TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TOMMI KAIPAINEN

KETTERÄ OHJELMISTOKEHITYS LÄÄKETIETEELLISESSÄ VIITEKEHYKSESSÄ

Diplomityö

Tarkastaja: Professori Kari Systä
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-neuvoston kokouksessa 9. marras-kuuta 2016

TIIVISTELMÄ

TOMMI KAIPAINEN: Ketterä ohjelmistokehitys lääketieteellisessä viitekehyksessä
Tampereen teknillinen yliopisto
Diplomityö, 44 sivua, 0 liitesivua
Tammikuu 2017
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Ohjelmistotuotanto
Tarkastaja: Professori Kari Systä

Avainsanat: ketterä ohjelmistokehitys, lääketieteellinen ohjelmistokehitys, IEC 62304

Lääketieteellisiin laitteisiin liittyy yhä enemmän ja enemmän ohjelmistoja sekä ohjelmakoodia. Potilasturvallisuuden vaarantavia seikkoja on alettu yhdistämään ohjelmakoodista peräisin oleviin virheisiin. Tämän vuoksi lääketieteellinen ala on vahvasti säädelty ja standardoitu. Standardit eroavat markkina-alueittain. Standardit ovat kirjoitettu vaiheesta toiseen etenevien, vesiputousmallin, ohjelmistokehitysmallien näkökulmasta. Ketteryys on kuitenkin yleistynyt ohjelmistokehityksessä jo joitain kymmeniä vuosia, mutta se ei ole vallitsevasti käytössä säädellyillä aloilla. Työssä perehdytään nykyisin käytössä olevaan ketterään lääketieteelliseen ohjelmistokehitysprosessiin, joka pohjautuu scrummalliin. Mallia päivitetään ja eritellään lääketieteellisen ohjelmistokehityksen tärkeimpiä yksittäisiä toimintoja pohjaten kirjallisuuteen.

Kirjallisuudesta on tehty löydöksiä, jotka puhuvat ketterien ohjelmistokehitysmallien puolesta myös säädellyillä aloilla. Ketterien menetelmien avulla on saatu liiketoimintahyötyjä kuten kustannussäästöjä. Liiketoiminta liittyy oleellisena osana ketterään ohjelmistokehitykseen, joten prosessimallia sekä lääketieteellistä alaa pohditaan myös liiketoiminnan näkökulmasta.

Tuloksena muodostettiin käsitys prosessimallista, jota lääketieteellinen ohjelmistokehitys vaatii Euroopan ja USA:n markkinoilla. Sovellettavat standardit löydettiin kirjallisuuteen pohjautuen. Tärkeimmät yksittäiset toiminnot eriteltiin ja kirjallisuuteen pohjautuen tehtiin liitoksia prosessimallin ja eri standardien vaatimien toimintojen, kuten riskienhallinta, välille. Yhtenä työn tuloksena muodostuu kokonaiskuvaa lääketieteellisen ohjelmistokehityksen eroista verrattuna säätelemättömään ohjelmistokehitykseen.

ABSTRACT

TOMMI KAIPAINEN: Agile software development in medical field
Tampere University of Technology
Master of Science Thesis, 44 pages, 0 Appendix pages
January 2017
Master's Degree Programme in Information Technology
Major: Software Engineering
Examiner: Professor Kari Systä

Keywords: agile software development, medical software development, IEC 62304

Medical devices include more and more software nowadays. The factors leading into patient endangering situations have been connected to errors that occur in software code. Therefore, the medical domain is strictly regulated and standardization for the domain has been formed. The standards related to medical devices differ in separate markets. The standards have been written in a way that they describe a software development process that proceeds from a phase to another phase like the waterfall model. However, agile software development has become more and more common in the recent decades but is not used dominantly in regulated software development. In this thesis a currently used agile software development model for medical software development is inspected. The model is based on scrum. The model is also updated and some most critical sub-processes are inspected that are based on literature.

In literature there have been findings that support agile software development methods also in regulated fields. With agile methods significant business benefits like lowered development costs have been achieved. Business is a very important aspect of agile software development so therefore the process model and the medical domain are also inspected in terms of their business value and effects on business.

As a result an overview of agile medical software development model was formed. Because the markets in USA and EU differ, the differences in the required standards in those markets was also inspected based on literature. Also based on literature, the most important sub-processes were presented and connections made between the sub-processes required by the standards and the presented development process. One example of an important sub-process is risk management. As another result, an overview was made to express the differences between regular software development and regulated software development such as medical software development.

ALKUSANAT

Ajatus työn aiheesta syntyi työpaikallani lääketieteellisiin projekteihin liittyen. Tavoitteenani työtä tehdessä oli syventää lääketieteellisiin ohjelmistoihin liittyvää osaamista sekä tarkastaa projekteissa käytettyjä prosesseja säädösten näkökulmasta. Molemmat tavoitteet tulivat täytetyiksi.

Erityiset kiitokset haluan esittää diplomityön ohjaajalleni Kari Syställe työn ohjaamisesta oikeaan suuntaan, jotta selkeä fokus työlle löytyi. Kiitos myös työpaikalleni mahdollisuudesta tutustua lääketieteelliseen alaan, minkä kautta syntyi idea työn aiheeksi.

Tampereella, 17.1.2017

Tommi Kaipainen

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

FDA	Food and Drug Administration, Yhdysvaltojen lääkinnällisten laitteiden myyntiä säätelevä laitos
FMEA	Failure mode effects analysis, riskienhallinnan työkalu riskien ja niiden vaikutusten kartoittamiseen
FTA	Fault tree analysis, riskienhallinnan työkalu riskiskenaarioiden kartoittamiseen
MDD	Medical device directive, määrittää Euroopan Unionin vaatimukset lääketieteellisille laitteille
MVP	Minimum viable product eli tuote jossa on juuri riittävästi ominaisuuksia, jotta saadaan lisää tietoa tuotteesta ja siitä, kannattaako sitä jatkokehittää

1. JOHDANTO

Lääketieteelliseen käyttöön tarkoitetuista laitteista löytyy tätä nykyä yhä enemmän ohjelmakoodia. Viimeisten vuosikymmenien aikana on alettu kiinnittää huomiota ohjelmakoodista peräisin oleviin virheisiin, jotka vaarantavat potilasturvallisuuteen. Näin ollen alasta on tullut yhä säädellympi ja on kehitetty standardeja, jotka asettavat vaatimuksia ohjelmistotuotteiden kehitykselle. Standardeilla on tarkoitus parantaa tuotteiden laatua ja potilasturvallisuutta. Standardit on kirjoitettu vesiputousmallin ohjelmistokehitysmenetelmien näkökulmasta. Nykyisin kuitenkin ohjelmia kehitetään ketterästi usealla eri alalla.

Työssä käsitellään lääketieteellistä ketterää ohjelmistokehitystä ja muun muassa ketteryyden tuomia liiketoimintahyötyjä vaiheesta toiseen eteneviin malleihin verrattuna. Käsittelyssä on ylläpitovaiheen prosessimalli, jossa ylläpitovaihe tarkoittaa sitä, että tuote on jo markkinoilla ja sitä jatkokehitetään. Prosessimallia tarkastellaan ja päivitetään kirjallisuudesta löytyneen materiaalin näkökulmasta. Lisäksi työssä luodaan katsaus lääketieteellisessä ohjelmistokehityksessä käytettäviin standardeihin ja siihen, onko käytettävillä standardeilla eroa markkina-alueittain. Tarkastelun kohteena ovat Eurooppa ja USA.

Luvussa 2 käsitellään työn keskeisin tausta. Luvussa käsitellään ketteryyden perusajatukset verrattuna vesiputousmallin ohjelmistokehitykseen. Scrum-ohjelmistokehitysmalli esitellään pääpiirteittäin. Ohjelmistokehitysmallien lähestymistapojen eroja havainnollistetaan myös liiketoiminnan näkökulmasta. Luvussa 2 käydään läpi myös keskeisimmät standardit, jotka liittyvät lääketieteelliseen ohjelmistokehitykseen. Luvussa 3 esitellään tutkimuskysymys sekä siihen liittyvää taustaa. Samassa luvussa luodaan katsaus olemassa olevaan tutkimukseen aiheesta. Luvussa 4 eritellään prosessimallin tärkeimmät erityispiirteet lääketieteellisen ohjelmistokehityksen näkökulmasta. Näiden pohjalta esitellään nykyinen ylläpitovaiheen prosessimalli. Tämän jälkeen kirjallisuuteen pohjautuen päivitetään prosessimallia. Lopuksi luodaan yhteenveto tärkeimmistä lääketieteelliseen ohjelmistokehitykseen liittyvistä piirteistä ja näiden liiketoimintaan vaikuttavista seikoista sekä arvioidaan ratkaisua.

2. KETTERÄ OHJELMISTOKEHITYS LÄÄKETIE-TEELLISESSÄ VIITEKEHYKSESSÄ

Lääketieteellinen ohjelmistokehitys on tarkoin säädeltyä. Tämä juontaa juurensa lisääntyneeseen koodin määrään lääkinnällisissä laitteissa sekä virhetilanteisiin, jotka on jäljitetty koodiin. Potilasturvallisuuden vuoksi ala on valvottua ja ohjelmistokehitys säädeltyä. Osoituksena tästä lääketieteellisessä ohjelmistokehityksessä täytyy noudattaa useita standardeja. Säädellyillä aloilla on perinteisesti kehitetty ohjelmia vesiputousmallin mukaan. Vesiputousmallissa ohjelmistokehitys etenee vaiheesta toiseen alkaen määrittelystä ja päättyen implementoinnin kautta testaukseen ja toimitukseen. Kuten nykyisin tiedetään, on ohjelmia mahdollista kehittää useilla eri tavoilla eivätkä kaikki mallit ole lineaarisesti vaiheesta toiseen eteneviä. Työssä käsiteltävissä ketterissä menetelmissä lähestymistapa on usein iteratiivinen. Tämä tarkoittaa sitä, että vaiheita tehdään usein päällekkäin ja niitä iteroidaan uudestaan ja uudestaan paremman lopputuloksen saavuttamiseksi.

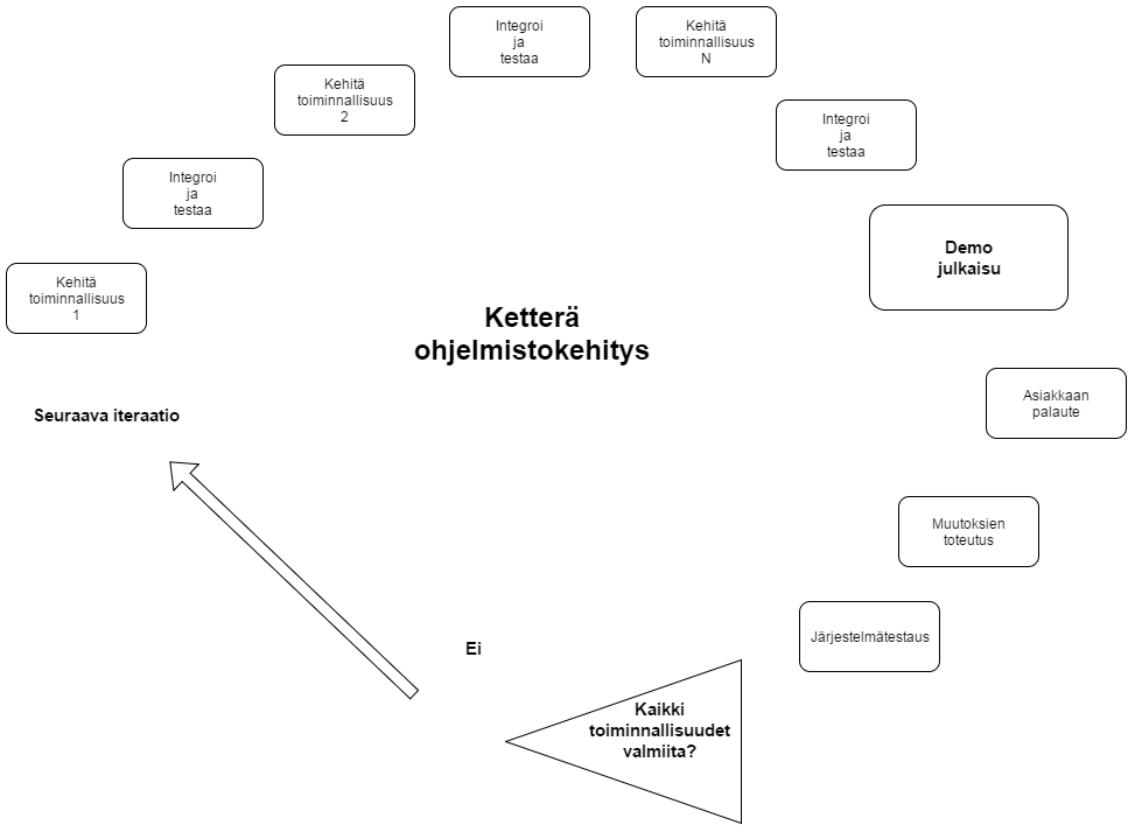
Säätelemättömillä ohjelmistokehityksen aloilla ketteristä menetelmistä on muotoutumassa alan normi [27]. Niiden etuina on muun muassa parempi muutosten hallinta, kustannustehokkuus, pienempi julkaisuaika sekä parantunut ohjelmiston laatu. Kaikki edellä mainitut seikat johtavat suurempaan asiakasarvoon. Nähtävillä olevista hyödyistä huolimatta säädeltyjen alojen ohjelmistoteollisuus on kuitenkin vasta siirtymävaiheessa kohti ketteriä menetelmiä. Esimerkki tällaisesta alasta on lääketieteellinen ohjelmistokehitys.

2.1 Ketterä ohjelmistokehitys lyhyesti

Ketteriä ohjelmistokehitysmenetelmiä on useita. Tarkemmassa tarkastelussa näistä on tässä työssä Scrum. Ketteriä menetelmiä yhdistää niin kutsuttu ketterä manifesti [2]. Ketterä manifesti listaa neljä tärkeintä ohjelmistokehityksen arvoa ketterästä näkökulmasta:

- Yksilöt ja vuorovaikutus prosessien ja työkalujen sijaan
- Toimivat ohjelmisto kattavan dokumentaation sijaan
- Asiakasyhteistyö sopimusneuvottelujen sijaan
- Muutokseen mukautuminen suunnitelman tarkan noudattamisen sijaan

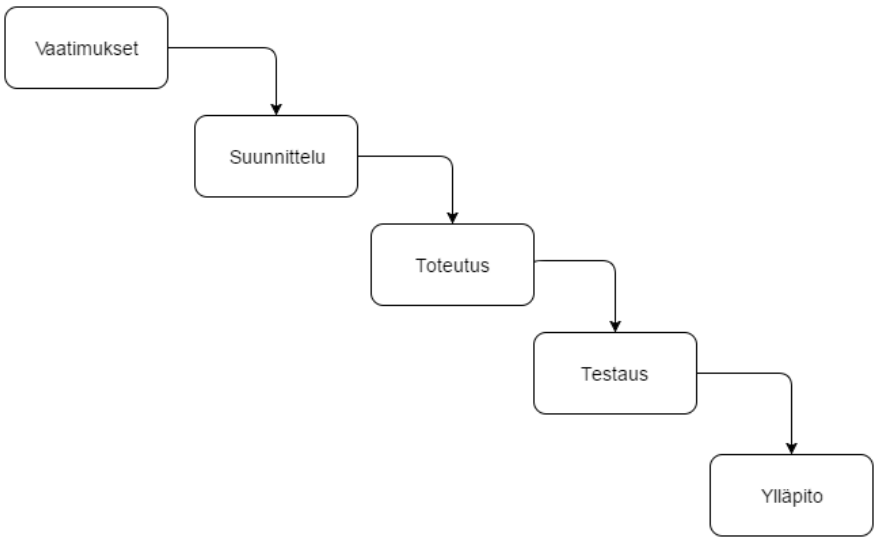
Yhteistä ketterille menetelmille on iteratiivinen, inkrementaalinen ohjelmistokehitys. Kuva 1. havainnollistaa ketterän ohjelmistokehityksen perusperiaatetta [16]. Kuvasta käy ilmi, että iteraatiossa tehdään useita sellaisia asioita, joita vesiputousmallin ohjelmistokehityksessä tehtäisiin kokonaan eri projektin vaiheissa. Yhden iteraation aikana kehitetään ominaisuuksia, integroidaan ja testataan ne, esitellään asiakkaalle saavutetut tulokset, pyydetään palautetta sekä muuta. Tästä edetään seuraavaan iteraatioon kehittäen joko samoja ominaisuuksia yhä tai kehittäen uusia ominaisuuksia.



Kuva 1. Ketterän ohjelmistokehityksen iteratiivinen luonne havainnollistettuna [16].

Ketterät menetelmät korostavat pyrkimystä mukautua muutoksiin pikemmin kuin muutosten estämistä tai niiden ennalta suunnittelua. Jatkuvaan mukautumiseen ja ketteryyteen liittyy olennaisena osana jatkuva kommunikaatio. Kommunikaatiota ylläpidetään ohjelmistokehitystiimin sisällä (mm. päivittäinen tapaaminen, *engl. ”daily scrum”*) sekä asiakkaan suuntaan. Yksi periaatteista onkin nopea ja jatkuva palautteen kerääminen, jonka voidaan katsoa myös sisältyvän muutoksiin mukautumiseen [38]. Ketterät menetelmät tähtäävät parempaan muutosten hallintaan. Tämän voidaan nähdä olevan yksi juurisyistä siihen, miksi ketterät menetelmät ovat kustannustehokkaita.

Perinteisissä vesiputousmallin ohjelmistokehitysprosesseissa työtä tehdään vaiheittain aina edeten vaiheesta seuraavaan. Tätä on havainnollistettu kuvassa 2.

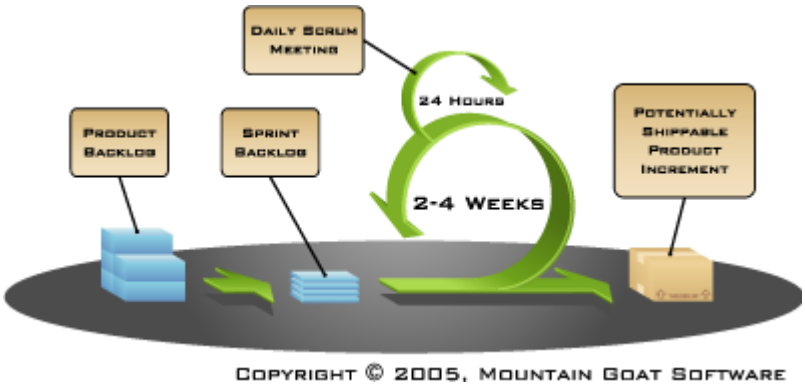


Kuva 2. Vesiputousmallin ohjelmistokehitys.

Työ alkaa määrittelystä, jatkuu suunnittelun kautta toteutukseen ja testaukseen ja päättyy valmiin tuotteen julkaisuun. Ketterissä prosesseissa näin ei ole. Niiden ideologia lähtee siitä, että siihen mennessä, kun valmis tuote on markkinoilla, ovat vaatimukset jo ehtineet muuttua. Sen vuoksi pyritään mahdollisimman aikaisessa vaiheessa valmiiseen ”osatuotteeseen”, jota inkrementaalisesti parannetaan palautteensaannin ja muuttuneiden vaatimusten myötä. Näissä prosesseissa aikaisemmin määriteltyjä työn eri vaiheita tehdään usein päällekkäin: jo määriteltyjä ominaisuuksia toteutetaan, jo toteutettuja ominaisuuksia testataan ja testattuja ominaisuuksia viedään tuotantokäyttöön. Työn päällekkäisen luonteen ja palautteenkeruun vuoksi ketterät prosessit ovat usein iteratiivisia.

2.2 Scrum

Scrum on iteratiivinen ja inkrementaalinen ketterän ohjelmistokehityksen projektinhallinnallinen viitekehys [39]. Scrumin perusajatusta on havainnollistettu kuvassa 3. Scrum rakentuu niin kutsuttujen pyrähdyksien (*sprint*) ympärille. Pyrähdykseen valitaan tietty määrä ominaisuuksia toteutettavaksi tuotteen kehitysjonosta (*product backlog*), joista muodostetaan pyrähdyksen kehitysjono (*sprint backlog*). Jokaisen pyrähdyksen tavoitteena on valmis tuote: pyrähdykseen valitut ominaisuudet tehdään vertikaalisuunnassa alusta loppuun, jotta ne voidaan pyrähdyksen päätteeksi integroida osaksi valmista tuotetta. Pyrähdyksiin liittyy useita tapaamisia, joille on säädetty eri tavoitteita.



Kuva 3. Scrumin rakenne havainnollistettuna [41].

Scrum soveltuu käytettäväksi haastaviin ja monimutkaisiin projekteihin. Tässä vaiheessa on hyvä huomauttaa, että scrum ei itsessään ole vain ohjelmistokehityksen projektiviitekehys, vaan sitä on mahdollista käyttää muillakin aloilla. Scrum on hierarkialtaan varsin matala. Varsinaisella ohjelmistokehitystiimillä on paljon päätöksentekovastuuta päivittäiseen tekemiseen ja toteutustapoihin liittyen.

2.2.1 Scrumin käsitteitä

- **Pyrähdyksellä** (*sprint*) tarkoitetaan scrum-ohjelmistokehityksen kiinteän mittaista osaa, joiden ympärille ohjelmistokehitys rakentuu [39]. Pyrähdyksien pituus voi vaihdella projektista riippuen kahden ja neljän viikon välillä. Pyrähdyksien keskeisenä ideana on valita pyrähdykseen toteutettavaksi tietty joukko ominaisuuksia (*user story*) kehitysjonosta (*product backlog*) ja kerätä palautetta jokaisen pyrähdyksen päätteeksi. Valittua joukkoa kutsutaan pyrähdyksen kehitysjonoksi (*sprint backlog*). Kun palautetta kerätään jatkuvasti, voidaan tuotetta viedä pienemmissä erissä kohti haluttua lopputulosta: jo seuraavassa pyrähdyksessä on mahdollista reagoida edellisen pyrähdyksen synnyttämiin muutostarpeisiin. Muutoksia voidaan haluta äskettäin toteutettuun ominaisuuteen tai vaihtoehtoisesti äskettäin toteutetut muutokset voivat luoda tarpeen kokonaan uudelle lisäominaisuudelle.

- **Käyttäjätarinalla** (*user story*) tarkoitetaan yksittäistä tuotteeseen liittyvää toteutettavaa ominaisuutta. Käyttäjätarina on käyttäjän näkökulmasta esitetty kuvaus toteutettavasta ominaisuudesta, jonka tulisi määrittää kenen näkökulmasta, mitä halutaan ja miksi halutaan ominaisuus toteutettavan. Esimerkki käyttäjätarinasta voisi olla ”Pääkäyttäjänä haluan hallita käyttöoikeuksia, jotta voin rajata pääsyä tuotteen eri osiin käyttäjäryhmien välillä.” Käyttäjätarinat myös pisteytetään työ-

määrän arvioimiseksi. Yksi piste ei mittaa käytettävää aikaa, vaan arvioitua työmäärää suhteessa muihin käyttäjätarinoihin. Käyttäjätarinat jaetaan usein pienempiin osiin eli tehtäviin (*task*). Tämä myös helpottaa tarinoiden arvioimista.

- **Nopeudella** (*engl. Velocity, yleisemmin käytetty)* kuvataan yhden pyrähdyksen aikana tehtyjen pisteiden määrää. Suureen avulla voidaan arvioida, kuinka paljon sisältöä yhdessä pyrähdyksessä tiimi ehtii toteuttamaan. Nopeuden avulla voidaan myös arvioida kokonaisprojektin kestoa, mikäli projektin kokonaispisteet on tiedossa.

- **Tuotteen kehitysjonolla** (*product backlog*) tarkoitetaan listaa ominaisuuksista, jotka valmiiseen tuotteeseen on vielä jossain vaiheessa tarkoitus toteuttaa [10]. Kehitysjono muodostetaan kehityksen alussa ja vastuu sen ylläpitämisestä on Scrum-ideologian mukaisesti tuotteen omistajalla (*product owner*). Kehitysjonon ylläpitäminen tarkoittaa sitä, että asiakasvaatimusten muuttuessa kehitysjonoa täytyy päivittää sen mukaan. Vaatimuksia saattaa poistua, niitä saattaa tulla lisää, niiden sisältö saattaa muuttua tai niiden prioriteetit saattavat muuttua. Kehitysjonon tulisi aina kuvata ajantasaisesti seuraavana kehitykseen tulevien käyttäjätarinoiden tilaa.

2.2.2 Scrum roolit

Matalasta hierarkiasta ja korkeasta itseohjautuvuudesta huolimatta Scrumissa on muutama keskeinen rooli [39].

- **Scrummaster (Scrum master):** Niin kutsuttu scrummaster voidaan rinnastaa osittain perinteiseen projektipäällikköön, mutta näiden välillä on myös suuria eroja. Scrummaster on eräänlainen tiimin ”ohjaaja”, jonka tehtävänä on mahdollistaa tiimin mahdollisimman tehokas työskenteleminen. Scrummasterilla ei ole kuitenkaan perinteistä määräysvaltaa ihmisiin, vaan määräysvaltaa on sen sijaan prosessiin [40].
- **Tuoteomistaja (product owner):** Tuoteomistaja on vastuussa tuotteen kaupallisen arvon kehittämisestä. Käytännössä tuoteomistaja hallinnoi tuotteen kehitysjonoa eli lisää, poistaa ja muokkaa sen käyttäjätarinoita sekä ennen kaikkea pitää huolta niiden keskinäisistä prioriteeteista. Prioriteetti vaikuttaa siihen, milloin käyttäjätarinat siirtyvät työn alle.
- **Scrum tiimi:** Tiimi on itseohjautuva ryhmä, joka vastaa tuotteen kehittämisestä. Tiimi määrittää työskentelytavat ja sitoutuu pyrähdyksen tavoitteeseen. Koska Scrum on todella matala hierarkialtaan, on tiimillä paljon päätäntävaltaa muun muassa tuotteen arkkitehtuurisiin valintoihin.

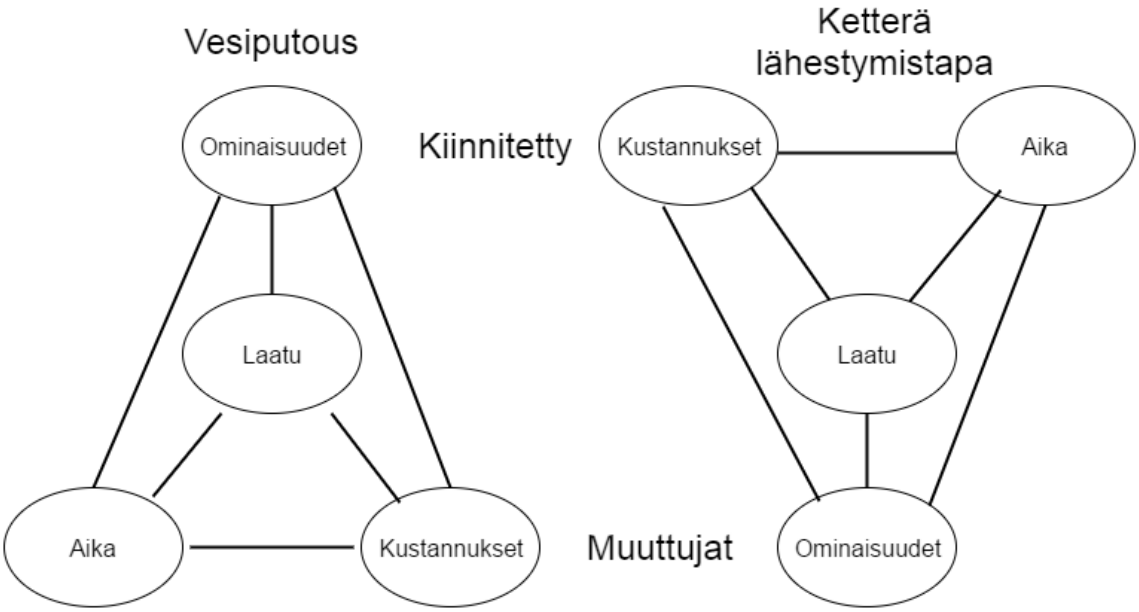
2.2.3 Scrum tapaamiset

Scrumin pyrähdys rakentuu määrättyjen tapaamisten ympärille.

- **Päivittäisessä tapaamisessa** (*engl. daily* scrum) tiimi käsittelee eilen tehtyjä asioita ja seuraavaksi tekoon tulevia tehtäviä sekä nostaa esiin mahdollisia esteitä.
- **Pyrähdyksen suunnittelupalaveri** (*engl. sprint planning meeting*) pidetään ennen pyrähdystä. Tarkoitus on katsoa hieman tulevaan ja suunnitella pyrähdystä sen verran kuin sillä hetkellä on näkyvyyttä. Usein tässä tapaamisessa esimerkiksi pilkotaan käyttäjätarinoita pienemmiksi tehtäviksi.
- **Pyrähdyksen arviointipalaverissa** (*engl. sprint review meeting*) näytetään sprintin tuotokset tuotteen sidosryhmille. Usein tiimi pitää demon viime sprintissä toteutetuista ominaisuuksista. Tilaisuudessa on tarkoitus kerätä palautetta.
- **Pyrähdyksen retrospektiivi** pidetään tiimin kesken pyrähdyksen jälkeen. Tapaamisen tarkoitus on jatkuva parantaminen. Tapaaminen on ajaltaan rajoitettu, kuten muutkin Scrum-tapaamiset. Tapaamisessa tiimi esittää näkemyksiä usein esimerkiksi kysymyksien ”Mikä meni hyvin?”, ”Missä voisimme parantaa?”, ”Mitä haluaisimme muuttaa?” avulla.

2.3 Ketterien menetelmien lähestymistapa liiketoimintaan

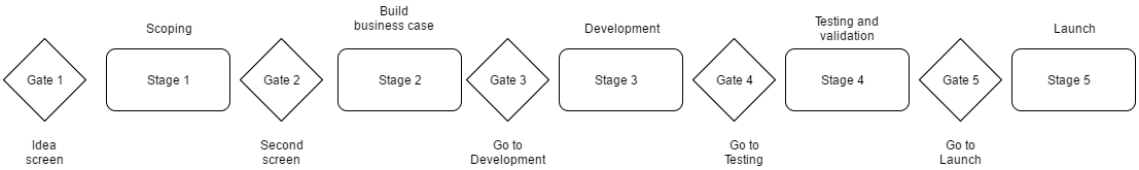
Liiketoiminnan kannalta ajateltuna sekä ketterillä että vesiputousmallin ohjelmistokehitysperiaatteilla on samat tavoitteet: onnistuneet projektit, joissa tehdään määrätty joukko laadukkaita ominaisuuksia sovitussa budjetissa ja sovitussa aikataulussa. Vesiputousmalli ja ketterät prosessit suhtautuvat kuitenkin eri tavalla projektissa esiintyviin muuttujiin kuten kuvasta 4 nähdään.



Kuva 4. Vesiputousmallin ja ketterän lähestymistavan tulkinta projektin ominaisuuksista [25].

Kuvasta käy ilmi, että vesiputousmallissa ja ketterissä menetelmissä projektin muuttujia tarkastellaan eri näkökulmista. Vesiputousmallissa ominaisuudet ovat kiinteitä ja jotta ominaisuudet saadaan tehtyä, niin venytetään tarvittaessa projektin aikaa ja kustannuksia. Tämä on käännetty päinvastoin ketterässä lähestymistavassa. Siinä kustannukset ja aika ovat asetettu kiinteiksi projektille ja tarvittaessa projektin ominaisuusjoukko, eli projektin laajuus (*engl. scope*), joustaa. Laatu on molemmissa lähestymistavoissa keskeinen asia.

Ketterät menetelmän pyrkivät vaikuttamaan myös projektiriskeihin lähestymistavallaan. Perinteisessä vesiputousmallisessa projektissa tehdään valmis tuote ”yhdessä osassa”. Tämä tarkoittaa sitä, että asiakas saa valmiin tuotteen, kun projekti on päättynyt. Jos kesken projektin todetaan, että aikataulu viivästyy tai budjetti ylittyy, voi projektia olla vaikea päättää kesken, sillä tuotos voi jäädä keskeneräiseksi ja valmistumatta ilman lisäpanostuksia. Tämä on tilaajan kannalta epämieluisa tilanne. Ongelmaan on pyritty vaikuttamaan esimerkiksi erilaisilla ”Stage-gate” –malleilla, joissa projektin tuotoksia katsellaan tiettyjen vaiheiden tai virstanpylväiden jälkeen ja tarkastellaan, onko jatkaminen mielekästä. Yksi esimerkki Stage-gate –mallista ja sen ideasta korkealla tasolla on esitetty kuvassa 5.

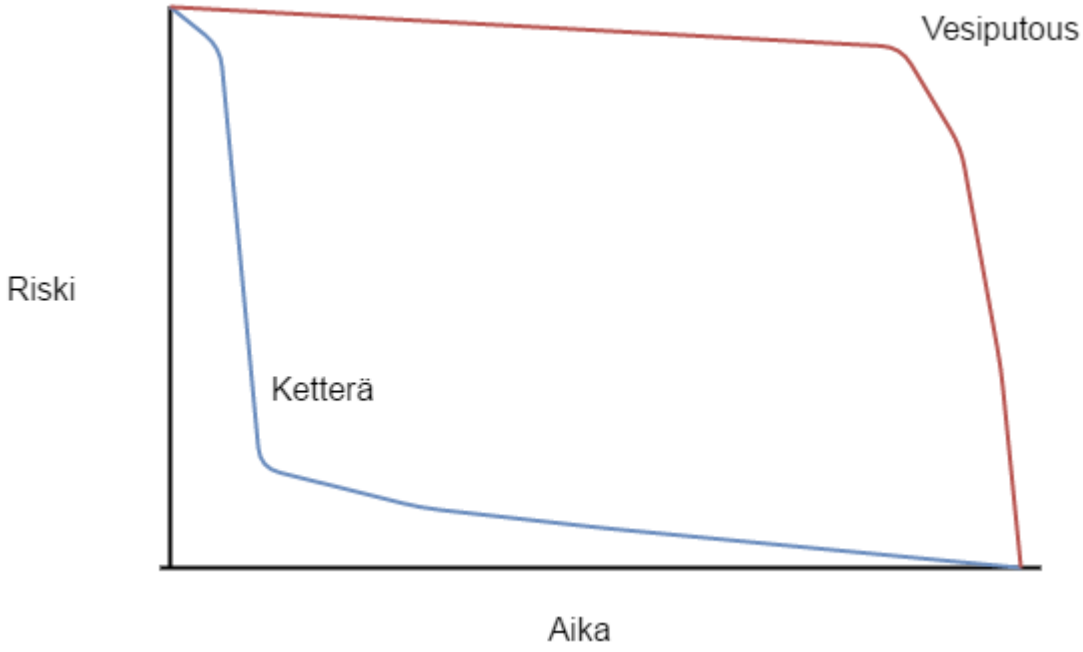


Kuva 5. Stage-gate malli havainnollistettuna.

Kuvassa näkyvät portit ovat niitä kohtia projektissa, jossa tehdään päätös siitä, jatketaanko projektia vai lopetetaanko se [51]. Samoissa kohdissa käydään läpi projektin tilan

keskeisiä kysymyksiä kuten: onko projekti aikataulussa, onko projekti budjetissa. Osaltaan tämä lisää läpinäkyvyyttä projektiin ja tarjoaa mahdollisuuden rajoittaa riskiä ennalta sovituissa pisteissä. Läpinäkyvyys on myös ketteryyden yksi tärkeistä arvoista.

Ketterät menetelmät suhtautuvat ongelmaan eri tavalla. Esimerkiksi Scrum-menetelmässä tehdään joka pyrähdyksen tuotoksena uutta toimivaa toiminnallisuutta. Uudet ominaisuudet kehitetään pystysuunnassa muuhun tuotteeseen sopivaksi. Tämä tarkoittaa sitä, että ominaisuudet määritellään, toteutetaan, testataan ja integroidaan osaksi tuotetta saman pyrähdyksen aikana. Tällä tavoin asiakas pystyy seuraamaan tilaamansa projektin etenemistä pyrähdys kerrallaan ja sopimuksesta riippuen kehitys on mahdollista keskeyttää pyrähdyksen päätteeksi siten, että tuotoksena syntyi kuitenkin valmis tuote. Tämä rajoittaa kokonaisprojektitoimitukseen sisältyvää riskiä. Asiaa on havainnollistettu kuvassa 6.

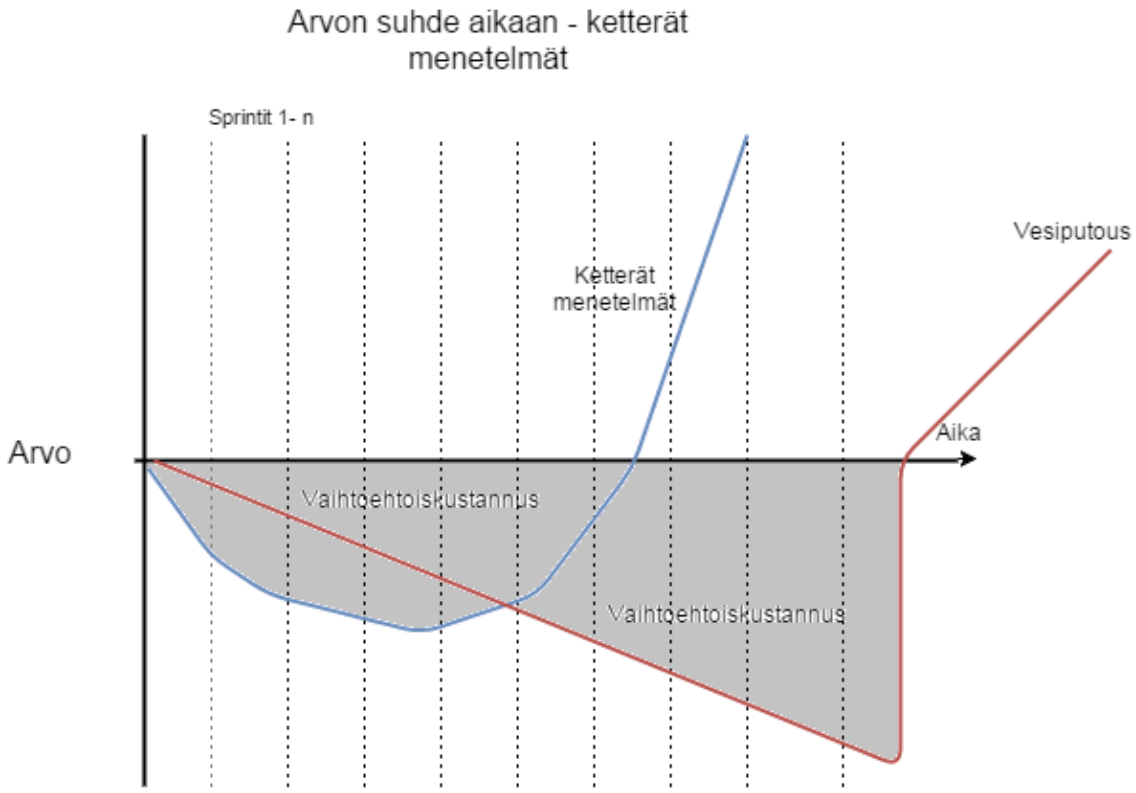


Kuva 6. Ketterän- ja vesiputousmallin riski kuvattuna suhteessa aikaan [7].

Kuvasta 6 nähdään, että vesiputousmallin projektissa riski pysyy melko korkealla tasolla aina projektin päättymiseen asti. Ketterissä projekteissa riski pienenee voimakkaasti heti alun jälkeen ja keskellä projektia riskiä on enää hyvin vähän.

Toinen kaupallinen seikka, jonka ketterät menetelmät lähestymistavallaan ratkaisevat, on mahdollisuus julkaista tuote nopeammin. Todettakoon, että vesiputousmallin pilotti-tyyppisillä projekteilla on mahdollisuus päästä jokseenkin samaan lopputulokseen, mutta ketterät menetelmät tekevät tämän osana normaalia prosessia. Nopeampaa julkaisua ja nopeampaa arvontuottamista havainnollistetaan kuvassa 7. Kuvassa vertaillaan sitä, kuinka kauan ketterillä projekteilla kestää tuottaa arvoa asiakkaalle verrattuna vesiputousmallin projekteihin. Kuvan ”opportunity cost” kuvaa sitä, kuinka paljon arvoa menetetään sillä

välin, kun markkinoille ei ole vielä tarjota tuotetta käytettäväksi. Erona tavoissa on vesiputousmallin kiinteästi määritelty ominaisuusjoukko, joka toimitetaan projektin päätteeksi, kaikkien vaiheiden jälkeen ja puolestaan ketterien projektien pala kerrallaan -tyyppinen toimitus, jossa asiakas saa käyttöönsä määrätyn joukon ominaisuuksia aina määrätyin väliajoin. Asiaan liittyy olennaisena osana sellaiset termit kuin jatkuva integroiminen ja jatkuva toimittaminen. Termit tarkoittavat sitä, että uusia ominaisuuksia integroidaan, eli yhdistetään, jatkuvasti osaksi valmista tuotetta ja valmista tuotetta toimitetaan jatkuvasti, eli esimerkiksi pyrähdyksien päätteeksi, kaikki vaiheet tehtyinä asiakkaalle. Todellisuudessa toimituksia ei tietenkään tehdä tuotantokäyttöön heti ominaisuuksien kehittämisen jälkeen, vaan väliin tarvitaan jonkinlainen laadunvarmistusympäristö etenkin lääketieteellisellä alalla.



Kuva 7. Ketterien projektien ja vesiputousprojektien arvontuottaminen suhteessa aikaan [1].

Projektin alussa on mahdollista määritellä niin kutsuttu ”minimum viable product” eli MVP. Koska ketterät menetelmät kehittävät ominaisuuksia vertikaalisti, on tällaisen tuotteen tekeminen usein mahdollista hyvin pienessä määrässä pyrähdyksiä, jotka ovat tyypillisesti kestoltaan 2-4 viikkoa. Vertikaali ohjelmistokehittäminen tarkoittaa sitä, että ominaisuus kehitetään käyttöliittymätasolta pohjalle asti sprintin aikana. Kehitetään siis osa valmista tuotetta. Syy tälle on ketterien menetelmien mukautuminen muutoksiin: kehitetään kaiken aikaa jotain sellaista, mitä asiakas voi testata ja mistä asiakas voi antaa palautetta. Edellä mainittujen syiden vuoksi MVP on mahdollista saada markkinoille par-

haimmassa tapauksessa jo muutamien viikkojen tai parin kuukauden aikana. Tähän vaikuttaa myös se, että Scrumissa ominaisuuksia määritellään sitä mukaa, kun tarpeita ilmenee. Näin koko määrittelyä ei tarvitse olla aluksi kasassa, vaan pelkästään kriittiset ydintoiminnot, joista lähdetään liikkeelle, riittävät. Jo ensimmäisten viikkojen aikana saattaa asiakkaalla ilmetä uusia tarpeita, joita on mahdollista ottaa jo MVP kehitykseen mukaan. Tuotteen nopean markkinoille julkaisun jälkeen voidaan tuotteeseen kehittää pyrähdys kerrallaan ja pala kerrallaan uusia ominaisuuksia, jotka integroidaan jatkuvasti osaksi tuotetta. Tuote on siis käytettävissä jo heti ensimmäisen julkaisun jälkeen, jonka jälkeen sitä päivitetään ja tuotetaan lisää asiakasarvoa paloittain ja ketterästi.

2.4 Lääketieteellisen ohjelmistokehityksen vaatimukset

Lääketieteellisten ohjelmien ja laitteiden kehitykseen liittyy monia sertifionteja ja säädöksiä [15]. Ohjelmiston ja sen elinkaariprosessin tulee aina noudattaa sen maan säädöksiä, missä sitä on tarkoitus käyttää. Kaksi suurinta markkinaa lääketieteellisille laitteille on tällä hetkellä USA ja Eurooppa [55]. Valvovat ja säätelevät elimet ovat USA:ssa FDA (*US Food and Drug Administration*) ja Euroopassa Euroopan komissio.

Lääketieteellisellä laitteella tulee olla CE-merkintä, jotta sitä voidaan myydä Euroopassa [56, 9]. CE-merkintä tarkoittaa, että myytävä tuote täyttää vaaditut turvallisuus- ja terveysmääräykset. Kilpailunäkökulmasta CE-merkintä on tärkeä, sillä sen avulla voidaan varmistaa, että kaikki yhtiöt noudattavat samoja sääntöjä. CE-merkinnän hankkimisessa on karkeasti kolme vaihetta: laitteen luokituksen selvittäminen, haku oikean CE-proseduurin mukaisesti sekä selvitys, jossa todetaan laitteen täyttävän valitun proseduurin mukaiset vaatimukset (*declare conformity*) [33].

CE-merkittäessä lääkinnällistä laitetta voidaan laite luokitella seuraaviin ryhmiin: I, IIa, IIb, III [10]. Luokitus pohjautuu aina aiottuun käyttötarkoitukseen. Jos ohjelmalla ajetaan lääkinnällistä laitetta tai ohjelma vaikuttaa lääkinnällisen laitteen käyttöön, kuuluu ohjelma tällöin saman riskiluokituksen piiriin kuin lääkinnällinen laite. Luokitus riippuu muun muassa laitteen invasiivisuudesta ja siitä, onko laite tekemisissä anatomisesti kriittisten toimintojen kanssa kuten verenkierto tai keskushermosto. Toisin sanoen luokitukseen vaikuttaa se, millaista vahinkoa laitteen käytöstä voi seurata esimerkiksi laitteen vikaantuessa [29]. Esimerkkejä luokituksien laitteista/ohjelmista:

- **Luokka I:** Esim. mittaustarkoituksiin käytettävät laitteet, lääkeannostelijat
- **Luokka IIa:** Esim. sovellus, joka tarkastelee sydämen sykettä rutiinitasolla
- **Luokka IIb:** Esim. sovellus, joka tarkastelee sydämen sykettä kriittisessä hoidossa
- **Luokka III:** Esim. sydämen tahdistimet, kriittisiin toimintoihin liittyvät ohjelmat

Itsenäiset, ajettavat ohjelmistot luokitellaan aktiivisiksi lääkinnällisiksi laitteiksi. Myös ohjelmistojen tapauksessa luokituksessa huomioidaan aiottu käyttötarkoitus ja se, kuinka

kriittisiin toimintoihin ohjelmistolla on mahdollisuus vaikuttaa. Konkreettisesti luokitus vaikuttaa siihen, mitkä Direktiivin 93/42/EEC liitteet koskevat CE-merkinnän hakuprosessia laitteen osalta. Liitteiden sisältö vaihtelee muun muassa valmistajan itse tekemästä ilmoituksesta, että laite on lääkinnällisten laitteiden direktiivin mukainen (liite VII) laitteen täydelliseen laadunvarmistussysteemiin (liite II) [31]. Muut liitteet koskevat laitteen verifiointia (IV), tuotteen laadunvarmistusta (VI) sekä tuotannon laadunvarmistusta (V). Luokitukset ja niiden vaatimat liitteet on esitetty taulukossa 1.

Taulukko 1. CE-merkinnän hakemiseen vaaditut liitteet lääkinnällisen laitteen luokituksen mukaan [10].

CONFORMITY ASSESSMENT PROCEDURES	CLASSES					
ANNEXES	I	I Sterile	I measure	IIa	IIb	III
II (+ section 4)						√
II (- section 4)		√	√	√	√	
III					√	√
IV		√	√	√	√	√
V		√	√	√	√	√
VI		√	√	√	√	
VII	√	√	√	√		

Taulukossa 1 on esitetty direktiivin liitteet ja Eurooppalaiset lääkinnällisen laitteen luokitukset. Luokituksissa on eritelty luokan I laitteista myös steriilit ja mittaustoiminnon sisältävät laitteet. Mitä korkeampi luokitus, sitä useamman liitteen vaatimukset lääkinnällisen laitteen täytyy täyttää.

Luokitus I muodostaa poikkeuksen hakuprosessissa. Sitä koskevat samat yleiset vaatimukset kuin muitakin laiteluokkia [49]. Yleisien vaatimuksien mukaan valmistajan täytyy itse arvioida, mitkä vaatimukset koskevat laitetta. Lisäksi yleisiin vaatimuksiin kuuluu teknisiä asiakirjoja sekä prosessi, jonka avulla valmistaja käsittelee valitukset, vaaratilanteet ja takaisinvedot. Luokassa I valmistaja arvioi itse, täyttyivätkö direktiivin vaatimukset. Ulkopuoliset tahot eivät osallistu arviointiprosessiin. Luokan I valmistaja voi siis saada CE-merkin tuotteelle ja myyntiluvan Eurooppaan ilman ulkopuolista auditointia.

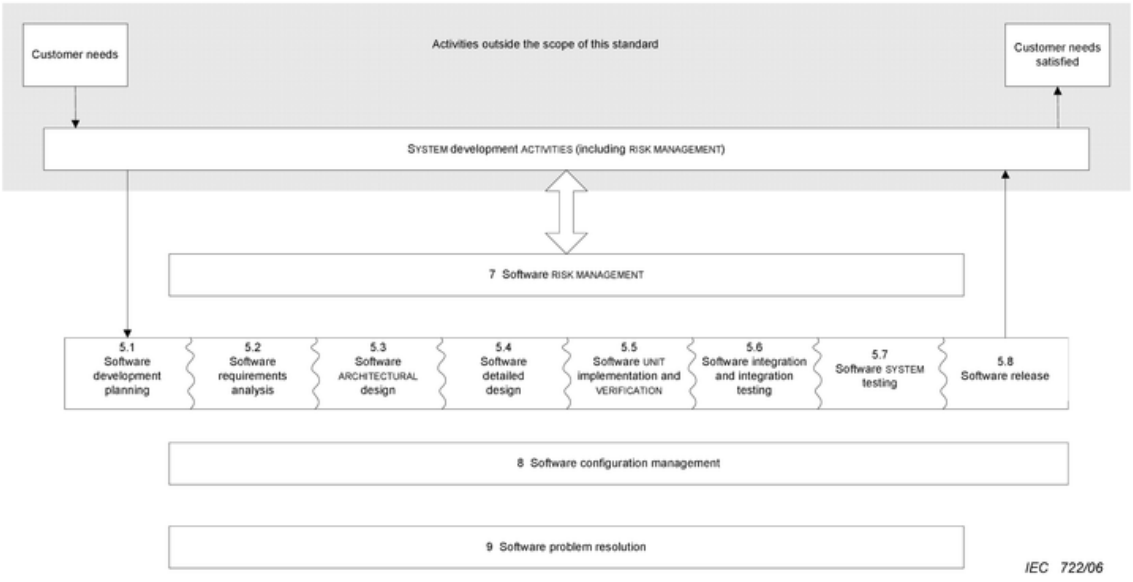
2.4.1 IEC 62304 Ohjelmiston elinkaaren standardi

IEC 62304 on kansainvälinen standardi, joka määrittelee ohjelmistokehityksen elinkaaren vaatimukset lääketieteelliseen käyttöön tarkoitetulle ohjelmistolle [48]. Standardia käytetään useimmissa maissa täyttämään säädösten asettamat vaatimukset. Sen lopullisena tavoitteena on kiinnittää huomiota koko ohjelmiston elinkaareen alkaen suunnittelusta sekä määrittelystä ja päättyen testaukseen sekä verifiointiin tavoitteenaan parantaa ohjelmiston laatua ja turvallisuutta. Vaikka standardin käyttäminen ei ole pakollista ja säädösten vaatimuksien täyttäminen on mahdollista muutenkin, on nykyisin vaikea määritellä vastaavaa tapaa kehittää lääketieteelliset vaatimukset täyttävää ohjelmistoa kuin IEC 62304 standardi ilman, että tulee täyttäneeksi sen vaatimuksia siinä samalla [29].

IEC 62304 esittää vaatimuksien täyttämisen riskilähtöisenä ajattelutapana [30]. Standardin mukaan valmistajan tulee tunnistaa ohjelmiston osat, jotka voivat aiheuttaa riskejä sekä näiden mahdolliset seuraukset. Nämä tulee dokumentoida riskienhallintatiedostoon. Jokaista tunnistettua riskiä kohden täytyy tehdä riskinhallinnallinen toteutus, joka täytyy myös verifioida ja dokumentoida. IEC 62304 luokittelee laitteet (ohjelmistot) riskiryhmiin niiden aiotun käyttötarkoituksen perusteella. Tästä suorana seurauksena korkeimman riskiryhmän laitteet joutuvat käymään läpi tarkemman hyväksymisprosessin.

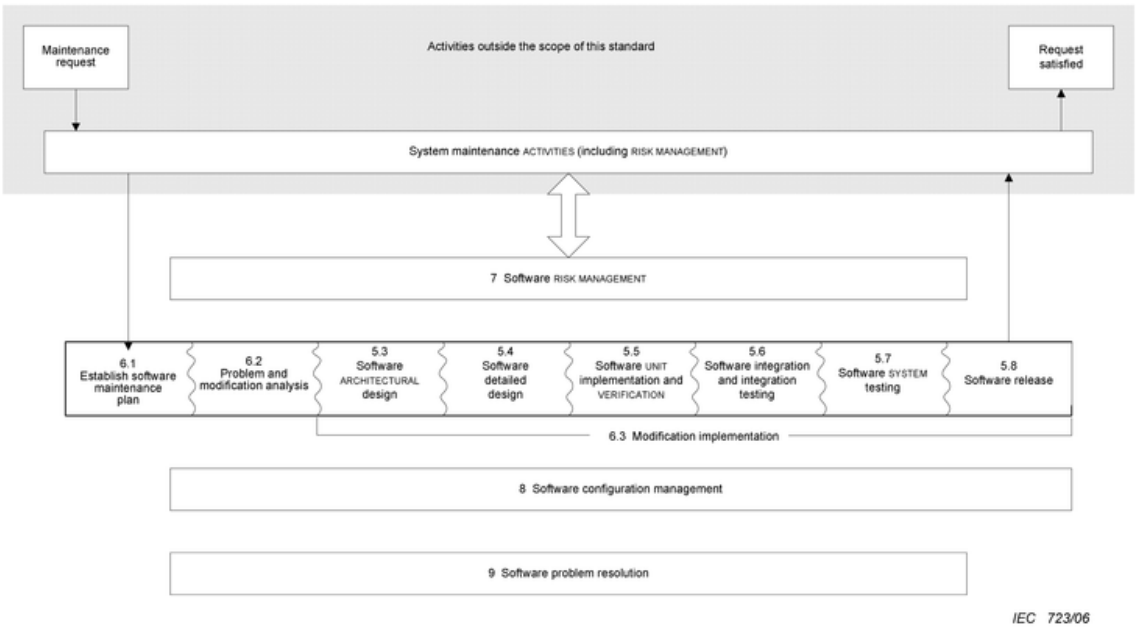
Riskienhallinnan näkökulmasta tarve erillisille ohjelmistot kattaville standardeille syntyy ohjelmistojen erilaisesta luonteesta verrattuna muihin teollisuudenaloihin tai elektroniikkaan [37]. Pelkästään tuotteen testaaminen ei ole riittävää potilasturvallisuuden kannalta, kun ohjelmistoja on mukana. Pienilläkin asioilla voi ohjelmistojen kohdalla olla suuria seurauksia. Tämän vuoksi, vaikka ISO 14971 on pääasiallinen riskienhallinnan standardi, niin IEC 62304 ottaa kantaa riskienhallintaan erityisesti ohjelmistojen näkökulmasta.

IEC 62304 tarjoaa viitekehyksen lääketieteellisen ohjelmiston elinkaarelle ja määrittelee siihen liittyvät prosessit (*processes*) [19]. Standardi määrittelee prosesseille vaatimukset. Jokainen prosessi on jaettu joukkoon aktiviteetteja (*activities*) ja näistä suurin osa on jaettu joukkoon tehtäviä (*tasks*). Kehityksen aikaiset prosessit ja aktiviteetit on esitetty kuvassa 8.



Kuva 8. IEC 62304 ohjelmiston kehitys prosessit ja aktiviteetit [19].

Standardiin kuuluu lisäksi ohjelmiston ylläpitoon liittyvät prosessit ja aktiviteetit [19]. Nämä on esitetty kuvassa 9.



Kuva 9. IEC 62304 ohjelmiston ylläpidon prosessit ja aktiviteetit [19].

Ylläpidon prosesseihin kuuluvat ohjelmiston ylläpidon suunnitelman laatiminen sekä ongelmien ja muutosten analysointi. Näiden jälkeen muutosten toteuttaminen noudattaa ohjelmiston kehityksen prosesseja ja aktiviteetteja.

Standardi ei aseta vaatimuksia valmistajan organisaatiorakenteelle tai sille, mikä osa organisaatiosta suorittaa vaaditut prosessit tai aktiviteetit [19]. Se ei myöskään aseta vaatimuksia sille, kuinka tehtävät tulisi dokumentoida, vaan pelkästään sille, että tehtävät dokumentoidaan. Dokumentaation muoto ja muu paketointi on valmistajan vapaasti valittavissa.

2.4.2 ISO 14971 Riskienhallinta

ISO 14971 standardi määrittää valmistajalle prosessit, joita noudattamalla valmistaja tunnistaa lääkinnällisiin laitteisiin liittyvät riskit, estimoi ja evaluoi riskit ja monitoroi riskien kontrolloinnin tehokkuutta [20]. Standardin vaatimukset ovat voimassa ja niitä täytyy soveltaa koko tuotteen elinkaaren ajan. Toisin sanoen tämä tarkoittaa esimerkiksi sitä, että uusia riskejä saatetaan tunnistaa tuotteen ollessa kehitysvaiheessa ja näitä täytyy tällöin evaluoida ja tehdä kontrollitoimenpiteitä riskien pienentämiseksi. Tuotteen julkaisun jälkeen saattaa löytyä myös uusia riskejä ja vaatimukset riskienhallintaprosessille ovat luonnollisesti tällöinkin voimassa.

Riskienhallintaprosessiin ISO 14971 standardin mukaan kuuluu seuraavat vaiheet: riskienhallinnan suunnittelu, riskien tunnistaminen, riskien arviointi, riskien kontrollointi

sekä jäännösriskien ja niiden hyväksyttävyyden arviointi [21]. Näistä tuotetaan riskienhallintaraportti sekä riskienhallintatiedosto, jota täytyy ylläpitää koko tuotteen elinkaaren ajan.

Nykyisin on käytössä kaksi eri versiota ISO 14971 -standardista: ISO 14971:2007 ja EN ISO 14971:2012 [20]. Jälkimmäinen näistä sisältää niin kutsutut "Z-liitteet" ja standardin käyttö onkin pakollista Euroopassa. Samaa standardia voi noudattaa myös USA:n sekä muiden valtioiden kuten Kanadan ja Japanin markkinoilla, mutta koska erona aikaisempaan standardiin 2012 versiossa on lisätyt liitteet, vuoden 2007 standardi on hyväksytty ja yleisemmin käytetty muilla kuin Euroopan markkinoilla.

2.4.3 ISO 13485 Laadunhallintajärjestelmä

ISO 13485 standardi määrittelee vaatimukset laadunhallintajärjestelmälle ja sille, mitä organisaation tarvitsee tehdä näyttääkseen, että se pystyy valmistamaan lääketieteellisiä laitteita tai -palveluja, jotka täyttävät asiakas- ja viranomaisvaatimukset [18]. Keskeisiä asioita standardissa ovat muun muassa laadunhallintajärjestelmä, johdon vastuut sekä tuotekohtaiset vaatimukset ja näiden kaikkien dokumentointi. Tärkeitä toimintoja ISO 13485 standardissa on muun muassa [45]:

- Laadunhallintajärjestelmän kehittäminen
    - Dokumentointi
- Laadunhallintajärjestelmän dokumentointivaatimusten määritteleminen
- Laadunhallintajärjestelmän tarvitsemien prosessien määrittäminen
- Laatuohjeen tekeminen, jonka avulla laadunhallintajärjestelmän asettamia vaatimuksia noudatetaan
- Laadunhallintatiedoston luominen ja sen ylläpitäminen jokaiselle lääketieteelliselle laitteelle tai lääketieteellisten laitteiden tyypille
- Johdon vastuisiin kuuluu:
    - Sitoutumisen näyttäminen, laadunhallintajärjestelmän tukeminen niin kehitys-, toteutus- kuin ylläpitovaiheessakin
    - Laatutavoitteiden määritteleminen
    - Laadun suunnittelun jatkaminen: kuinka kehittää laadunhallintajärjestelmää ja sen dokumentointia, valvontaa ja muita siihen liittyviä asioita jatkuvasti

Lisäksi standardi määrittää joukon muita asioita ja prosesseja, jotka organisaation täytyy toteuttaa pystyäkseen kehittämään laadukasta lääketieteelliseen käyttöön tarkoitettua laitetta tai ohjelmistoa. Sekä USA:n että Euroopan markkinoilla täytyy noudattaa ISO 13485 standardin mukaista laatujärjestelmää [55]. ISO 13485 standardia käytetään yhdessä muiden standardien lisänä lääketieteellisessä ohjelmistokehityksessä.

2.4.4 FDA vaatimukset

Jotta tuotetta voidaan myydä USA:ssa, täytyy valmistajan rekisteröidä se FDA:lle [54]. FDA on julkaissut listan ohjeistavista dokumenteista valmistajia varten. FDA vaatii, että lääkinnällisestä laitteesta tehdään ns. "Design History File", johon dokumentoidaan kaikki lääkinnällisen laitteen vaatimukset ja suunnitelmat [60]. Tiedosto tulee säilyttää 12 vuotta lääketieteellisen laitteen elinkaaren päätöksestä.

FDA-säädellyillä markkinoilla riskienhallinta alkaa tuotteen aiotusta käyttötarkoituksesta [14]. Valmistajan tulee listata aiotut käyttötapaukset ja ennakoitavissa olevat väärinkäyttötavat. ISO 14971 standardi tarjoaa näiden määrittelyyn apua kysymyssarjan muodossa, johon vastaamalla valmistaja voi tunnistaa tuotteeseen liittyviä riskejä. Tietojen perusteella kootaan lista riskeistä. FDA luokittelee laitteet lähestulkoon samalla tavoin kuin Euroopan komissio: FDA käyttää luokituksia I, II ja III [59]. Luokituksessa ei siis ole jaettu tasoa II kahteen alatasoon a ja b, mutta tasot tarkoittavat samaa kuin Euroopassakin. Tason I pitävät sisällään vähiten riskiä ja tason III laitteet eniten.

Riskiluokituksen I ja II laitteille riittää markkinointia varten niin kutsuttu ”Premarket notification”, joka tehdään FDA:lle [59]. Sen tarkoituksena on osoittaa dokumenttien avulla, että myytäväksi tarkoitettu laite on turvallinen ja toimiva. On kuitenkin huomattava, että suurin osa luokituksien I ja II laitteista on vapautettu tästä velvoitteesta. Korkeimman riskiluokituksen III laitteet joutuvat käymään läpi tarkemman hyväksyntäprosessin. Tätä kutsutaan termillä ”Premarket Approval”. Kuten Euroopassa, niin myös USA:ssa suurin osa riskiluokituksen I laitteista on vapautettu laatujärjestelmää koskevista velvoitteista [58]. Poikkeuksen muodostavat esimerkiksi steriilit laitteet. Laatujärjestelmä toteutetaan lääkinnällisten laitteiden tapauksessa useimmiten standardin ISO 13485 avulla.

FDA vaatimukset ohjelmiston osalta ovat hyvin samanlaiset kuin Euroopan vaatimukset [57]. Vaaditut aktiviteetit kuten suunnittelu, verifiointi, testaus, jäljitettävyys, konfiguraatioiden hallinta sekä riskienhallinta, ovat samoja aktiviteetteja tai tuotteen elinkaaren läpi jatkuvia prosesseja kuin IEC 62304 standardissa. Myös samaan tapaan kuin Euroopassa, FDA ei vaadi käytettäväksi tietynlaista ohjelmiston elinkaaren mallia. Valmistaja voi siis päättää, kehittääkö ohjelmaa vesiputousmallin mukaisesti vaiko esimerkiksi ketterillä menetelmillä.

Yksityiskohtaisemmalla tasolla FDA vaatimukset ohjelmistojen osalle ottavat kantaa tärkeiksi havaittuihin asioihin ohjelman turvallisuuden kannalta [57]. Ohjeistuksesta poimittavia keskeisiä asioita ovat muun muassa ohjelmiston monimuotoinen testaus, vaatimusten validointi, muutoksenhallinta sekä toteutettujen muutoksien toimivuuden verifiointi, riskien arviointi ja hallinta sekä suunnitelman että toteutuksen pohjalta. Samat asiat nousevat esille myös Eurooppalaisissa vaatimuksissa.

2.4.5 IEC 82304 Terveysalan ohjelmistot

IEC 82304 käsittelee IEC 62304 standardia laajemmin terveysalan käyttöön tarkoitettua itsenäisesti ajettavaa ohjelmistoa [23]. Se on marraskuussa 2016 julkaistu standardi [34]. IEC 62304 käsittelee ohjelmistotuotteen valmistamiseen käytettäviä prosesseja, kun taas IEC 82304 ottaa kantaa laajemmin tuotteeseen. Standardin kuvaus ”Terveysalan ohjelmat: Yleiset vaatimukset tuotteen turvallisuudelle (*engl. Health Software: General requirements for product safety)*” kuvaa sitä näkökulmaa, kuinka IEC 82304 eroaa IEC 62304 standardista. Koska standardi on niin uusi, sitä ei ole vielä harmonisoitu EU:n toimesta eikä FDA ole vielä tunnustanut sitä [34].

3. TUTKIMUSKYSYMYKSIEN ESITTELY

Työn tarkoituksena on päivittää ylläpitovaiheessa olevan lääketieteellisen projektin prosessimalli. Projektissa on käytössä paljon hyviä käytänteitä, mutta selvityksen tuloksena saadaan tietää, mitkä käytännöt jäävät käyttöön päivityksen jälkeen. Samalla on kuitenkin hyvä huomioida tulevia projekteja ajatellen myös projektin aloitus- ja lopetusvaiheet prosessissa. Nämä pohjautuvat kirjallisten löydöksien hyväksi havaittuihin käytäntöihin.

Työssä tarkastellaan erästä ylläpitovaiheessa olevaa projektia ja siinä käytettyjä prosesseja sekä menetelmiä lääketieteellisen ohjelmistokehityksen näkökulmasta. Ylläpitovaiheessa olevasta projektista on kiinnostava tarkastella, mitä prosessien näkökulmasta tarkoittaisi, jos tuote vietäisiin muille markkinoille. Kyseinen tuote on myynnissä Euroopan markkinoilla, mutta kysymys muista markkinoista muuttuisi relevantiksi, jos valmista tuotetta markkinoitaisiin esimerkiksi USA:han. Tämän vuoksi työssä tarkistellaan standardoinnin eroja USA:n ja Euroopan markkinoiden välillä ja sivutaan hieman myös Aasiaa.

Kyseistä ylläpitovaiheessa olevaa projektia on kehitetty ketterästi. Ketteryydestä on muotoutumassa vallitseva suuntaus myös säädellyillä aloilla, joten tutkimuskysymyksen näkökulmasta on oleellista tarkastella nimenomaan ketterää lääketieteellistä ohjelmistokehitystä. Tutkimuskysymyksen asetteluun vaikuttaa myös se, että standardit on kehitetty aikana, jolloin vesiputousmallin ohjelmistokehitysprosessit olivat valtavirtaa ja niitä pidettiin parhaimpina käytäntöinä [4]. Näin ollen ketteriä prosesseja ja standardeja täytyy sovittaa toisiinsa, jotta niistä saadaan aikaan toimiva kokonaisuus.

Viimeisimpänä asiana työssä käsitellään prosessien liiketoimintavaikutuksia. Ketterillä menetelmillä ja vesiputousmallin ohjelmistokehitysprosesseilla on todettu olevan liiketoiminnallisia eroja projektien keston ja kustannusten suhteen. Lisäksi lääketieteelliset vaatimukset tuovat mukanaan liiketoimintavaikutuksia ketterään prosessiin. Työssä tarkastellaan näiden lisäysten aiheuttamia vaikutuksia prosessiin ja tätä peilataan vesiputousmallin ja ketterien menetelmien liiketoimintaeroihin.

Tutkimuskysymykset ovat:

- Nykyisen ylläpitovaiheen prosessimallin hyvien käytänteiden selvittäminen ja mallin päivittäminen kirjallisuuteen pohjautuen
- Scrum-malliin lääketieteellisten vaatimusten vuoksi lisättävien prosessien selvitys
- Selvitys siitä, voidaanko samaa prosessimallia käyttää Euroopan ja USA:n markkinoilla standardoinnin eroavaisuuksien puolesta

- Prosessimallin liiketoiminnalliset tavoitteet ja lääketieteellisten lisäysten vaikutukset liiketoimintaan

3.1 Tutkimuskysymyksien taustaa

Lääketieteellinen ohjelmistokehitys on tarkoin säädeltyä. Ohjelmistoja täytyy kehittää vaadittujen prosessien mukaisesti, joiden avulla yritetään varmistaa ohjelmistojen parempi laatu. Laadun parantaminen vähentää ohjelmistoista johtuvia tapaturmia. Huomion kiinnittäminen tähän seikkaan on lisääntynyt 2000-luvun puolella, jolloin yhä useammissa lääketieteellisissä laitteissa on mukana koodia. Edellä mainittujen asioiden vuoksi myös tarve erityisesti ohjelmistoja koskeville standardeille on syntynyt ja koska operoidaan nopeasti muuttuvalla alalla, standardit saattavat muuttua melko lyhyissä sykleissä.

Lääketieteelliseen toimialaan liittyy useita eri standardeja, kuten luvussa 2 todettiin. Standardit eivät ole vielä nykyisin globaalisti harmonisoituja: eri valtioissa ja talousalueilla on käytössä eri standardeja eikä tiukkaa, globaalisti yhtenäistä linjaa ole olemassa. Tämän vuoksi prosessimallia mietittäessä on tärkeä huomioida, millä alueella ohjelmistoa halutaan ensisijaisesti myydä, sillä tämä vaikuttaa oleellisesti noudatettavien standardien, ja tätä kautta noudatettavien prosessien, valintaan.

Lääketieteellisten ohjelmistojen suurimmat markkinat ovat tällä hetkellä USA ja Eurooppa [6]. Tämän vuoksi erityisesti suomalaisesta ja eurooppalaisesta näkökulmasta on kiinnostavaa, kuinka standardointi näiden markkinoiden välillä eroaa. Nämä markkinat ovat myös niitä, joihin lääketieteellisten laitteiden kehitys Suomesta tai Euroopasta useimmiten suuntautuu. Tätä voi perustella esimerkiksi sillä, että Aasia ollessaan iso markkina, on vielä melko hajallaan ja esimerkiksi Kiina on todella haastava kohde laitteiden tai ohjelmistojen myynnille. Aasian talousalueella standardointi lääketieteellisten laitteiden osalta vaihtelee suuresti maakohtaisesti, mikä tekee laitteiden kehityksen laajalle markkinalle haastavaksi. Vaikka markkina onkin laaja, pilkkoutuu se standardoinnin vuoksi pienempiin palasiin ja muuttuu vähemmän houkuttelevaksi kohteeksi. Näin ei ole Euroopassa eikä Yhdysvalloissa, sillä molemmilla maantieteellisillä alueilla on oma sääntelevä elimensä, joka hallinnoi muun muassa lääketieteelliseen standardointiin kuuluvia asioita. Euroopassa tämä on Euroopan komissio (European Commission, EC) ja Yhdysvalloissa Food and Drug Administration, FDA.

Koska standardointi vaihtelee maantieteellisesti, on tärkeää selvittää, kuinka paljon erilaisia prosesseja vaadittaisiin, jotta tuotetta voitaisiin myydä toisella markkinalla. Tällä on vaikutusta heti tuotteen elinkaaren alussa noudatettavia prosesseja valittaessa, jos aikeissa on tuotteen myyminen heti useammalla markkinalla. Toisaalta tällä on vaikutusta myös siinä vaiheessa, jos tuotetta on myyty jo yhdellä markkinalla ja on aikeissa ryhtyä myymään tuotetta toisella markkinalla.

Standardien lisäksi kiinnostava asia on, että säädellyillä aloilla ohjelmistoja on kehitetty ja kehitetään edelleen vesiputousmallin mukaisesti, vaikka ei-säädellyillä aloilla on pystytty osoittamaan ketterien menetelmien etuja verrattuna vesiputousmallin ohjelmistokehitykseen [27]. Tapausesimerkkejä näistä on luvussa 3.1. Tästä pääteltynä voitaisiin olettaa, että ketteristä menetelmistä voisi olla tuotavissa käytänteitä ja hyötyjä myös säädellyille aloille, joita lääketieteellinen ohjelmistokehitys edustaa.

3.2 Olemassa olevan tutkimuksen esittely

Ketteristä ohjelmistokehitysprosesseista lääketieteellisessä ohjelmistokehityksessä on tehty tutkimuksia aikaisemmin. On olemassa muun muassa tutkimuksia, joissa esitetään tietynlaista viitekehystä ketterään lääketieteelliseen ohjelmistokehitykseen kohteelle sopivalla tavalla. Tällaisista tutkimuksista löytyy tietoa esimerkiksi siitä, millaisia prosessimalleja on käytetty vastaamaan lääketieteellisiin vaatimuksiin lisäämällä ketteryyttä vaiheesta toiseen etenevissä ohjelmistokehitysmalleissa kuten vesiputousmalli tai V-malli. On olemassa myös tutkimuksia siitä, kuinka osassa prosessista hyödynnetään ketteristä menetelmistä peräisin olevia työkaluja.

Purojärvi [43] ja Baggström [4] käsittelevät tahoillaan ketterää lääketieteellistä ohjelmistokehitystä. Kumpikin heistä on täydentänyt ketterää mallia lääketieteellisiin vaatimuksiin sopivaksi. Purojärvi ottaa huomioon lääketieteellisen arkkitehtuurivaatimuksen esittelemällä ”Pyrähdys 0” -käsitteen täydennettyssä Scrum-mallissa. Baggström puolestaan lähestyy asiaa julkaisusuunnittelun kautta. Tämän lisäksi molemmat käsittelevät lääketieteellisen ohjelmistokehityksen erityispiirteitä kuten riskienhallinta ja jäljitettävyys.

3.2.1 Miksi vesiputousmallia käytetään vallitsevasti säädellyillä aloilla

Vesiputousmallin kulku mukailee valmiiksi vaadittua, formaalia validointi- ja arviointiprosessia, joka lääketieteellisen ohjelmistokehitykseen liittyy [24]. Vesiputousmallissa välietappien (vaatimusmäärittely, suunnittelu, toteutus, testaus, käyttöönotto ja ylläpito) tuotokset ovat valmiiksi siinä muodossa, että ne on helppoa arvioida ja validoida. Lääketieteelliset standardit ovat osin kirjoitettu niihin aikoihin, kun vesiputousmalli oli vallitseva suuntaus [4]. Tämä selittää osaltaan vesiputousmallin laajan käytön säädellyillä aloilla.

On olemassa viitekehys nimeltä MDevSPICE [55]. Viitekehys on suhteellisen tuore ja sitä on vielä vuoden 2016 puolella kehitetty kattamaan paremmin mobiilien lääkinnällisten ohjelmistojen tarpeet. Nykyisessä muodossaa MDevSPICE kokoaa yhteen kaikki lääketieteellisen laitteen kehitykseen vaadittavat prosessit yhden viitekehyksen alle. Viitekehys kuitenkin pohjautuu toistaiseksi paljolti V-malliin. Tämän hetkisessä mobiiliohjelmistoille suunnitellussa versiossa siinä on käytössä niin kutsuttu "vähennetty V-malli",

johon on lisätty joitain ketteriä periaatteita. Tulevaisuuden tarkoituksena MDevSPICEN kehittäjillä on viedä mallia vielä ketterämpään suuntaan, jotta lääketieteellisiä mobiilisovelluksia olisi mahdollista julkaista markkinoille nopeammin kuitenkin kattaen lääketieteelliset vaatimukset.

3.3 Ketterien menetelmien havaitut edut olemassa olevissa tutkimuksissa

McHugh et al. huomasivat omassa tutkimuksessaan, että vesiputoustyyppisiä ohjelmistokehitysmalleja voitaisiin korvata ketterämmillä malleilla myös säädellyillä aloilla [27]. Heidän tutkimuksensa aiheena oli "Ketterä V-malli lääketieteelliseen ohjelmistokehitykseen". Keskeinen havainto heidän tutkimuksessaan oli, että muutosten toteuttaminen luo paljon ylimääräisiä kustannuksia ja tästä johtuen muutosten hallinta täytyy toteuttaa paremmin. Vesiputousmallin kaltainen prosessi on todella kankea erityisesti muutostenhallinnassa [24]. Tästä syntyvä paine ohjautuu suoraan kustannuksiin, ajallisiin resursseihin sekä työvoimaresursseihin. Tämän vuoksi jo ei-säädellyillä aloilla laajalti käytössä olevat ketterät ohjelmistokehitysmenetelmät ovat alkaneet siirtyä käyttöön myös säädellyillä aloilla, joihin myös lääketieteellinen ohjelmistokehitys lukeutuu.

On olemassa tutkimuksia ei-säädellyiltä aloilta, joissa on pystytty osoittamaan, että ketteriä menetelmiä käyttämällä voidaan saada huomattavia etuja verrattuna vaiheesta toiseen eteneviin ohjelmistokehitysmenetelmiin [27]. Näitä etuja ovat muun muassa alhaisemmat kustannukset, tuotteen nopeampi julkaisu markkinoille ja parempi laatu. Kuitenkaan ketterät menetelmät eivät ole laajalti käytössä lääketieteellisessä ohjelmistokehityksessä. Tähän on etsitty syitä, mutta tyhjentävää vastausta ei ole löytynyt. Yksi mahdollinen selitys on kuitenkin, että ketterät menetelmät vaikuttavat ensinäkemältä olevan ristiriidassa lääketieteellisten vaatimusten kanssa. Näin ei kuitenkaan ole, sillä ketteriä menetelmiä on käytetty lääketieteellisen ohjelmistokehityksen saralla menestyksekkäästi kuten vaikkapa Abbott Diagnosticsin tapaus myöhemmin samassa luvussa osoittaa.

3.3.1 Abbott Diagnostics

Eräässä tutkimuksessa yritys nimeltä Abbott Diagnostics vertasi kahta projektia toisiinsa [46]. Toinen näistä vedettiin läpi vesiputoustyyppisellä ohjelmistokehitysmallilla ja toinen käyttäen ketteriä menetelmiä. Molemmat projektit saivat FDA-hyväksynnän. Yritys teki havainnon, että monimutkaisten lääketieteellisten laitteiden ohjelmistokehitysprosessit voivat kestää kolmesta viiteen vuotta. Tänä aikana on ymmärrettävää, että vaatimukset muuttuvat. Niinpä myös heidän mielestään yksi avaintekijöistä lääketieteellisissä ohjelmistokehitysprojekteissa on muutosten hallinta. Mikä heidän tutkimuksessaan on kuitenkin merkittävää, on se, että vaikkei tarkkaa numeerista dataa olekaan saatavilla, niin he saavuttivat ketterillä menetelmillä huomattavia kustannussäästöjä.

Abbott Diagnosticsin vertailemat projektit olivat erikokoisia, mutta ketterän projektin läpivientiin käytettiin vain 24 kuukautta aikaa, kun taas vesiputoustyyppisen projektin läpivientiin käytettiin 54 kuukautta. He arvioivat, että pidemmän projektin kestoa olisi voitu lyhentää 20–30% käyttämällä ketteriä menetelmiä. Samoin henkilöstöjen koossa projektien välillä oli huomattavia eroja: ketterä projekti työllisti keskimäärin 20 henkilöä kun taas vesiputousmallin projektin parissa työskenteli keskimäärin 60 henkeä. Myös tämän osalta he arvioivat, että työvoimassa olisi voitu säästää 20–30% käyttämällä ketteriä menetelmiä. Yhteissäästöiksi kustannusten osalta he arvioivat 35–50% ketterän projektin eduksi.

3.3.2 Cochlear

Yritys nimeltä Cochlear ryhtyi 2000-luvun alkupuolella käyttämään Scrumia lääketieteellisessä ohjelmistokehityksessään [44]. Alun perin heilläkin oli käytössään V-mallin ohjelmistokehitysprosessi, jota he asteittain veivät ketterämpään suuntaan ja päätyivät lopulta käyttämään Scrumia. He kiinnittävät tutkimuksessaan erityistä huomiota työkalujen valintaan ja mahdollisimman monen asian automatisointiin. Työkaluilla he hallitsivat muun muassa vaatimuksia ja näiden jäljitettävyyttä. Koodin tasolla automatisoituja asioita olivat heillä testit ja näiden tulokset sekä jatkuva integraatio yöllisten versioiden ja näiden testauksen avulla. Heidän näkemyksensä mukaan IEC 62304 standardissa on paljon sellaista, mitä voidaan toteuttaa ketterillä menetelmillä ja vieläpä suhteellisen yksinkertaisesti. Heillä Scrumin käyttö on näkynyt laadun ja tuottavuuden harppauksessa.

3.3.3 Angiografia projekti

Erään case-tutkimuksen tuloksena kehitettiin hybridi prosessimalli käyttäen ketteriä menetelmiä sekä formaalia vesiputoustyyppistä prosessimallia [24]. Kuten aikaisemmin esitellyt tutkimukset, myös tämä tutkimus lähti liikkeelle siitä näkökulmasta, että formaalit vesiputoustyyppiset ohjelmistokehitysprosessit lisäävät ylimääräisiä kustannuksia prosessiin eivätkä prosessin tuottamat artefaktit tuo lisäarvoa asiakkaalle. Tutkimuksen laitetta oli tarkoitus myydä USA:n markkinoilla, joten sen täytyi täyttää FDA vaatimukset ja yrityksen täytyi pystyä dokumentein osoittamaan, että laatustandardien mukaisia prosesseja noudatetaan.

Koska tutkimuksen ryhmä oli aikaisemmin käyttänyt ketteriä menetelmiä onnistuneesti, tekivät he olettamuksen, että he pystyvät tuottamaan korkealaatuista ohjelmistoa, joka täyttää paremmin markkinoiden vaatimukset, ketterien menetelmien avulla. Heidän täytyi ketterien menetelmien käytön lisänä kuitenkin pystyä osoittamaan, että FDA vaatimukset täyttyvät. Ennen kaikkea tämä lisää oleellisesti dokumentaation määrää. Kyseisen tutkimuksen lähestymistapa oli, että ohjelmistoa kehitetään ketterästi ja dokumentaatiota hallitaan vesiputousmaisesti. Tällä yritetään ennen kaikkea tasapainottaa ketterää joustavuutta ja vaadittua formaalisuutta prosessissa.

3.4 Ketterän lääketieteellisen ohjelmistokehityksen haasteet

Lääketieteelliset standardit on kirjoitettu aiemmin vallinneiden valtavirran ohjelmistokehityssuuntausten mukaisesti [4]. Keskeinen haaste on sovittaa standardien vaatimukset, kuten eräät prosessit, osaksi ketterää prosessia. Käytännössä tämän työn osalta tämä tarkoittaa lisäprosessien integroimista valmiiseen ketterään projektimalliin kuten Scrumiin. Lisäprosessien voidaan katsoa ”kuormittavan” ketterää prosessia ja siksi integrointi täytyy tehdä siten, että ne tuntuvat mahdollisimman luonnolliselta osalta muuta prosessia.

Liiketoimintapuolella eritoten ylläpitovaiheessa olevan projektin näkökulmasta lääketieteellisiä ohjelmistoja kehittävien yritysten voi olla vaikea siirtyä perinteisistä ohjelmistokehitysmenetelmistä (*engl. plan driven*) ketteriin menetelmiin kesken tuotteen elinkaaren. Tuote voi olla esimerkiksi jo saanut säätelevän elimen hyväksynnän ja tämä voisi olla vaikeaa tai kustannustehotonta hankkia uudestaan uusilla menetelmillä [27]. Vaihtoehtoisesti yrityksessä on voitu harjoittaa esimerkiksi vesiputousmallia monia vuosia, jolloin menetelmä on nykyiselle henkilökunnalle ja organisaatiolle jo tuttu. Tällöin siirtyminen ketteriin menetelmiin esimerkiksi uuden tuotteen elinkaaren alussa voidaan kokea epämukavaksi, ylimääräiseksi riskiksi eikä potentiaalisesti saavutettava hyöty ole tarpeeksi suuri, jotta menetelmää vaihdettaisiin.

3.5 Liiketoimintanäkökulma

Lääketieteellinen teknologia on Euroopan innovatiivisin teollisuuden ala, mikäli tätä mitataan patenttihakemusten määrässä [12]. Kaikista aloista juuri lääketieteellisille teknologisille innovaatioille haetaan vuosittain eniten patentteja Euroopassa [11]. Tämä suuntaus on toistunut ainakin vuosina 2012–2015. Lisäksi tällä välillä lääketieteellisen sektorin patenttihakemusten määrä on lisääntynyt vuosittain. Talouden kriiseistä huolimatta, lääketieteellisen teknologian saralla on pystytty vuosittaiseen kasvuun ja yksi päätekijä tähän on innovatiivisuus. Alan liikevaihdosta noin 30 % muodostuu viimeisen kolmen vuoden aikana markkinoille julkaistuista laitteista [12].

Ala on kilpailtu ja kustannuspaineet ovat suuret. Valmistajien täytyy pystyä valmistamaan laadukkaita ja turvallisia tuotteita nopeammin, pienemmillä kustannuksilla ja nopeammissa tuotesykleissä, yhä säädellymmässä toimintaympäristössä [12]. Tämän vuoksi ketterät menetelmät ovat vahva kandidaatti tarjoamaan ratkaisun haasteisiin. Esimerkiksi Scrum projektiviitekehys on vahvimmillaan monimutkaisissa projekteissa, joiden alkuvaiheissa on varsinkin paljon tuntematonta. Syynä tähän on Scrumin tarjoama läpinäkyvyys ja mukautumiskyky, mikä on vahvuus etenkin uusia tuotteita kehittäessä. Tämä puolestaan tukee erittäin innovatiivista alaa.

Liiketoiminnan näkökulmasta lääketieteellinen ohjelmistokehitys on prosessien puolesta raskasta. Standardien noudattaminen, ja sitä kautta hyväksyttävän ohjelmistokehityspro-

sessin toteuttaminen, lisää tavanomaisiin projekteihin verrattuna aktiviteetteja, joita säätelemättömien alojen ohjelmistokehitysprosesseissa ei tarvitsisi tehdä. Näitä aktiviteetteja ovat muun muassa lisääntyneet dokumentointi- ja testausvaatimukset. Standardit ja näiden myötä lisääntyneet suoritettavat prosessit kasvattavat siis kustannuksia.

Organisaatioiden, jotka ovat uusia tai joilla on vain vähän kokemusta alalta voi olla vaikea todentaa IEC 62304 vaatimusten noudattaminen [50]. Liiketoiminnan ja kustannusten näkökulmasta tämä tarkoittaa sitä, että standardien sertifiointi aiheuttaa kustannuksia sekä hankintavaiheessa että ylläpitovaiheessa. Itse sertifikaatin hankinta ja sen auditointi on suhteellisen vähäinen kertainvestointi. Kuitenkin organisaation sisäiset toimet, jotka valmistavat siihen, että organisaatio on auditointikelpoinen voivat olla kyseistä kertakustannusta korkeammat. Tämä riippuu organisaation aikaisemmista toimintatavoista ja organisaation koosta siinä määrin, että molemmat osatekijät vaikuttavat osaltaan sisäiseen työmäärään sertifikaatin saamiseksi. Uusien toimintatapojen omaksuminen organisaatiotasolla ei ole aina helppoa eikä tapahdu hetkessä. Sertifikaatin hankkimisen jälkeen organisaation tulisi pystyä pitämään toimintatavoista kiinni ja läpäistä ylläpitovaiheen tarkastukset. Ylläpitovaiheen tarkastuksista lankeaa myös pienehköjä kustannuksia, joiden voidaan laskea kuuluvan sertifikaattien vuotuisiin ylläpitokustannuksiin.

Ketterä ohjelmistokehitys on omalta osaltaan ja ohjelmistokehitysprosessien näkökulmasta pyrkinyt vastaamaan kustannushaasteeseen. Ei-säädellyillä aloilla on pystytty osoittamaan, että ketterät menetelmät ovat kustannustehokkaampia kuin esimerkiksi perinteisemmän tyyliset ohjelmistokehitysmallit kuten vesiputousmalli [27]. Liiketoimintanäkökulmasta on myös tärkeää ketterien menetelmien tavoite nopeampaan tuotteen markkinoille vientiin sekä niiden pyrkimys pienentää riskiä koko projektin elinkaaren ajalta.

Näitä näkökulmia yhdistelemällä voidaan päätellä, että olisi tarkoituksenmukaista kehittää mahdollisimman kustannustehokas prosessimalli, joka kuitenkin kattaa lääketieteelliset vaatimukset ja jonka seurauksena saadaan tuotettua laadukkaita ohjelmistoja. Eri lähteistä on löydettävissä hyviä käytäntöjä sovellettaviksi ketteriin projekteihin. Näitä hyviä käytäntöjä yhdistelemällä ja ketterään prosessiin lääketieteellisien vaatimuksien mukaisia askelmia lisäämällä on aiempienkin tutkimuksien valossa mahdollista kehittää ketterä prosessimalli, joka kattaa lääketieteelliset vaatimukset ja on samalla kustannuksiltaan alhaisempi kuin vastaava vesiputoustyyppinen projektimalli. Myös muitakin projektihyötyjä on saavutettavissa.

4. PROSESSIMALLI

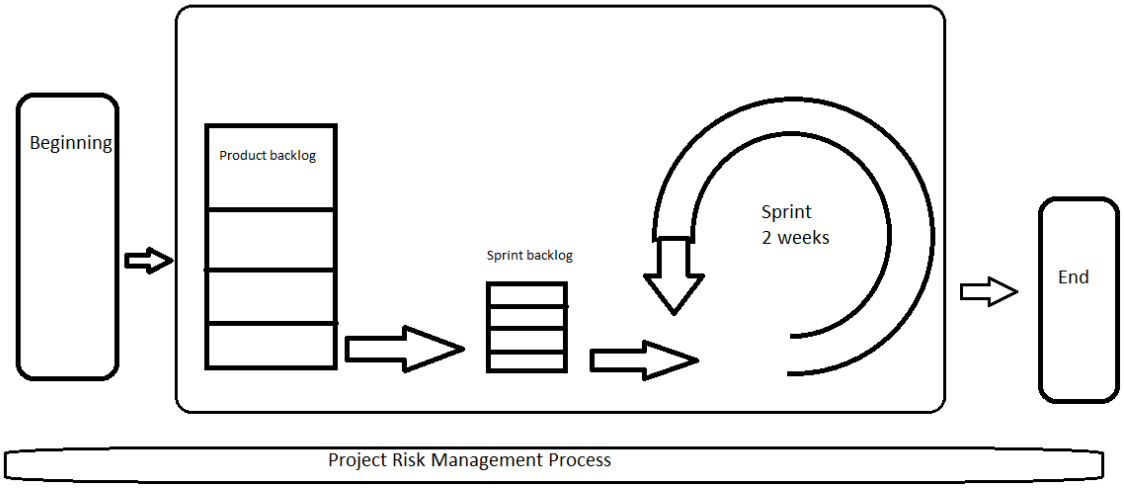
Scrum ei ole yksinään sopiva lääketieteelliseen ohjelmistokehitykseen, sillä se ei prosessimallina täytä kaikkia lääketieteellisen ohjelmistokehityksen vaatimuksia. Siihen on tehtävä lisäyksiä, jotta lääketieteelliset vaatimukset täyttyvät. Tämän lisäksi lisäykset täytyy integroida osaksi ketterää prosessia, jotta ketterästä prosessista voidaan täysimääräisesti hyötyä. Tämä tarkoittaa sitä, että lisäysten jälkeenkin prosessin tulisi säilyä ketteränä, vaikka työmäärä eri vaiheissa saattaakin lisääntyä. Prosessimallin tulisi kattaa kaikki ohjelmiston elinkaaren vaiheet [15]. Säädellyillä aloilla keskeisiä elinkaaren osia ovat muun muassa riskienhallinta, verifiointi ja validointi.

Luvussa käsitellään ensin nykyinen prosessimalli ja erotellaan sen tärkeimmät osat. Alaluvuissa eritellyt osat ovat sellaisia, joita Scrum ei prosessimallina valmiiksi tunne. Tämän jälkeen kuvataan lisäykset prosessimalliin kirjallisuuteen pohjautuen. Lopuksi tehdään yhteenveto vaadituista standardeista ja verrataan maantieteellisten alueiden kuten EU ja USA standardoinnin eroja toisiinsa. Ratkaisun arvioinnissa käsitellään prosessimalliin lisättyjen osasten ja standardien liiketoimintavaikutuksia verrattuna säätelemättömään ohjelmistokehitykseen ja säädeltyyn vesiputousmallin ohjelmistokehitykseen.

4.1 Nykyinen ylläpitovaiheen prosessimalli

Ylläpitovaiheen prosessimallilla tarkoitetaan tässä kohdin sellaista prosessia, jonka käytännöt ovat varsin vakiintuneet ja jossa tuote, eli tässä tapauksessa ohjelma, ei enää arkkitehtuurin suhteen koe merkittäviä muutoksia. Ylläpitovaiheen prosessimalli ei työn kontekstissa huomioi myöskään projektin aloitusta tai pienemmissä määrin lopetusta, vaikka lopetus ja ainakin nykymuotoisen tuotteen alasajo on tulevaisuudessa edessä. Tuote voi toki uudistua ja voi alkaa uusi projekti mukaillen vanhan pohjaa tai asiakkaalle tarjottavaa arvoa, sillä ohjelmistotuotteetkin vanhenevat.

Ylläpitovaiheen prosessimallia voisi kuvata kuvan 10 mukaiseksi. Prosessimallissa ensimmäisenä on aloitusvaihe, joka vielä tässä kohdin käsittelyä ohitetaan. Aloitusvaiheen jälkeen on siirrytty ylläpitovaiheeseen, joka on Scrum-prosessin mukainen prosessi määrätyillä lisäyksillä. Projektin aloitus- ja lopetusvaihetta käsitellään tarkemmin luvussa 4.3.



Kuva 10. Ylläpitovaiheen Scrum-prosessimalli korkealla tasolla.

Tuotetta kehitetään inkrementaalisesti, pyrähdys kerrallaan Scrum viitekehyksen mukaisesti. Kuten aiemmin todettua, tuotteen arkkitehtuuri ei koe enää suuria muutoksia, sillä se on projektin alkuvaiheessa (alkuprojektin pyrähdyksissä) jo suurissa määrin kiinnitetty. Sen sijaan lisäykset ovat pääosin pieniä lisäominaisuuksia tai kokonaisuuksia, joita tehdään nykyisen rungon päälle.

Seuraavassa prosessi on esitetty alla yhden käyttäjätarinan näkökulmasta tarinan synnystä sen toteutukseen. Myös toimintoihin osallistuvia rooleja on kuvattu alla.

- Tarina syntyy **vaatimuksesta**. Vaatimuksen esittää tuoteomistaja, joka tämän prosessin tapauksessa on useimmiten asiakkaan organisaatiosta. Tarinan validointi alkaa jo tässä vaiheessa: onko tarina mielekäs? Mihin käyttäjätarpeeseen se vastaa? Vaatimuksen yhteydessä noudatetaan riskienhallintaprosessia. Jo tässä vaiheessa voi olla nähtävissä, mihin ohjelman toimintoihin vaatimus vaikuttaa ja mikäli näihin kohdistuu riskiä uuden tarinan myötä. Riskin suuruus arvioidaan ja toteutukseen suunnitellaan muutos, jos riski tai jäännösriski arvioidaan liian suuriksi.
- Scrum-prosessin mukaisessa **suunnittelussa** tarina pilkotaan pienemmiksi tehtäviksi. Pilkkomisen suorittaa tiimi. Riskienhallintaprosessi koskee alaluvussa 4.2.2 kuvatulla tavalla myös määrittelyvaihetta. Määrittelyvaiheessa määritellään viimeistään hyväksymiskriteerit tarinalle sekä määritellään hyväksymistestit. Lisäksi tarinalle määritellään integraatiotestit ennen toteutusta.
- **Toteutusvaiheessa** tiimi toteuttaa tarinan. Tarinan toteutukseen voi osallistua useampikin henkilö tehtävien määrän mukaan. Kaikki koodi vertaiskatselmoidaan laadun parantamiseksi. Tarinan tekijä tekee yksikkötestit toteuttamalleen tehtävälle toteutusvaiheessa. Tehty koodi integroidaan jatkuvasti siihen tarkoitetuilla työkaluilla. Koodi tulee jatkuvasti regressiotestattua sekä tarkasteltua tyylivirhei-

den varalta automaattisilla työkaluilla. Tärkeä asia toteutusvaiheessa on niin kutsuttu valmiin määritelmä. Kyseisessä prosessissa esimerkiksi kaikkien yksikkötestien sekä hyväksymistestien täytyy olla mennyt virheettä läpi, jotta tehtyä tarinaa voidaan pitää valmiina.

- Varsinainen **testausvaihe** hieman hämärtyy Scrum-prosessissa, sillä sitä suoritetaan kaiken aikaa. Kyseisessä ylläpitovaiheen prosessimallissa kuitenkin tarinat hyväksymistestataan niiden teon jälkeen. Hyväksymistestit ovat automaattisia ja jäävät tulevaisuuden käyttöön regressiotestaukseen, jolloin niitä ajetaan joka yö. Integraatiotestit ajetaan pyrähdyksien päätteeksi, kun pyrähdyksen tuotoksia integroidaan erilliselle palvelimelle testejä varten ja josta ne jatkossa liikkuvat eteenpäin testipalvelimelle sekä tuotantoon.
- **Käyttöönotto** on viimeisin vaihe, jonka tarina kokee. Tarina on siihen mennessä otettu käyttöön testiympäristössä, jossa sille on suoritettu vielä lisätestejä mahdollisimman realistisissa olosuhteissa. Käyttöönotolle tuotantoon on oma tarkasti määritetty prosessinsa. Prosessiin kuuluu muun muassa määrättyjen tarinoiden hyväksyminen käyttöönotettavaksi ja lopuksi tuotantoon asentaminen, joka noudattaa myöskin tarkkaa prosessia. Jatkuvan integraation työkaluilla jokainen tuotantoon vietävä versio on automaattitestattu sekä automaattisesti buildattu. Prosessien automatisointi vähentää ihmisistä johtuvien virheiden määrää.

Jokainen tarina käy läpi kuvatut testauskäytänteet ja käytössä on jatkuvan integraation työkalut. Toimituksia tuotantoon tehdään mahdollisimman usein, jotta asiakas saa käyttöönsä mahdollisimman paljon arvoa tuottavia ominaisuuksia ja aina mahdollisimman aikaisin. Jäljitysvaatimus toteutuu osin manuaalisesti ja osin automaattisesti. Projektissa on käytössä JIRA. Vaatimusmäärittely kirjoitetaan aina toteutettavien tarinoiden yhteyteen. JIRA pitää tallessa projektin historiatiedon ja linkittää tarinoita versioihin ja versiot julkaisuajankohtiin. Koodikatselmointeja varten on käytössä oma automaattinen työkalu, joka tallettaa historiatietoja tehdyistä kooditalletuksista ja niiden katselmoijista. Lopulta jokainen talletettu koodinpätkä voidaan jäljittää takaisin vaatimukseen ja vaatimuksesta voidaan tarkastella kaikkia siihen liittyviä koodimuutoksia.

4.2 Prosessimallin keskeisimmät osat

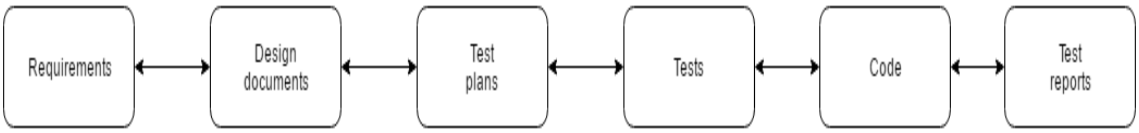
Seuraavissa alaluvuissa on eriteltyinä muutama tärkeä prosessin osa, joka lääketieteellisessä ohjelmistokehityksessä täytyy ottaa huomioon. Nämä liittyvät lääketieteellisen ohjelmistokehityksen hyviin käytäntöihin ja näitä toteutetaan nykyisellään ylläpitovaiheessa olevan projektin prosessimallissa. Esille nostetut asiat ovat keskeisimpiä lisäyksiä, joita Scrum prosessiin täytyy lisätä tai joihin tarvitsee kiinnittää erityistä huomiota lääketieteellistä ohjelmistokehitystä silmällä pitäen. Dokumentointia ei ole erikseen nostettu omaksi kohdakseen alaluvuissa, mutta on syytä huomioida, että dokumentointi on olen-

nainen osa prosessimallia säädellyillä aloilla. Näin ollen alalukujen prosessien noudattamisesta tulee jäädä dokumentoitu jälki. Vaadittuja dokumentteja on eroteltu myöhemmin luvussa.

4.2.1 Lääketieteellinen jäljitettävyysvaatimus

Ketterissä prosesseissa ei ole valmiiksi määriteltynä vaatimusta jäljitettävyydestä. Vaatimus jäljitettävyydestä tarkoittaa sitä, että uudet ominaisuudet, niiden vaatimukset, niiden toteutus ja niiden testaus tulisi olla jäljitettävissä, jotta voidaan todentaa, miksi asioita on tehty ja toteutettu siten kuin on [47]. Jäljitettävyys itsessään käsittää ymmärryksen suunnitelmasta ja sen pyrkimys on osaltaan parantaa kehitettävän tuotteen laatua.

Käytännössä jäljitettävyys tarkoittaa sitä, että eri dokumenttien ja prosessien tulosten seurauksena täytyy jäädä jälki tuotoksista ja niiden täytyy olla linkattuina toisiinsa [49]. Tätä havainnollistetaan kuvassa 11. Kuvasta käy ilmi, kuinka eri vaiheiden tulisi olla jäljitettävissä toisiinsa. Vaatimuksien tuloksena syntyy suunnitelma. Suunnitelman pohjalta syntyy toteutus ja tälle testaussuunnitelma, testit ja testiraportit.



Kuva 11. Jäljitettävyys vaatimuksista suunnitteluun, testeihin ja toteutukseen sekä takaisin.

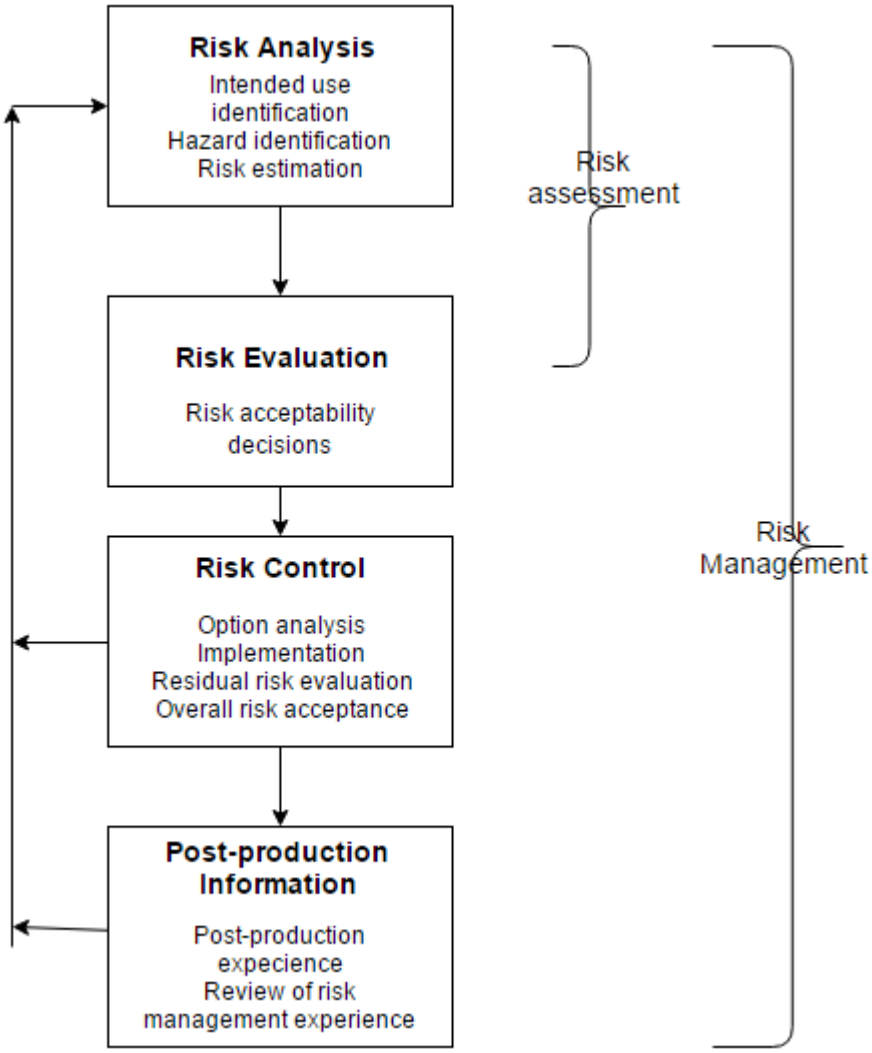
Kuvasta on lisäksi huomioitava, että jäljitettävyys toimii molempiin suuntiin. Vaatimuksista voidaan jäljittää koodiin ja testitapauksiin. Lisäksi koodista ja testitapauksista voidaan jäljittää takautuvasti vaatimuksiin.

4.2.2 Riskienhallinta ketterässä prosessissa

Lääketieteelliset standardit ovat riskilähtöisiä. Ne määrittelevät toimenpiteitä ja prosesseja, jotta lääketieteellisten laitteiden tai ohjelmien käyttöön liittyviä riskejä voidaan havaita, arvioida ja pienentää. Riskilähtöinen ajattelutapa perustuu puolestaan potilasturvallisuuteen. Lääkinnällisen ohjelman kehityksessä ongelmallisimpia tilanteita potilasturvallisuuden kannalta on suunnittelu sekä muutosten hallinta. Suurin osa ohjelmien virheistä johtuvat näiden kahden toiminnon puutteista [26]. Kuten on aikaisemmin mainittu, ei ketterissä ohjelmistokehitysprosesseissa ole määritelty ohjelmistotuotteen toimintoihin liittyvää riskienhallintaa erikseen osaksi ketterää prosessia. Standardit kuitenkin velvoittavat riskienhallinnan osaksi ohjelmistokehitysprosessia kaikissa lääkinnällisen laitteen riskiluokissa. Niinpä riskienhallinnallinen prosessi täytyy määritellä ja liittää osaksi ketterää menetelmää kuten Scrumia.

Riskejä voidaan jakaa projektiriskeihin sekä prosessi-ja tuoteriskeihin. Projektiriskeihin kuuluvat kustannukset, aika ja projektin tuotos. Kirjallisuudessa esitetään väitteitä siitä, että ketterissä prosesseissa ei tarvita erikseen riskienhallintaa, mikäli prosessia noudatetaan tiukasti [13]. Ketterien menetelmien voidaankin katsoa vaikuttavan projektiriskeihin lähestymistavallaan ja väitteen voidaan tulkita koskevan nimenomaan budjetti- ja aikatauluriskejä. Eritoten ei-säädellyillä aloilla ja kevyemmissä projekteissa yksin Scrumin noudattaminen voi riittää projektiriskien hallitsemiseen, sillä riskit usein kohdistuvat suorasti tai epäsuorasti muutoksien hallintaan. Muutoksenhallinta on se ydinalue, johon ketterät menetelmät pyrkivät prosesseillaan vaikuttamaan. Kuitenkin myös ei-säädellyiltä aloilta on saatu tutkimustuloksia siitä, että riskienhallinnalla voidaan saavuttaa hyötyjä esimerkiksi Scrum-prosessissa. Säädellyillä aloilla riskienhallinta on usein erityinen vaatimus, sillä perinteisten projektiriskien lisäksi täytyy hallita ohjelmistotuotteen turvallisuuteen ja toimintaan liittyviä riskejä.

Riskienhallinta on prosessi, joka jatkuu koko lääkinnällisen ohjelmistotuotteen elinkaaren ajan. ISO 14971 mukainen riskienhallintaprosessi on esitetty kuvassa 12 [32]. Riskienhallintaprosessi alkaa analyysistä. Analyysissä tuotteen aiottuun käyttötarkoitukseen pohjaten tunnistetaan riskejä ja vikatilanteita. Evaluointivaiheessa arvioidaan, ovatko tunnistetut riskit hyväksyttäviä. Kontrollointivaiheessa tarkastellaan vaihtoehtoja ja toteutetaan mahdollisesti uusia toimintoja riskien vaikutusten pienentämiseksi sekä arvioidaan jäännösriski. Prosessi on jatkuva ja pyörii syklissä läpi koko tuotteen elinkaaren. Riskejä hallitaan yksittäisen ominaisuuden tasolta lähtien.



Kuva 12. ISO 14971 riskienhallintaprosessi.

Käytännön työkaluja riskienhallintaprosessiin on esimerkiksi niin kutsuttu ”Failure Mode Effect Analysis” (*suomeksi vikatilanteiden vaikutusten analyysi, myöhemmin FMEA*) sekä ”Fault Tree Analysis” (*suomeksi vikapuu-analyysi, myöhemmin FTA*) [49]. Myös muita työkaluja on, mutta riskienhallintaa toteutetaan työn prosessimalleissa näiden työkalujen avulla. FMEA on työkalu, jonka avulla analysoidaan tuotteen mahdollisia vikatilanteita [3]. Virhetilanteista, eli riskeistä, arvioidaan kuinka todennäköisiä ne ovat ja kuinka vakavia niiden seuraukset on. Esimerkki FMEA-matriisi on esitetty kuvassa 13.

Failure Mode	Effects	Cause	Severity	Probability	Risk	Suggested Action	Severity	Probability	Risk
Risk 1	Patient safety at risk	Incorrect user input	2	2	4	Verify inputs	2	1	2
Risk 2	Discomfort caused	Incorrect data shown due to specification error	1	2	2	Additional feature to be implamented	1	1	1

Kuva 13. Esimerkki FMEA-matriisi.

Matriisista käy ilmi riski, sen syy, vaikutukset, vakavuus alkuperäisellä tasolla, todennäköisyys alkuperäisellä tasolla ja lopulta riski alkuperäisellä tasolla. Riskienhallintaprosessin mukaisesti riskiä pienennetään jatkotoimenpiteillä ja lopuksi arvioidaan jäännösriski.

Riskin todennäköisyyden ja seurauksen perusteella saadaan riskille määritettyä riskiluokka. Esimerkki tällaisesta riskimatriisista on kuvassa 14. Punainen väri tarkoittaa riskiä, jota ei voida hyväksyä. Keltainen väri kuvaa sellaista riskiä, jonka suhteen riskinvähennystoimenpiteitä tulisi kartoittaa. Vihreä väri kuvaa vähäistä riskiä.

		Seuraukset				
		S1	S2	S3	S4	S5
Todennäköisyys	T5					
	T4					
	T3					
	T2					
	T1					

Kuva 14. Riskienluokittelumatriisi.

Mikäli riski on hyväksymättömällä tasolla, täytyy tehdä korjaavia toimenpiteitä ja pienentää riskiä. Korjaavat toimenpiteet voivat suunnitteluvaiheessa olla esimerkiksi arkkitehtuuriset muutokset tai toteutusvaiheessa lisäominaisuudet, joilla riskiä rajataan. Toimenpiteiden jälkeen arvioidaan jäännösriski ja menettelyä toistetaan niin kauan, että riski on hyväksyttävällä tasolla.

4.2.3 Testauskäytäntöjä ketterään lääketieteelliseen prosessiin

Testaaminen on kriittinen osa mitä tahansa ohjelmistokehitysprosessia. Kun puhutaan potilasturvallisuudesta ja lääketieteellisistä ohjelmistokehitysprojekteista, testaamisen merkitys korostuu entisestään [42]. Perinteisissä vesiputousmallin ohjelmistoprojekteissa tes-

taaminen ajoittuu ja kasaantuu projektin loppupuolelle. Tällainen lähestymistapa on kuitenkin vaarallinen niin tuoteriskien kuin projektiriskienkin puolesta. Sitä mukaa, kun tuotteen ominaisuuksia jätetään kunnolla testaamatta, kehittyy niin kutsuttua teknistä velkaa. Teknisen velan kasvaessa projektin loppuvaiheessa paljastuvien vikojen määrän ennustaminen vaikeutuu jatkuvasti. Tästä seuraa puolestaan epävarmuustekijöitä projektin aikataululle ja budjetille; syntyy niin kutsuttuja projektiriskejä, joita ketterissä projekteissa ei vastaavalla tavoin syntyisi. Ketterissä projekteissa uusia ominaisuuksia testataan sitä mukaa, kun niitä kehitetään ja integroidaan osaksi valmista tuotetta. Säädellyillä aloilla tuotteen testaamisen on syytä olla erityisen huolellista, sillä tällä on mahdollista rajoittaa tuoteriskiä.

Erään määrittelyn mukaan ohjelman laatuominaisuudet voidaan jakaa neljään ryhmään [42]:

- **Luotettavuus** tarkoittaa sitä, että ohjelma suoriutuu sille määritetyistä tehtävistä luotettavasti ilman kaatuilua tai jumiutumista.
- **Toiminnallisuus** tarkoittaa sitä, että ohjelma suorittaa sille määritetyt liiketoimintavaatimukset oikein.
- **Sovelluksen suorituskyky** tarkoittaa sitä, että ohjelma vastaa ja suorittaa annetut operaatiot sujuvasti.
- **Systeemin suorituskyky** tarkoittaa sitä, että laite jatkaa määriteltyä suorittamista, vaikka se joutuisi kuormituksen alaiseksi.

Lääkinnällisille laitteille erityisen tärkeää on ohjelman luotettavuus. Viat tällä osa-alueella aiheuttavat taatusti epämukavuutta ja potentiaalisesti taloudellisia tappioita tai potilasturvallisuuden vakavan vaarantumisen.

Lääketieteellisessä ohjelmistokehityksessä tärkeitä asioita ovat niin ikään **verifiointi** ja **validointi**. FDA:n määritelmän mukaan verifiointi tarjoaa objektiivisen todisteen, että suunnittelun tuotos täyttää kaikki ohjelmiston elinkaarimallin vaatimukset kyseisessä elinkaaren vaiheessa [57]. Validoinnilla tarkoitetaan sitä, että ohjelman määrittely noudattaa aiottua käyttötarkoitusta sekä vastaa käyttäjien tarpeisiin ja että tietyt vaatimukset täytetään läpi koko ohjelman elinkaaren ajan. Ohjelman testauksella pyritään vaikuttamaan näihin molempiin tärkeisiin toimintoihin, verifiointiin ja validointiin.

Alla on eritelty muutamia erilaisia testauskäytänteitä, joita lääketieteellisessä ohjelmistokehityksessä voidaan käyttää. Kevyemmissä ja toiminnoiltaan ei-kriittisissä ohjelmistoprojekteissa näitä kaikkia ei välttämättä tehdä ja käytännössä joskus voi olla myös niin, ettei asiakas halua maksaa testaamisesta. Lääketieteellisen ohjelmistokehityksen täytyy sen sijaan olla testaukseltaan todella kattavaa, mikä tekee prosessista myös raskaamman.

- **Yksikkötestauksella** tarkoitetaan yksittäisen, pienen osan testaamista koodista [52]. Osa voi olla esimerkiksi yksittäinen metodi, josta käydään läpi eri vaihtoehdot, joita metodissa voi tapahtua annetuilla syötteillä. Yksikkötestit ovat nopeita tehdä ja ajaa sekä hyvä työkalu myös regressiotestaamiseen, kun koodi muuttuu. Yksikkötestauksen suorittaa itse kehittäjä, joka tietää, kuinka hänen tekemänsä metodin tulee toimia eri tilanteissa.
- **Hyväksyntätestaus** testaa yksittäistä ominaisuutta hyväksymiskriteerejä vasten [53]. Hyväksymistestiin määritellään vaaditut ulostulot tarinan määrittelyn yhteydessä, jo ennen toteutusta. Tarinan toteuttamisen jälkeen tarina testataan määriteltyjä ulostuloja vasten. Hyväksyntätestissä on tärkeää validoida, että määritelty ulostulo on tarinan määrittelyn mukainen, jotta testi testaa oikeita asioita.
- **Integraatiotestauksella** tarkoitetaan pyrähdyksien päätteeksi suoritettavaa testausta, jossa testataan kuinka tehdyt käyttäjätarinat toimivat osana kokonaista tuotetta [52]. Integraatiotestit suunnitellaan ennen tarinan toteutusta ja ne ajetaan tarinan valmistumisen jälkeen. Integraatiotestauksessa voidaan hyödyntää esimerkiksi eri ympäristöä, jossa testi tehdään. Tällä saadaan testattua kokonaisuuden robustisuutta muualla kuin kehitysympäristössä.
- **Kuormitustestauksessa** järjestelmä asetetaan kovan kuormituksen alaiseksi ja testataan sen suoriutumista ja luotettavuutta tällaisissa tilanteissa [52]. Tällainen voi olla tärkeää sovelluksesta riippuen. Kuormitustestausta voi ohjata määrittelyssä asetetut rajat ja vaatimukset yhtäaikaiselle käytölle tai yhtäaikaisille operaatioille.

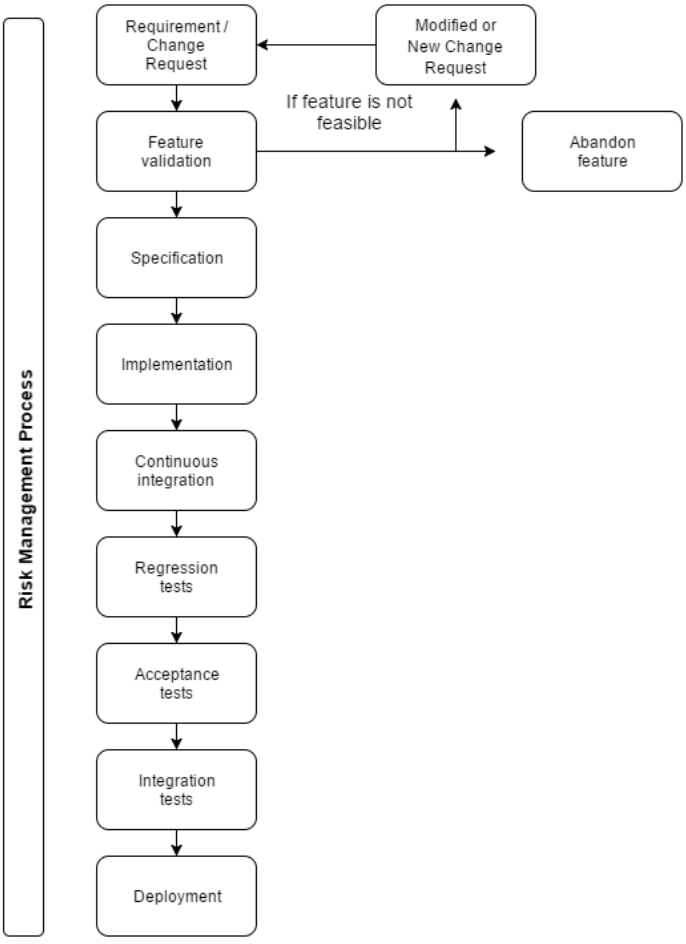
Testaus on olennainen osa lääketieteellistä ohjelmistokehitystä. Sen avulla parannetaan ohjelman laatua ja luotettavuutta, jotka johtavat suoraan parempaan potilasturvallisuuteen. Eri testausmenetelmillä on mahdollista ottaa kiinni useamman tyyppisiä virheitä ohjelmassa.

4.2.4 Jatkuva integraatio

Jatkuva integraatio on käytäntö, jossa koodia talletetaan tiimin yhteiseen koodirepositorioon mahdollisimman usein ja jokainen uusi ohjelman versio käännetään ja testataan automaattisesti jatkuvaan integraatioon soveltuvalla palvelimella [54]. Jatkuva integraatio on käytössä nykyisin todella laajalti ja siihen tarkoitettuja työkaluja on useita: Jenkins, MagnumCI ja TeamCity esimerkkeinä. Jatkuvan integraation edut ovat mittavat ja osin jopa itsestään selvät. Regressiotestaus voidaan automatisoida jatkuvalla integraatiolla, sillä jokaisen version yhteydessä ajetaan versiolle automaattitestit. Ongelmien ja bugien löytäminen ja jäljittäminen on näin paljon helpompaa sekä samalla kustannustehokkaampaa. Samalla vältetään isojen integroitavien palasten manuaalinen integroiminen projektiin. Kaikki tämä nostaa laatua ja pienentää kustannuksia. Jatkuva integraatio on lisäksi halpaa, sillä hyviä avoimen lähdekoodin työkaluja on saatavilla ja tulee jatkuvasti lisää.

4.2.5 Muutostenhallinta

Seuraavassa luvussa esitellyn julkaisusuunnitteluvaiheen vuoksi prosessista tulee ”vesiputousmaisempi” kuin normaalista Scrum-prosessista. Ideaalitilanteessa tuotteen vaatimuksista tiedetään ennalta hieman enemmän kuin täysin ketterässä projektissa. Mikäli näin ei kuitenkaan ole ja muutoksia tulee kesken julkaisuiden, on tärkeää huolehtia muutoksenhallintaprosessista ja säilyttää projektin ketteryys tuotteen laadun ja paremman asiakasarvon takaamiseksi. Varsinkin projektin muutostenhallintaprosessissa korostuvat lääketieteelliset erityisvaatimukset. Muutoksien tekeminen aikaisempaan ohjelman toimintaan on helppo tapa tuottaa ohjelmakoodiin virheitä ja arvaamattomia toimintoja. Muun muassa tähän seikkaan lääketieteellisillä laatuvaatimuksilla pyritään vaikuttamaan ja kasvattamaan potilasturvallisuutta. Muutosten hallintaan käytetään apuna vaatimusten validointia, regressiotestausta, jatkuvaa integraatiota, riskienhallintaprosessia sekä hyväksyntätestausta. Kun muutospyyntö on käynyt läpi kaikki nämä vaiheet onnistuneesti, on se tuotantokelpoinen. Muutostenhallintaprosessia on kuvattu kuvassa 15. Muutostenhallintaprosessi noudattelee ylätasolla Purojärven [43] esittelemää muutostenhallintaprosessia. Kuvaan 15 on kuitenkin kuvattu yksityiskohtaisemmalla tasolla prosessimallin toimenpiteitä, jotka ovat olennaisia muutostenhallinnalle.



Kuva 15. Muutoksenhallintaprosessi

Muutos etenee kuvassa 15 esitetyllä tavalla kohti valmista. Muutos tai uusi ominaisuus voidaan kuitenkin hylätä mikäli se todetaan huonoksi. Optimitilanteessa tämä tehdään ajoissa, mutta myös myöhemmin voidaan todeta, ettei vaatimus sovellu tuotteen käyttötarkoitukseen, joten vaatimus voidaan hylätä myös myöhemmin. Riskienhallintaprosessi on oleellisena osana läsnä koko muutoksenhallintaprosessin ajan.

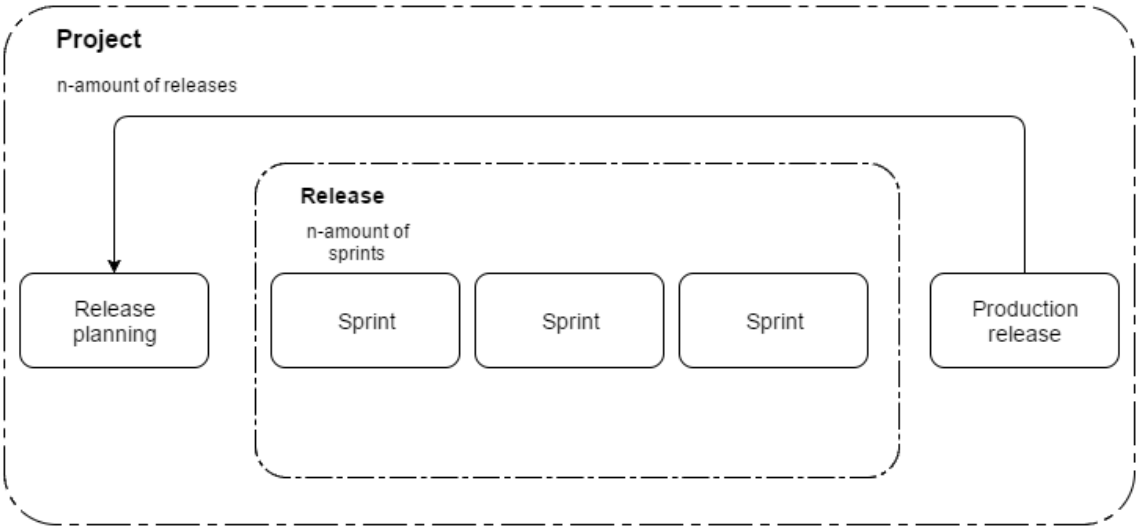
Muutos toteutetaan sen kiireellisyydestä riippuen seuraavissa pyrähdyksissä. Tuotantoon vienti tapahtuu kyseisen projektin prosessien mukaisesti. Muutoksenhallintaprosessilla voisi kuvata myös normaalien tarinoiden reittiä suunnittelusta tuotantoon. Kun näitä prosesseja on käynnissä useita samanaikaisesti, päällekkäin ja limittäin ja kun itsenäisesti ohjautuvan tiimin jäsenet kukin työstävät tarinoiden eri vaiheita, puhutaan Scrum-prosessin mukaisesta ketterästä ohjelmistokehityksestä. Muutosta toteutettaessa ”muun tuotteen kehitys” etenee rinnalla samanaikaisesti eikä muutoksenhallinta lisää prosessiin ylimääräistä kuormaa, sillä normaali tuotekehityskin noudattaa samanlaista prosessia. Tällöin on kyse ketteryydestä, vaikka julkaisut olisivatkin suunniteltu aikaisemmin jo pääpiirteiltään tietynlaisiksi: muutokset ovat mahdollisia ja niitä voidaan tehdä ongelmitta, kuitenkin noudattaen potilasturvallisuuden takaavaa prosessia.

4.3 Lisäykset prosessimalliin

Prosessimalli sisältää nykyisellään lääketieteellisiin vaatimuksiin vastaavia prosessin osia, joita on eritelty aikaisemmissa alaluvuissa. Päivitetty prosessimalli pohjaa kirjallisuudesta tehtyihin löydöksiin. Prosessimalli on nykyisellään varsin ketterä ja hyvin toimiva. Sitä on kuitenkin mahdollista tarkentaa rakenteellisuuden avulla sekä käydä läpi sen oleellisimmat kohdat puutosten tai tarpeettomien osien varalta. Koska kyseessä on ylläpitovaiheen prosessimalli, ei etenkään projektin alkupuolta ole otettu siinä erityisen hyvin huomioon. Projektin alkuvaiheen vuoksi prosessimalliin on lisätty julkaisusuunnitteluvaihe. Kun projekti on jo tuotantovaiheessa voidaan soveltaa ylläpitovaiheen prosessimallia sellaisenaan tuotteen jatkokehitykseen.

4.3.1 Julkaisusuunnitteluvaihe

Kirjallisuudessa prosessimalliin on esitetty lisättäväksi julkaisusuunnitteluvaihe (*engl. release planning*) [4]. Julkaisusuunnitteluvaiheen voidaan katsoa tuovan lisää rakenteellisuutta ja ennakoitavuutta Scrum-prosessiin. Luvussa 3 esitellyissä aikaisemmissa tutkimuksissa rakenteellisuuden lisääminen oli myös useilla tahoilla esille noussut asia pohdittaessa prosessimallia ketterään lääketieteelliseen ohjelmistokehitykseen. Rakenteellisuuden avulla voidaan Scrum-prosessista tehdä hiukan formaalimpi ja sisällyttää siihen lääketieteellisten vaatimusten mukaisia toimintoja säilyttäen kuitenkin yhä prosessin ketteryys. Prosessimalli on kuvattu korkealla tasolla kuvassa 16.



Kuva 16. Julkaisusuunnitteluvaiheen sisältävä prosessimalli.

Projekti sisältää mielivaltaisen määrän julkaisuja. Projekti etenee siten, että julkaisuja suunnitellaan muutama julkaisu eteenpäin ja tämän jälkeen julkaisuja toteutetaan pyrähdyksissä. Pyrähdyksien määrää per julkaisu ei ole rajattu. On lisäksi huomioitava, että Scrum-mallin mukainen ohjelmistokehitys mahdollistaa ketteryyden, joten julkaisuun voidaan tehdä ketterästi muutoksia sen toteutusvaiheessa. Toteutusvaihetta seuraa tuotantoon julkaisu. Testaaminen, riskienhallinta ja jäljitettävyys toteutetaan edellisessä luvussa kuvatulla tavalla. Ne ovat yhä oleellinen osa yksityiskohtaista prosessia, jonka avulla varmistetaan tuotteen laatu lääketieteellisessä kontekstissa. Kuvassa 16 ei ole kuvattu projektin perustamisvaihetta eikä projektin päätösvaihetta.

Julkaisusuunnittelulla päästään vaikuttamaan etenkin ohjelmistotuotteen arkkitehtuuriin projektin aikaisessa vaiheessa eri tavalla kuin Scrum-ideologiassa. Scrum-ideologiassa arkkitehtuuria rakennetaan ensimmäisissä sprinteissä ja sitäkin on tarkoitus iteratiivisesti kehittää. Lääketieteellisessä ohjelmistokehityksessä vaaditaan arkkitehtuuridokumentti ja eritoten tämän vuoksi rakenteellisemman prosessimallin käyttäminen on hyödyllistä. Arkkitehtuurin suunnitteluun on varattu prosessissa aikaa ja sen tärkeimmät osat saadaan lukittua aikaisessa vaiheessa projektia. Julkaisusuunnittelu vaikuttaa myös arkkitehtuuriin julkaisujen tasolla. Koska näkyvyys on parempi olettaen, ettei julkaisuun tehdä liian suuria muutoksia, voidaan arkkitehtuurin kannalta tuotteeseen tulevat lisäominaisuudet tai parannukset suunnitella perinteistä Scrum-mallia huolellisemmin.

Etenkin projektin aloituksessa dokumentoitavaa on enemmän kuin varsinaisen kehitysprosessin aikana [4]. Aiottu käyttötarkoitus, arkkitehtuuridokumentti sekä muu projektidokumentaatio voidaan tehdä ensimmäisen julkaisusuunnitelman kohdalla prosessissa. Tämä on myös paikka määrittää ja automatisoida projektin käytänteet jäljitettävyyden ylläpitämiseksi. Ensimmäisen julkaisusuunnitelman aikana syntyy lisäksi vaatimusmäärittelyä, joka tuotetaan tuotteen kehitysjonon muotoon, joskin tämä tarkentuu ja muuttuu

projektin aikana. Tulevat julkaisusuunnittelukohdat prosessissa tarjoavat luonnollisen kohdan suunnitella ja dokumentoida projektia jo ennalta, tulevaa julkaisua silmälläpitäen, vaikka dokumentointia tehtäisiinkin myös pyrähdyksien aikana.

Koska standardointi ja säätelevien elimien kriteerit ovat Euroopan ja USA:n markkinoilla hyvin samanlaiset lääketieteellisessä ohjelmistokehityksessä, voidaan todeta, että samanlaisella prosessimallilla voidaan tuottaa lääketieteelliseen käyttöön tarkoitettuja ohjelmia molemmilla markkinoilla. Tuotteen markkinointiluvan saamiseksi hyväksyntäprosessi vaihtelee markkinoiden välillä. Tällä ei kuitenkaan ole vaikutusta siihen, kuinka ohjelmaa tulee kehittää.

4.4 Yhteenveto lääketieteellisten vaatimusten täyttämisestä Euroopan ja Yhdysvaltojen markkinoilla

IEC 62304 on harmonisoitu standardi Euroopan Unionissa ja Yhdysvalloissa [17]. Standardi kattaa direktiivin 93/42/EEC ja sen lisäyksen M5 (2207/47/EC) vaatimukset, joten IEC 62304 standardi on yksinkertaisin reitti MDD vaatimusten täyttämiseen ohjelmiston osalta. Myös Yhdysvaltojen FDA katsoo, että IEC 62304 standardi kattaa lääketieteelliselle ohjelmistokehitykselle asetetut vaatimukset. Standardia voidaan käyttää siis mittarina vaadittujen lääketieteellisten vaatimusten täyttämiseen ohjelmistojen osalta sekä USA:n että Euroopan markkinoilla. Standardi ei kuitenkaan koskaan ole vaatimus itsessään hyväksyttävän ohjelman kehittämiseen [35]. Koska viranomaiset kuten FDA ja Euroopan komissio tunnustavat kuitenkin määrätyn joukon standardeja, niin näistä standardeista on muodostunut hyväksyttävä perusjoukko säädökset täyttäviä toimintatapoja lääketieteelliselle ohjelmistokehitykselle.

Muita keskeisiä standardeja ovat ISO 13485, ISO 14971, EN 60601 sekä IEC 62366. IEC 62304 standardin lisäksi yritysten täytyy täyttää näiden standardien vaatimukset sekä Yhdysvaltojen että Euroopan markkinoilla, kuitenkin tapauskohtaisesti [28]. Standardeista ISO 13485 on yrityksen kokonaisvaltaisen laadunjohtamisen standardi ja ISO 14971 riskienhallintastandardi, jota laajennetaan ohjelmiston riskien osalta vielä standardissa IEC 62304. Standardit ISO 13485 sekä ISO 14971 ovat korkeamman tason standardeja verrattuna IEC 62304 standardiin. IEC 62304 keskittyy vain ohjelmiston osuuteen prosessien näkökulmasta. EN 60601 on sähkökäyttöisten terveydenhuollon laitteiden turvallisuusstandardi, jossa käsitellään lisäksi muun muassa verkkolaitteita [5]. Kyseistä standardia tarvitsee noudattaa vain näiden piiriin kuuluvien laitteiden tapauksessa, joten tämä rajaa osan ohjelmistoista standardin ulkopuolelle. Aikaisemmin kuitenkin kaikki viittaukset ohjelmistoon oli löydettävissä kyseisestä standardista, kun ohjelmistoja koskevia standardeja ei ollut olemassa. IEC 62366 standardi asettaa vaatimuksia lääkinnällisen laitteen ergonomialle ja käyttäjä-interaktiolle [36]. Osa ohjelmistoista rajautuu selkeästi myös tämän standardin ulkopuolelle.

On lisäksi huomioitava, että riskiluokituksen I ohjelmaa valmistavan yrityksen ei tarvitse toteuttaa standardin ISO 13485 organisaatiotason laadunhallintajärjestelmää lääketieteellisille laitteille [8]. Euroopan ja USA:n markkinoilla noudatettavia standardeja on koottu yhteen taulukossa 2. FDA käyttää riskiasteikkonaan I-III. Euroopan komissio on jakanut tason II tasoihin IIa sekä IIb. Yksinkertaistuksen vuoksi taulukossa 2 on taso II esitetty FDA-mallin mukaisesti.

Taulukko 2. EU:n ja USA:n markkinoilla käytettävät standardit.

	EU			USA		
	I	II	III	I	II	III
ISO 13845		x	x		x	x
ISO 14971:2007				x	x	x
ISO 14971:2012	x	x	x	x	x	x
IEC 62304	x	x	x	x	x	x
EN 60601	x	x	x	x	x	x
IEC 62366	x	x	x	x	x	x

Kuten taulukosta 2 käy ilmi, on lääketieteellinen ohjelmistokehitys viranomaisvaatimuksien puolesta hyvin samankaltaista Euroopassa ja USA:ssa. Lopuksi on huomioitava, että kaikkia standardeja ei tarvitse sertifioida voidakseen valmistaa lääkinnällisiä laitteita. Riittää, että valmistaja noudattaa standardien vaatimuksia ja pystyy osoittamaan sen.

4.5 Ratkaisun arviointi

Lääketieteellisestä ohjelmistokehitysprosessista tunnistettiin tärkeitä osa-alueita, joita ovat muun muassa jäljitettävyys, riskienhallinta, validointi ja verifiointi sekä dokumentointi. Näistä riskienhallinta on syytä erotella erittäin keskeisenä osana, sillä lääketieteellisestä ohjelmistokehityksestä puhuttaessa potilasturvallisuus on keskiössä. Oikein toteutetun riskienhallinnan avulla lisätään ohjelman laatua ja parannetaan potilasturvallisuutta.

Nykyisessä prosessimallissa vaatimukset on otettu huomioon eikä näitä ole syytä muuttaa päivitetyssä prosessimallissa. Projektin alkua on syytä tarkentaa päivitettyyn prosessimalliin, sillä ylläpitomoodissa oleva projekti ei ole projektin aloituksen tarkasteluun enää

otollinen. Tärkeäksi kohdaksi projektin alussa nähdään arkkitehtuuri, johon on puolestaan vaikea enää vaikuttaa ylläpitovaiheen projektissa. Muutostenhallinta nostettiin omaksi aliluvukseen, sillä se on yksi olennaisimmista asioista, missä ketterä projekti voi erottua vesiputousmallisesta projektista. Julkaisusuunnittelun voitaisiin helposti nähdä vievän projektia vesiputousmaisempaan suuntaan, mutta tavoitteena on tähdentää, ettei näin ole. Muutoksia tehdessä voidaan yhä käyttää samaa prosessia ja muutokset ovat yhä sallittuja, vaikka julkaisuja suunnitellaankin. Standardit ovat kirjoitettu vesiputousnäkökulmasta, mutta ne eivät vaadi käytettäväksi tietynlaista prosessimallia. Ketterillä menetelmillä on mahdollista kehittää lääketieteellisiä ohjelmistoja.

Liiketoiminnan näkökulmasta eroja nykyisen ja päivitetyn prosessimallin välillä on vaikea analysoida tarkasti. Molemmat prosessit ovat ketteriä ja hyvin samankaltaisia. Projektien kestojen ja kustannuksien eroista saatiin viitteitä aiemmista tutkimuksista. Näitä väitteitä voidaan perustella muun muassa muutoksen hallinnan hyvällä toteuttamisella sekä usean samanaikaisen prosessin suorittamisella. Näitä kumpaakin toteutetaan esitellyssä projektimallissa. Sen sijaan näiden toteuttaminen on vaikeaa tai perusajatuksen puolesta mahdotonta vesiputousmallin ohjelmistokehityksessä.

Eräs esille nostettava asia on lääketieteellisten laitteiden (ohjelmistojen) luonne ketteryyteen peilattuna. Lääketieteelliset ohjelmistot tarvitsevat usein paljon suunnittelua potilasturvallisuuden vuoksi, vaikka erilaisia tapauksiakin on. Ketteryyden näkökulmasta tämä tarkoittaa kuitenkin sitä, että ketteryyden tuomia liiketoimintahyötyjä voi olla vaikea realisoida esimerkiksi tuotteen ”pilotoinnissa”. Sen sijaan kehitettävän tuotteen suuntaa on helpompi ohjailla ketterillä prosesseilla muuttuvan ympäristön ja muuttuvien vaatimusten suhteen. Tämä puolestaan lisää tuotteen mahdollisuuksia vastata markkinoiden kysyntään ja uusiin vaatimuksiin nopeammilla vasteajoilla.

Yleisesti ottaen verrattuna tavalliseen ohjelmistoliiketoimintaan säädellyt alat ovat kustannuksiltaan raskaampia. Lisätyt prosessit ja tehtävät voidaan nähdä ”ylimääräisinä kustannuksina” säätelemättömiin aloihin verrattuina. Säätelemättömillä aloilla ja ketterillä ohjelmistokehitysprosesseilla voidaankin tuottaa hyvin kustannustehokkaita projekteja, mikä ei vastaavissa määrin ole mahdollista säädellyillä aloilla. Tämä ei kuitenkaan sinänsä ole ongelma jatkuvan lääketieteellisen ohjelmistokehitysliiketoiminnan suhteen, sillä alalla on tällä hetkellä varsin hyvin nostetta, mikä ilmenee muun muassa luvussa 3 esitetyistä Eurooppalaisista patenttihakemuksista. Kysyntää tuotteille ja ohjelmistokehitysosaamiselle on. Markkinoille tunkeutuminen on kuitenkin pienemmille yrityksille työläämpää kuin säätelemätön ohjelmistokehitys vaadittavien prosessien ja tätä kautta alkupääoman suhteen.

Lääketieteellisen ohjelmistokehityksen lisätyt prosessit ja modifioitu ohjelmistokehitysmalli ei ensisilmäyksellä vaikuta kovin työläältä. On kuitenkin tärkeää huomioida, että mikäli tällaista prosessia aletaan jalkauttaa tyhjästä ohjelmistokehitystiimiin, ottaa se oman aikansa saada vaatimuksiin liittyvät käytännöt ja näiden käytännön toteuttaminen

kohdalleen tiimille sopivalla tavalla. Työkaluja lääketieteellisten vaatimusten toteuttamiseen on markkinoilla paljon. Mahdollisimman monen asian automatisointi prosessissa alentaa kustannuksia.

5. YHTEENVETO

Jo 1990-luvun puolella ohjelmistot osana lääketieteellisiä laitteita alkoivat yleistymään. Pian murroksen yhteydessä alkoi käydä ilmi, että ohjelmistoihin täytyy kiinnittää erityistä huomiota lääkinnällisissä laitteissa potilasturvallisuuden vuoksi. Useita ongelma- ja vikatilanteita pystyttiin yhdistämään ohjelmakoodista peräisin olevaan virheeseen. Tämän myötä lääketieteellisille ohjelmistoille alkoi kehittyä erityisvaatimuksia ohjelmien laadun parantamiseksi. Ohjelmakoodissa sijaitsevat virheet voivat olla usein arvaamattomia ja vaikeita ennakoida.

Lääketieteellisten ohjelmien laadun parantamiseksi on kehitetty useita standardeja. On olemassa organisaatiotason standardeja kuten ISO 13485 ja ISO 14971 sekä erityisesti ohjelmaan keskittyviä standardeja kuten IEC 62304. ISO 13485 on organisaation laatujärjestelmän standardi, ISO 14971 on lääketieteellisten laitteiden riskienhallintastandardi ja IEC 62304 on lääketieteellisten ohjelmistojen elinkaaren standardi. Euroopassa lääketieteellisten ohjelmistojen laatua valvoo Euroopan komissio. Jotta laitetta voi myydä Euroopassa, tarvitsee se CE-merkinnän. Yksinkertaisin reitti CE-merkinnän saamiseen on vaadittujen standardien mukainen ohjelmistokehitys. Edellä mainituista standardeista IEC 62304 määrittelee ohjelmiston tuottamiseen vaaditut prosessit, jotta ohjelma kattaa lääketieteellisille laitteille asetetut laatuvaatimukset. Yhdysvalloissa vastaava elin on Food and Drug Administration eli lyhennettynä FDA. Jotta laitetta voi myydä Yhdysvalloissa, tarvitsee se FDA:n hyväksynnän. Hyväksynnän saamiseksi on täytettävä tiettyjä FDA:n asettamia vaatimuksia. Koska IEC 62304 on harmonisoitu standardi Yhdysvaltojen ja Euroopan välillä, niin standardin katsotaan täyttävän FDA:n lääketieteelliselle laitteelle asettamat vaatimukset. Täten kyseinen standardi on yksinkertaisin tie lääkinnällisen ohjelmiston kehittämiselle niin Euroopan kuin Yhdysvaltojenkin markkinoilla.

Noudatettavien standardien sekä standardien sisältämien pykälien määrä riippuu siitä, mille riskiluokitukselle lääkinnällinen laite on asetettu. Lääkinnällisen laitteen riskiluokitukseen sisältyy muun muassa näkökulmat siitä, voiko laite aiheuttaa millaista vahinkoa, jos se vikaantuu. Olennaisena osana tähän liittyy se, siirtyykö laitteen ja potilaan välillä energiaa ja mikä on läpäisyn (*engl. invasiveness*) taso. Euroopassa riskiluokitukset ovat I, IIa, IIb ja III. Korkein riskiluokitus on III ja näin ollen sitä koskevat tiukimmat määräykset.

Ketterät prosessit eroavat vaiheesta toiseen etenevistä ohjelmistokehitysmalleista, kuten vesiputousmalli, niin käytännön kuin liiketoiminta-ajatuksenkin tasolla. Käytännössä ketterät prosessit ovat usein iteratiivisia ja niissä tehdään useita asioita sekä tuotteen vaiheita rinnakkain. Scrum on eräs ketterä prosessimalli, joka soveltuu erityisen hyvin vaikeisiin olosuhteisiin ja muuttuvien vaatimusten projekteihin. Scrumissa tarkoitus on kehittää tuotetta pyrähdyksittäin ja jokaisen pyrähdyksen päätteeksi valmis tuote. Tämä tarkoittaa

sitä, että pyrähdyksessä tehtävät asiat kehitetään vertikaalisti alusta loppuun ja integroidaan osaksi valmista tuotetta. Liiketoiminnan kannalta tämä tarkoittaa fundamentaalista eroa vaiheesta toiseen eteneviin prosesseihin.

Liiketoimintahyötyinä ketterien menetelmien on tarkoitus rajoittaa projektiriskiä ja tuottaa arvoa aikaisemmin. Käytäntöön sidottuna tämä tarkoittaa sitä, että jo aikaisten pyrähdyksien jälkeen on mahdollista käyttää tuotetta, jossa osa ominaisuuksista on valmiina. Lisää arvoa ja ominaisuuksia tuotetaan pala kerrallaan seuraavissa pyrähdyksissä. Tämä rajoittaa projektiriskiä siten, että jo aikaisessa vaiheessa on mahdollista viedä tuotantoon niin kutsuttu ”minimum viable product” eli MVP, jonka avulla saadaan jo aikaisessa vaiheessa palautetta potentiaalisista asiakkaista tai tuotteen käyttäjistä. Vaiheesta toiseen etenevissä projekteissa tuotantoon vienti tapahtuu vasta lopuksi valmiin tuotteen kanssa ja projektiriski säilyy näin korkealla tasolla läpi projektin.

Säädellyillä aloilla kehitetään ohjelmia vielä vallitsevasti vesiputousmallin ohjelmistokehitysprosesseilla. Tämän on arveltu johtuvan siitä, että standardien asettamat määräykset sopivat hyvin vaiheesta toiseen eteneviin prosesseihin: vaiheiden lähtötiedot ja tuotokset on helppo todentaa standardien vaatimalla tavalla. Kuitenkin asiassa on tapahtumassa murros, sillä aikaisemmissa tutkimuksissa on saatu rohkaisevia tuloksia ketterien menetelmien käytöstä myös säädellyillä aloilla. Ei-säädellyillä aloilla on jo aiemmin tiedetty ketterien menetelmien luovan tuottavuus- ja laatuparannuksia prosesseihin ja tuotteeseen. Nyt kuitenkin vesiputousmalliin on joissain lääketieteellisissä projekteissa yhdistetty ketteriä ominaisuuksia ja tuotteita on myös kehitetty kokonaan ketterästi saavuttaen tuntuvia liiketoimintahyötyjä projektin kustannuksissa, kestossa sekä asiakastyytyväisyydessä.

Scrum ei täytä yksin lääketieteellisten standardien asettamia vaatimuksia. Scrum-ohjelmistokehitysprosessiin tarvitsee tehdä lisäyksiä. Lääketieteelliseen ohjelmistokehitykseen kuuluu muun muassa jäljitettävyys-, verifiointi-, ja riskienhallintavaatimuksia, jotka täytyy ottaa huomioon ketterässä Scrum-prosessissa lisäämällä sen kehitys- ja määrittelyprosesseihin vaiheita. Lisävaatimukset pyrkivät kaikki nostamaan tuotteen laatua sekä potilasturvallisuutta.

Riskienhallinta lääkinnällisten laitteiden tapauksessa tarkoittaa potilaaseen liittyvää riskiä laitteen osalta. Ohjelman mahdollisesti tuottama riski kuuluu arvioida ominaisuuden määrittelyvaiheessa. Jos riski arvioidaan liian suureksi tai sitä on muutoin mahdollista pienentää sen huomioon ottavalla suunnittelulla, tulee riskiä pienentää hyväksyttävälle tasolle. Arvioinnin seurauksena saattaa suunnittelusta löytyä puutteita, joita on mahdollista paikata määriteltävillä lisäominaisuuksilla. Erilaisilla testausmenetelmillä kuten yksikkötestaus, hyväksyntätestaus, integraatiotestaus, kuormitustestaus, vähennetään ohjelman vikoja ja rajataan virheellisen toteutuksen osalta riskiä. Riskienhallintaprosessi on yksi IEC 62304 standardin määrittelemistä prosesseista, joka kulkee tuotteen valmistuksen rinnalla koko tuotteen elinkaaren ajan.

Vastoin aiempaa vallitsevaa suuntausta, lääketieteelliseen käyttöön tarkoitettuja ohjelmia on mahdollista kehittää ketterillä menetelmillä. Vesiputousmalli noudattelee standardien asettamia vaatimuksia prosessien lähtötietojen ja tuotoksien osalta, mutta ketterää mallia, kuten Scrum, muokkaamalla voidaan valmistaa laadukkaita, lääketieteelliseen käyttöön suunniteltuja ohjelmia, joiden elinkaareen on mahdollista sisällyttää ketterien menetelmien liiketoimintahyödyt.

LÄHTEET

[1] A. Adomavicius, Five steps for agile transformation in financial services, 2016. Viitattu 28.12.2016. Saatavissa: https://www.devbridge.com/articles/agile-transformation-financial-services/

[2] Agile Alliance, Manifesto for Agile Software Development, 2001. Saatavissa: https://www.agilealliance.org/agile101/the-agile-manifesto/

[3] ASQ, Failure mode effects analysis. Viitattu 28.12.2016. Saatavissa: http://asq.org/learn-about-quality/process-analysis-tools/overview/fmea.html

[4] S. Baggström, SW development process in regulated health tech area, 2014. Viitattu 15.11.2016. Saatavissa: http://www.slideshare.net/StefanBaggstrm/medical-sw-development-process-lecture-material-case-movendos-oy

[5] CUI Inc, IEC 60601-1 Medical Design Standards - 3rd Edition, 2013. Viitattu 15.9.2016. Saatavissa: http://www.cui.com/catalog/resource/iec-60601-1-medical-design-standards.pdf

[6] J. Cunningham, B. Dolan, D. Kelly, C. Young, Medical Device Sectoral Overview, 2015. Viitattu 28.12.2016. Saatavissa: http://galwaydashboard.ie/publications/medical-sector.pdf

[7] N. Donaldson, Managing software development risk with agile. Viitattu 28.12.2016. Saatavissa: https://leanpub.com/managing-risk-with-agile/read

[8] Emergo Group, Europe CE Marking Regulatory Process for Medical Devices, 2013. Viitattu 28.12.2016. Saatavissa: https://www.emergogroup.com/resources/europe-process-chart

[9] European Commission, CE marking, 2016. Viitattu 7.9.2016. Saatavissa: http://ec.europa.eu/growth/single-market/ce-marking/

[10] European Commission, Medical devices: Guidance document - Classification of medical devices, 2010. Saatavissa: http://ec.europa.eu/consumers/sectors/medical-devices/files/meddev/2_4_1_rev_9_classification_en.pdf

[11] European Patent Office, European Patent Applications. Viitattu 28.12.2016. Saatavissa: http://www.epo.org/about-us/annual-reports-statistics/annual-report/2015/statistics/patent-applications.html#tab3

[12] B. Gloger, 3 reasons for introducing agile product development in medical technology. Viitattu 28.12.2016. Saatavissa: https://borisgloger.com/wp-content/uploads/2014/07/Whitepaper-MedTec-english.pdf?0190af

[13] B. Gold, C. Vassell, Using risk management to balance agile methods: A study of the Scrum process, 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), pp. 49-54.

[14] V. Hegde, Case study — Risk management for medical devices (based on ISO 14971), Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual, pp. 1-6.

[15] N. Hrgarek, Certification and regulatory challenges in medical device software development, Software Engineering in Health Care (SEHC), 2012 4th International Workshop on, pp. 40-43.

[16] Ichanical Technologies, Agile Software Development: Everything you need to know. Viitattu 28.12.2016. Saatavissa: http://www.ichanical.com/agile-software-development-everything-you-need-to-know/

[17] International Medical Device Regulators Forum, Statement regarding Use of IEC 62304:2006 ”Medical device software -- Software life cycle processes”, 2015. Saatavissa: http://www.imdrf.org/docs/imdrf/final/procedural/imdrf-proc-151002-medical-device-software-n35.pdf

[18] International Organization for Standardization, ISO 13485 Standardi, 2016.

[19] ISO Online Browsing Platform, IEC 62304:2006(en), 2006. Saatavissa: https://www.iso.org/obp/ui/#iso:std:iec:62304:ed-1:v1:en

[20] Jon Speer, The Definitive Guide to ISO 14971 Risk Management for Medical Devices, 2015. Viitattu 15.9.2016. Saatavissa: http://blog.greenlight.guru/iso-14971-risk-management

[21] Jon Speer, Why FMEA is NOT ISO 14971 Risk Management, 2016. Viitattu 15.9.2016. Saatavissa: http://blog.greenlight.guru/fmea-is-not-iso-14971-risk-management

[22] C. Larman, V. R. Basili, Iterative and incremental developments. a brief history, Computer, Vol. 36, No. 6, 2003, pp. 47-56.

[23] K. Larsson, 5 quick questions on IEC 82304, 2015. Viitattu 28.12.2016. Saatavissa: http://www.aligned.ch/blog/53-tips-tricks/232-5-quick-questions-on-iec-82304

[24] W. Lin, X. Fan, Software Development Practice for FDA-Compliant Medical Devices, Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on, pp. 388-390.

[25] Patrik Malmquist, The Project Triangle Turned Upside Down, 2011. Viitattu 1.11.2016. Saatavissa: http://www.applitude.se/2011/02/the-project-triangle-turned-upside-down/

[26] F. McCaffery, J. Burton, I. Richardson, Improving software Risk Management in a Medical Device Company, 2009 31st International Conference on Software Engineering - Companion Volume, pp. 152-162.

[27] M. Mc Hugh, O. Cawley, F. McCaffcry, I. Richardson, X. Wang, An agile V-model for medical device software development to overcome the challenges with plan-driven software development lifecycles, Software Engineering in Health Care (SEHC), 2013 5th International Workshop on, pp. 12-19.

[28] M. Mchugh, F. Mccaffery, V. Casey, Software process improvement to assist medical device software development organisations to comply with the amendments to the medical device directive, IET Software, Vol. 6, No. 5, 2012, pp. 431-437.

[29] Medical Device and Diagnostic Industry, Developing Medical Device Software to IEC 62304, 2010. Viitattu 13.9.2016. Saatavissa: http://www.mddionline.com/article/developing-medical-device-software-iec-62304

[30] Medical Device and Diagnostic Industry, Simplifying IEC 62304 Compliance for Developers, 2011. Viitattu 13.9.2016. Saatavissa: http://www.mddionline.com/article/simplifying-iec-62304-compliance-developers

[31] Medical Device Certification GmbH, Basic Information about the European Directive 93/42/EEC on Medical Devices, 2009. Viitattu 17.1.2016. Saatavissa: https://www.mdc-ce.de/fileadmin/user_upload/Downloads/mdc-Dokumente/Broschueren/040100_basic_info_93-42-EEC_06_e.pdf

[32] Meditec, Innovative Risk Management for Medical Devices. Viitattu 28.12.2016. Saatavissa: http://www.meditec.hia.rwth-aachen.de/en/research/former-projects/innorisk/

[33] C. Michaud, How to qualify, classify and CE mark software, 2015. Viitattu 11.9.2016. Saatavissa: http://blog.cm-dm.com/pages/How-to-qualify%2C-classify-and-CE-mark-software

[34] C. Michaud, IEC 82304, 2016. Viitattu 28.12.2016. Saatavissa: http://blog.cm-dm.com/post/2016/11/01/IEC-82304-1%3A2016-Health-software-Part-1%3A-General-requirements-for-product-safety

[35] C. Michaud, IEC 82304-1 - latest news about the standard on Health software, 2016. Viitattu 28.12.2016. Saatavissa: http://blog.cm-

dm.com/post/2016/01/15/IEC-82304-1-latest-news-about-the-standard-on-Health-Software

[36] C. Michaud, ISO and IEC standards for software in medical devices in a nutshell, 2011. Viitattu 15.9.2016. Saatavissa: http://blog.cm-dm.com/post/2011/11/01/ISO-and-IEC-standards-explained-to-software-engineers-and-quality-managers

[37] Minnetronix, Understanding the Importance of IEC 62304, 2016. Viitattu 13.9.2016. Saatavissa: http://www.minnetronix.com/resources/news/understanding-importance-of-IEC-62304.html

[38] Mountain Goat Software, New to Agile and Scrum. Viitattu 28.12.2016. Saatavissa: https://www.mountaingoatsoftware.com/agile/new-to-agile-or-scrum

[39] Mountain Goat Software, Scrum, 2016. Viitattu 8.9.2016. Saatavissa: https://www.mountaingoatsoftware.com/agile/scrum

[40] Mountain Goat Software, ScrumMaster, 2016. Viitattu 8.9.2016. Saatavissa: https://www.mountaingoatsoftware.com/agile/scrum/scrummaster

[41] Mountain Goat Software, Scrum Images. Viitattu 28.12.2016. Saatavissa: https://www.mountaingoatsoftware.com/agile/scrum/images

[42] V. Podgorelec, M. Hericko, M. B. Juric, I. Rozman, Testing reliability of medical software, Proceedings of 15th IEEE Symposium on Computer-Based Medical Systems (CBMS 2002), pp. 185-190.

[43] J. Purojärvi, Lääkinnällisen laitteen ohjelmistokehitys täydennetyllä Scrum-mallilla, 2010. Viitattu 17.1.2016. Saatavissa: https://jyx.jyu.fi/dspace/handle/123456789/25544

[44] P. A. Rottier, V. Rodrigues, Agile Development in a Medical Device Company, Agile, 2008. AGILE '08. Conference, pp. 218-223.

[45] Praxiom Research Group Limited, ISO 13485 Translated into Plain English. Viitattu 28.12.2016. Saatavissa: http://www.praxiom.com/iso-13485.htm

[46] R. Rasmussen, T. Hughes, J. R. Jenks, J. Skach, Adopting Agile in an FDA Regulated Environment, Agile Conference, 2009. AGILE '09. pp. 151-155.

[47] G. Regan, F. McCaffery, K. McDaid, D. Flood, Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model, 2013 Computer Standards & Interfaces volume 31, Issue 1, pp. 3-9.

[48] RTC Magazine, Understanding IEC 62304 for Medical Device Compliance, 2015. Viitattu 13.9.2016. Saatavissa: http://www.rtcmagazine.com/articles/view/115994

[49] L. Salminen, EU ja CE-merkki, 2013. Viitattu 28.12.2016. Saatavissa: https://www2.uef.fi/documents/976466/1745345/06-19Salminen+EU+CE/945a3d50-9925-4aac-977a-8546cdb44450

[50] Software process improvement and roadmapping - a roadmap for implementing IEC 62304 in organizations developing and maintaining medical device software, Springer Verlag, 2015, 19-30 p.

[51] Testing Excellence, Types of Software Testing - Complete list. Viitattu 28.12.2016. Saatavissa: http://www.testingexcellence.com/types-of-software-testing-complete-list/

[52] Testing Excellence, What is Acceptance Testing in Agile. Viitattu 28.12.2016. Saatavissa: http://www.testingexcellence.com/acceptance-testing-agile/

[53] Thoughtworks, Continuous Integration. Viitattu 28.12.2016. Saatavissa: https://www.thoughtworks.com/continuous-integration

[54] K. Trektere, F. McCaffery, M. Lepmets, Development of a software process framework to assist organizations developing mobile medical apps, 2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), pp. 461-464.

[55] Turvallisuus- ja kemikaalivirasto, CE-merkki, 2015. Viitattu 7.9.2016. Saatavissa: http://www.tukes.fi/fi/Toimialat/Kuluttajaturvallisuus/Kulutustavarat/CE-merkki/

[56] U.S Food & Drug Administration, General Principles of Software validation; Final Guidance for Industry and FDA Staff, 2002. Viitattu 28.12.2016. Saatavissa: http://www.fda.gov/RegulatoryInformation/Guidances/ucm085281.htm

[57] U.S Food & Drug Administration, Medical Device Exemptions 510(k) and GMP Requirements, 2016. Viitattu 28.12.2016. Saatavissa: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpcd/315.cfm

[58] U.S Food & Drug Administration, Overview of Device Regulation, 2015. Viitattu 28.12.2016. Saatavissa: http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Overview/

[59] K. Wong, C. Callaghan, Managing requirements baselines for medical device software development, Systems Conference (SysCon), 2012 IEEE International, pp. 1-5.