robotille, montako kertaa tehtävä aloitettiin alusta. Se kertoo yleensä siitä, että ohjelmiston tekemisessä tai suunnittelussa on tapahtunut virhe tai muu poikkeama, mitä on syytä tarkastella tarkemmin. Ohjelmistokoodia saadaan viedä versionhallintaan vain, jos se ei ole testauksessa tai kokonaan valmis. Jos ohjelmakoodia on viety versionhallintaan testauksessa tai sen jälkeen, kun tehtävä on valmis tilassa sitä ei arvioida muiden kehittäjien toimesta tai testata ollenkaan, jolloin laatua ei voida taata. Pahimmassa tapauksessa viallinen katselmoimaton ja testaamaton koodi menee tuotantoon ja aiheuttaa ongelmia asiakkaan järjestelmässä. Ratkaisukuvausta käytetään, kun luodaan versiosta julkaisukuvaus. Näillä tiedoilla saadaan erittäin

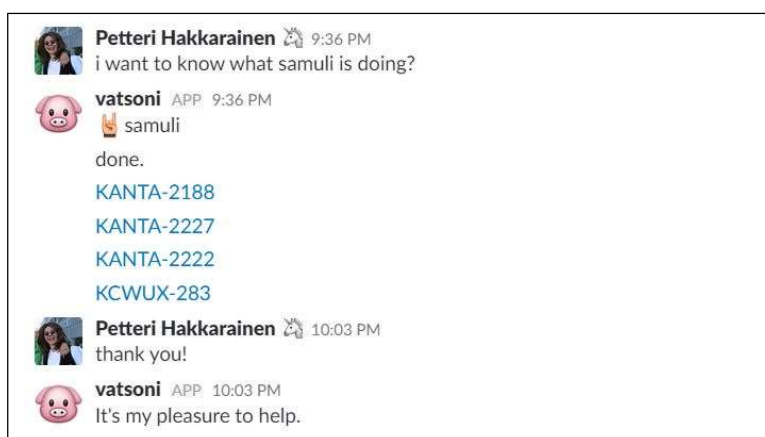
kattava kokonaiskuva version sisällöstä ja siellä olevista mahdollisista laatu-eroista. Robotti antaa analyysistä puolipilku erotteisen luettelon, joka voidaan avata Excel-tilaukkelaskennassa ja jatkaa analysointia manuaalisesti.



Kuva 16. Robottia on pyydetty analysoimaan versio 4.3.0

Liitteessä 4 on nähtävillä robotin palauttamaa analyysia Excel-tilaukkelaskennassa. Rivillä 5 robotti on analysoinut, että tehtävässä ei ole suoritettu koodikatselmointia, testausta eikä annettu ratkaisukuvausta. Tämä ei kuitenkaan ole laatuero, koska tuossa tehtävässä ei tarvittu koodausta, testausta tai ratkaisukuvausta, tätä robotti ei pysty analysoimaan tekstimuotoisesta tehtäväkuvauksesta. Robotti tuo kuitenkin nopeasti näkyville mahdolliset ongelmakohdat ja ne voidaan tarkistaa ihmisvoimin. Uskon, että robotilla on merkittävästi laatu parantava vaikutus tulevaisuudessa. DialogFlow tekoälymoottori ei tue vielä suomen kieltä, mutta se tulee tukemaan sitä jossain vaiheessa.

Toinen ohjelmoitu komento oli, että mitä tietty henkilö tekee? Robotti käy viimeisen viikon projektien tehtävät läpi ja listaa ne linkkeinä suoraan Slack-pikaviestintävälineeseen. Tämä nopeuttaa esimerkiksi omaan työlistaan käsiksi pääsemiseksi.



Kuva 17. Robotilta on kysytty mitä Samuli tekee?

## 7 YHTEENVETO JA JATKO

Laadun kehittäminen on pitkäjänteistä jatkuvan parannuksen prosessia. Isoja muutoksia ei voi tehdä kerralla, koska muutoksen positiivisesta vaikutuksesta laatuun ei voi olla varmuutta. Henkilöstön vaikutus laatuun ja laatuajatteluun on isossa roolissa. Henkilöstö on sitoutettava noudattamaan toimintajärjestelmää ja tuotantoprosessia. Muutoksen pitää olla myös mielekästä, jotta henkilöt sitoutuvat siihen. Tämä voi olla jossain tapauksessa haasteellista, jos henkilö joutuu toimimaan omaa luontoaan vastaan. Esimerkiksi, jos henkilö pitää dokumentaation katselmointia turhana ja toimintajärjestelmä vaatii, että kaikki dokumentaatio on katselmoitava.

Osaan laatua käsittävästä kirjallisuudesta en voi millään tavoin yhtyä. Jos kilpailijan laatu on huonoa, niin kirjallisuuden mukaan meidänkin pitäisi tehdä huonoa laatua tai samaa laatua kuin kilpailija, muuten se olisi resurssien tuhlaamista. Jos kaikki ajattelisivat näin, meillä olisi vain huonoja tuotteita. Tällainen ajattelu suosii huonon laadun tekemistä ja se ei ole oma tai Mediconsultin arvo ja tavoite. Täytyy muistaa, ettei nykymarkkinoilla pärjää, jos kustannukset karkaavat, joten täytyy osata innovoida uusia näkökulmia ja ratkaisuja, sellainen syntyi tässä työssä. Tuotekehityksen pitää tehdä niin hyviä ja laadukkaita tuotteita, että niistä olisi itse valmis maksamaan. Terveystuotteen ohjelmistokehitys tuo omat vaatimuksensa laatuun: kyse on kuitenkin pahimmassa tapauksessa ihmishengestä.

Kirjallisuudessa esitettiin myös, että suorituskyvystä puhuttaessa lyhennetään tehtävään käytettävää aikaa. Tuotekehityksessä on paljon kokouksia ja niiden lyhentäminen lyhentää tehtävään kuluva prosessiaikaa. Tämä ei kuitenkaan vaikuta tuotekehityksen suorituskykyyn tai se voi haitata siihen negatiivisesti. Ohjelmistotuotannossa informaatio on keskeistä ja kokouksissa jaetaan informaatiota. Jos viestintä ei toimi ja tiimillä on eri käsitys tai puutteellinen käsitys asioista, siitä seuraa tehottomuutta, vaikka aikaa olisi yhdessä kohdassa säästetty. Scrum-tuotekehitysmallia voidaan hienosäätää ja kokeilla erilaisia variaatioita. Erityisesti suunnitteluun käytettävää aikaa pitää pohtia enemmän. Työssä tehtyä valmistuksen arvovirtakuvausta voidaan hyödyntää uusien kehittäjien perehdytyksessä.

Laadun tarkkailu ohjelmistotuotannossa julkaisujen avulla on paljon aikaa vievää työtä ilman ohjelmistorobotia. Mittareita saadaan myös esille paljon enemmän ja nopeammin ohjelmistorobotilla. Työssä syntyi kahdeksan konkreettista toimenpidettä laadun parantamiseen. Opinnäytetyön tuloksia voidaan hyödyntää sekä uusien tuotteiden ja ominaisuuksien ennustettavuudessa. Ohjelmistorobotia ollaan vasta ottamassa käyttöön, joten sen avulla oletetaan saavan lisää laadun mittareita esille ja käytettäväksi.

Laadun tarkkailua opinnäytetyössä käytetyillä mittareilla tullaan jatkamaan ja se otetaan osaksi julkaisujen jatkuvan parantamisen prosessia. Opinnäytetyön sivutuotoksena syntyi Watsoni-tekoälyn

ohjelmistorobotti. Robotti on aiheuttanut hieman vastarintaa, kun sitä on testattu tiimissä. Todennäköisesti ei vielä ymmärretä mitä kaikkea robotti pystyy tai ei pysty tekemään, vaan pelätään, että tarkkaillaan työn tekemistä tehokkuuden eikä laadun kannalta.

Tarkoitus on jatkojalostaa ohjelmistorobottiin toimintoja myös eri tiimien käyttöön. Robottia pystytään laajentamaan ja hyödyntämään esimerkiksi hälytysten vastaanottamisessa, tarvittaessa päivittäisessä daily-kokouksessa, aikataulun muistutuksessa tai muissa muistutteissa tai jatkuvasti toistuvissa työtehtävissä. Jatkojalostuksessa otetaan huomioon myös Jenkins- ja Git-integraatiot paremmin, nyt julkaisun laatutieto perustuu JIRA-toiminnanohjausjärjestelmään. Ohjelmistorobotin ohjelmistokoodi ja DialogFlow -tunnukset laitetaan kaikkien Mediconsultin kehittäjien saataville ja innoitavaksi. Uskon että Watsoni tulee olemaan yrityksen kilpailuetu laadun tarkkailussa.

Opinnäytetyö sai ajattelemaan kehän ulkopuolelta ja opetti ettei pidä tyytyä keskinkertaiseen laatuun, vaikka se veisikin paljon aikaa. Tulevaisuudessa ohjelmistorobotteja ja tekoälyä pystytään käyttämään mitä moninaisimpiin tehtäviin: yksi niistä on ohjelmistotuotteiden laaduntarkkailu.

## LÄHTEET JA TUOTETUT AINEISTOT

AGILE MANIFESTO, (2001): Manifesto for Agile Software Development. <http://agilemanifesto.org> (19.8.2017)

CHACON, Scott ja STRAUB, Ben 2017. Pro Git book. [Verkkoaineisto, Viitattu: 28.10.2017] Saatavissa: <https://git-scm.com>

EDUIX, 2017 Mikä jira on? [Verkkoaineisto, Viitattu 30.11.2017] Saatavissa: <http://www.eduix.fi/9-tuotteet-ja-palvelut/9-jira>

GREEN, M. DAVID 2016, Scrum: Novice to ninja. SitePoint

HAIKALA, Ilkka ja MÄRIJÄRVI Jukka 2004. Ohjelmistotuotanto. Helsinki: Talentum.

INSPECTA OY. Terveystuotteen laitteiden ja tarvikkeiden johtamisjärjestelmän sertifiointi. [Verkkoaineisto] Inspecta. [Viitattu: 26. 10 2017.] Saatavissa: <http://www.inspecta.com/fi/Palvelut/Sertifiointi/Jarjestelmasertifiointi/Terveystuotteen-laitteiden-ja-tarvikkeiden-Medical-Devices-johtamisjarjestelman-sertifiointi-ISO-13485/>

KANTA, 2017. [Verkkoaineisto, Viitattu: 28.10.2017] Saatavissa <http://www.kanta.fi>

LAATUKESKUS, 2010. EFQM Excellence Model -Erinomaiset organisaatiot kehittävät toimintaansa ja yllyttävät sitä tasolla, joka täyttää tai ylittää kaikkien sidosryhmien odotukset.

LAITINEN, Miku. 2011. Jatkuvan integration käyttöönotto. Metropolia Ammattikorkeakoulu

LECKLIN, Olli 2002. Laatu yrityksen menestystekijänä. Jyväskylä: Gummerus Kirjapaino Oy.

LILLRANK, Paul 1998. Laatuajattelu. Keuruu: Otavan kirjapaino.

LINUX 2017. [Verkkoaineisto, Viitattu: 28.10.2017] Saatavissa: <https://www.linux.fi/wiki/Git>

MEDICONSULT 2017. Laatukäsikirja. Helsinki.

MUSTANIEMI, Joni. 2013. Jatkuva integrointi Java Enterprise -kehityksen apuna. Tampereen ammattikorkeakoulu. Tampere.

PESONEN, Herkko 2007. Laatu! WS Bookwell Oy, Juva

PÖYHÖNEN, Ilpo 2006. Lääkintälaitteiden ohjelmistot Suunnittelun kehityskohteita vesiputous- ja XP-mallin näkökulmasta. Espoo: Otamedia Oy.

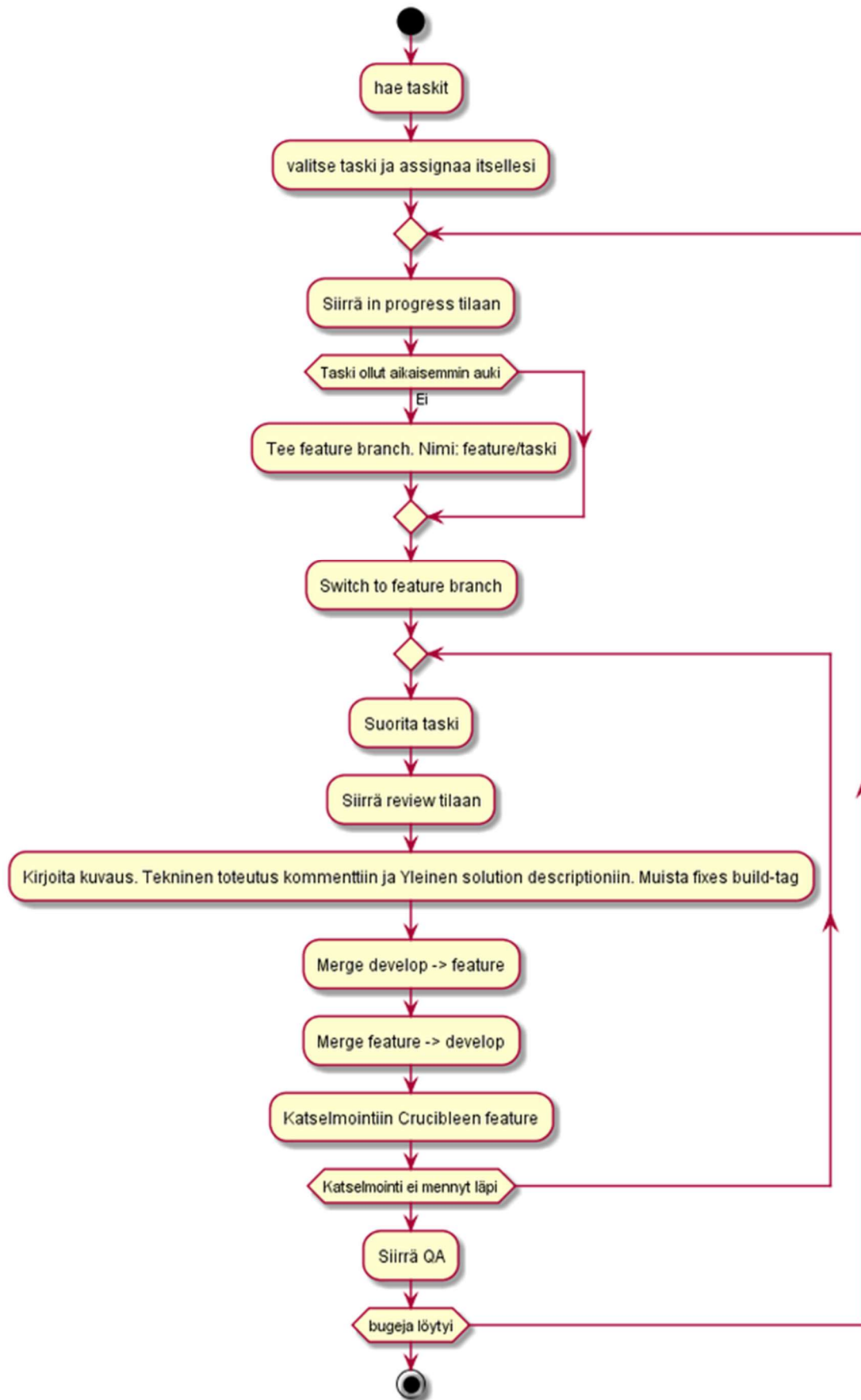
PÖYHÖNEN, Ilpo ja KYLMÄLÄ, Kaarle 2010. EU:n lääkintälaitteiden direktiivin mukaisten terveydenhuollon tuotteiden suunnittelu ja valmistus viranomaisvaatimukset huomioiden. Tampere: Tammerprint Oy.

SCHWABER, Ken ja SUTHERLAND Jeff 2011. The Scrum guide. Scrumin määritelmä ja pelisäännöt. Scrum.org.

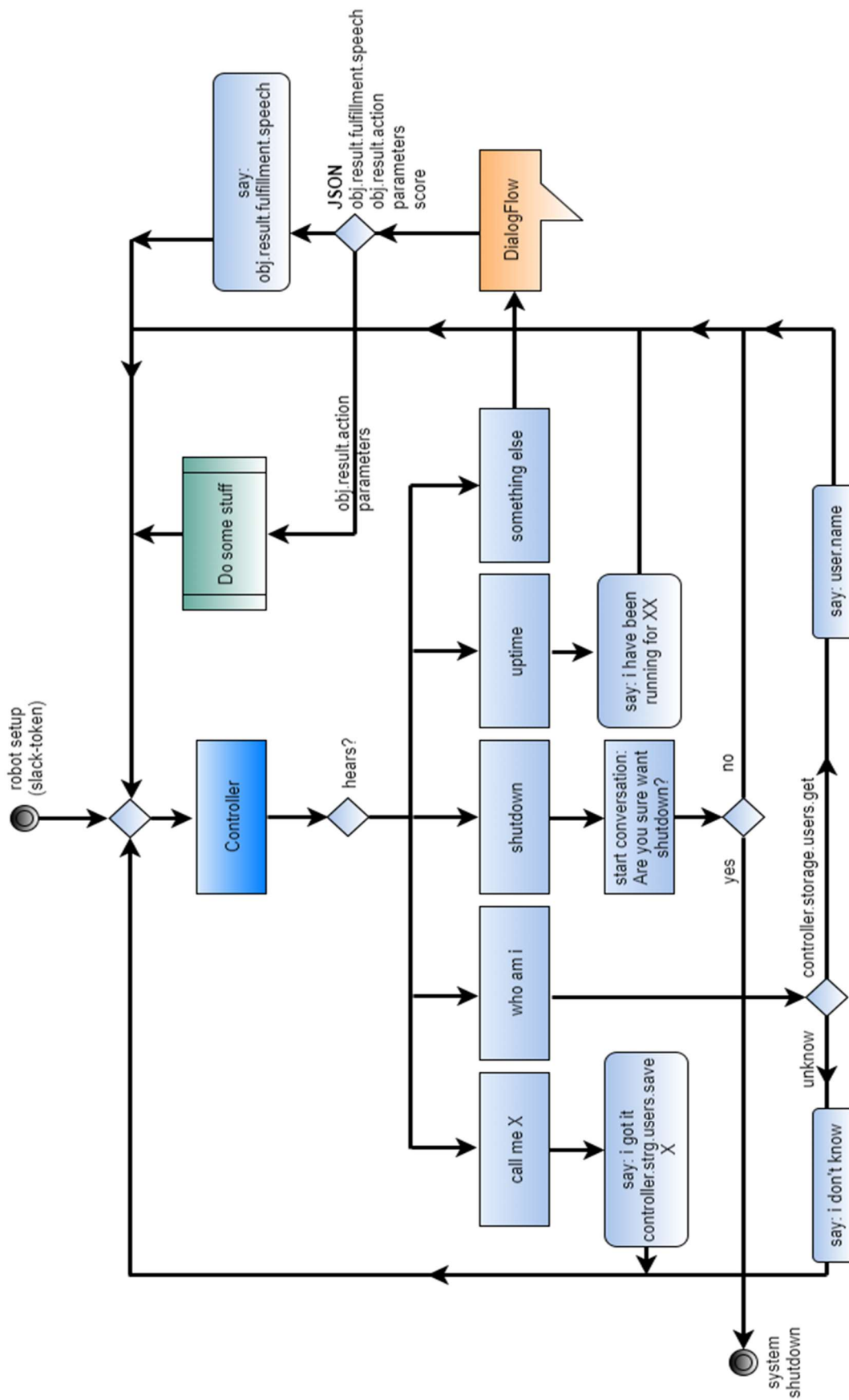
TORKKOLA, Sari 2015. Lean asiantuntijatyön johtamisessa.

WAKARU 2015. ITIL kertaus. 2015. ITIL kertaus -kurssimateriaali.

## LIITE 1: PROSESSIKUVAUS VALMISTUKSESTA



## LIITE 2: OHJELMISTOROBOTIN VUOKAAVIO







## LIITE 4: ROBOTIN PALAUTTAMAA TIETOA JULKAISUSTA

AutoSave On versiodata.csv - Excel Petteri Hakkarainen

File Home Insert Page Layout Formulas Data Review View Tell me what you want to do

Get From Text/CSV From Web From Table/Range

Data Recent Sources Existing Connections Refresh All

Queries & Connections Properties Edit Links

Sort & Filter Filter Advanced Reapply

Text to Columns Data Validation Remove Duplicates Flash Fill

Data Tools Manage Data Model What-If Analysis Forecast Group Ungroup Subtotal Outline

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
05															
1	url	key	created	todo	inprogress	review	qa	resolved	closed	inprogcou	solde	hascodi	hasrevit		
2	https://jir.kanta-2187	14.9.2017 13:25	14.9.2017 13:33	25.9.2017 12:27	26.9.2017 13:22	26.9.2017 13:22	26.9.2017 13:22	26.9.2017 13:22	26.9.2017 13:22	0	1	1	1	1	
3	https://jir.kanta-2099	10.5.2017 15:17	10.5.2017 15:21	10.5.2017 15:21	23.5.2017 14:51	23.5.2017 14:51	23.5.2017 14:51	23.5.2017 14:51	23.5.2017 14:51	0	0	1	1	2	
4	https://jir.kanta-2106	19.5.2017 15:43	30.6.2017 7:57	30.6.2017 7:57	6.7.2017 9:19	6.7.2017 9:19	6.7.2017 9:19	6.7.2017 9:19	6.7.2017 9:19	0	1	1	1	1	
5	https://jir.kanta-1753	4.10.2016 14:02	11.1.2017 11:59	11.1.2017 11:59	11.1.2017 11:59	11.1.2017 11:59	11.1.2017 11:59	11.1.2017 11:59	11.1.2017 11:59	0	0	0	0	0	
6	https://jir.kanta-1650	5.8.2016 12:55	14.9.2016 8:39	14.9.2016 8:39	14.9.2016 15:02	14.9.2016 15:02	14.9.2016 8:39	14.9.2016 11:57	14.9.2016 11:57	0	1	1	1	1	
7	https://jir.kanta-2179	7.9.2017 13:48	19.9.2017 9:31	19.9.2017 9:31	20.9.2017 11:47	20.9.2017 11:47	20.9.2017 9:31	9.11.2017 9:16	9.11.2017 9:16	0	1	1	1	1	
8	https://jir.kanta-1373	12.11.2015 15:47	24.1.2017 9:05	24.1.2017 9:05	24.1.2017 9:53	24.1.2017 9:53	24.1.2017 9:53	24.1.2017 9:53	24.1.2017 9:53	0	1	1	1	1	
9	https://jir.kanta-2014	1.3.2017 15:21	19.5.2017 9:05	19.5.2017 9:05	29.5.2017 14:51	29.5.2017 14:51	29.5.2017 9:05	4.7.2017 11:27	4.7.2017 11:27	0	1	1	1	1	
10	https://jir.kanta-1912	24.1.2017 8:50	24.1.2017 15:57	24.1.2017 15:57	24.1.2017 8:59	24.1.2017 15:57	24.1.2017 15:57	25.1.2017 14:51	25.1.2017 14:51	0	1	1	1	1	
11	https://jir.kanta-1892	4.1.2017 11:06	18.1.2017 16:12	18.1.2017 16:12	16.1.2017 13:44	18.1.2017 16:12	18.1.2017 16:12	19.1.2017 13:04	19.1.2017 13:04	1	0	1	1	1	
12	https://jir.kanta-1720	21.9.2016 9:00	21.9.2016 9:20	21.9.2016 9:20	21.9.2016 9:00	21.9.2016 9:20	21.9.2016 9:20	28.9.2016 13:24	28.9.2016 13:24	0	0	1	1	1	
13	https://jir.kanta-2095	9.5.2017 10:10	10.5.2017 11:46	10.5.2017 11:46	9.5.2017 10:59	10.5.2017 11:46	10.5.2017 11:46	24.5.2017 9:30	24.5.2017 9:30	0	0	1	1	1	
14	https://jir.kanta-2165	22.8.2017 9:35	27.8.2017 14:37	27.8.2017 14:37	27.8.2017 14:37	27.8.2017 14:37	27.8.2017 14:37	3.10.2017 10:17	3.10.2017 10:17	0	1	1	1	1	

## LIITE 5: ESIMERKKI LUONNOLLISESTA KESKUSTELUSTA OHJELMISTOROBOTIN KANSSA

☆ **vatsoni**

Messages About

Today

🔍 nakit

ⓘ ⚙️ @ ☆ ⋮

😊 It's my pleasure to help.

**Petteri Hakkarainen** 10:08 PM  
what you can do?

😊 **vatsoni** APP 10:08 PM  
Did that not make sense? Maybe I misunderstood what you said.

**Petteri Hakkarainen** 10:08 PM  
get me a coffee

😊 **vatsoni** APP 10:08 PM  
Till next time!

**Petteri Hakkarainen** 10:09 PM  
or tell me jokes

😊 **vatsoni** APP 10:09 PM  
Did you know that dolphins are so smart that within a few weeks of captivity, they can train people to stand on the very edge of the pool and throw them fish?

**Petteri Hakkarainen** 10:09 PM  
stop

😊 **vatsoni** APP 10:09 PM  
Cancelled. Waiting for more commands.

**Petteri Hakkarainen** 10:09 PM  
i am hungry get lunch in Kuopio

😊 **vatsoni** APP 10:09 PM  
🍴🍷🍷 Please wait a moment, getting lunch..  
just a second..

😊 **vatsoni** APP 10:09 PM  
added this Plain Text snippet: [Lounaslista Kuopio](#)

- 1 Hurma Kuopio
- 2 Ginger chicken
- 3 Spicy eggplant
- 4
- 5 Viikinkiravintola Harald
- 6 Kermainen maa-artisokkakeitto
- 7 Metsästäjän pihvi oluttipuljin kera

+ Message @vatsoni

@ 😊