Saara Lindgren

Vaatimusmäärittely ja testauksen suunnittelu potilasmonitorin ohjelmiston uudelle toiminnallisuudelle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

15.5.2024

Tiivistelmä

Tekijä:	Saara Lindgren
Otsikko:	Vaatimusmäärittely ja testauksen suunnittelu potilasmonitorin ohjelmiston uudelle toiminnallisuudelle
Sivumäärä:	33 sivua
Aika:	15.5.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Hyvinvointi- ja terveysteknologia
Ohjaajat:	Lehtori Sakari Lukkarinen Lead Automation Engineer Tahiti Konttinen

---

GE HealthCare on suuri lääkinnällisiä laitteita, kuten potilasmonitoreita, valmistava yritys. Potilasmonitoreita käytetään sairaaloissa monilla eri osastoilla monitoroimaan potilaan elintoimintoja. Potilasmonitori ja siihen kytketyt lisälaitteet muodostavat yhdessä potilasmonitorijärjestelmän.

Eri markkina-alueet säätelevät lääkinnällisten laitteiden valmistusta asettaen vaatimuksia, jotka laitevalmistajan on toteutettava. Vaatimuksia ovat muun muassa tiukat riskien- ja laadunhallinnan vaatimukset. Lisäksi laitteen valmistuksen eri vaiheiden tulee olla jäljitettäviä.

Lääkinnällisten laitteiden ohjelmistokehitys vastaa tavallista ohjelmistokehitystä. Erona on tarvittavan dokumentaation määrä lääkinnällisiä laitteita koskevien jäljitettävyysvaatimuksien takia. Ohjelmistokehityksen vaiheiksi voidaan lukea mallintaminen, vaatimusmäärittely, suunnittelu, toteutus, testaus ja käyttöönotto. Se, miten nämä vaiheet toteutetaan, riippuu paljon siitä, millainen ohjelmistokehitysprojekti on kyseessä.

Tässä insinöörityössä tehtiin yrityksen potilasmonitoreihin kehitettävälle uudelle toiminnallisuudelle ohjelmistotason vaatimusmäärittely. Vaatimusmäärittelyprosessi jaettiin kolmeen osaan, jotka olivat toiminnallisuuden hahmottaminen, vaatimuksien kirjoittaminen ja vaatimusmäärittelyn hyväksyttäminen. Työn aikana valmistunut vaatimusmäärittely ei ollut lopullinen versio vaatimusmäärittelystä.

Vaatimusmäärittelyn lisäksi insinöörityössä suunniteltiin, miten testata uuden toiminnallisuuden järjestelmätason vaatimukset. Suunnittelussa jokainen vaatimus tutkittiin yksitellen. Lopputuloksena oli, että suurin osa vaatimuksista voitaisiin testata automaatiotestauksella.

Avainsanat: potilasmonitori, ohjelmistokehitys, vaatimusmäärittely, ohjelmistotestaus

---

Abstract

Author:	Saara Lindgren
Title:	Requirement Specifications and Testing for a New Software Functionality of Patient Monitor
Number of Pages:	33 pages
Date:	15 May 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Health Technology
Supervisors:	Sakari Lukkarinen, Senior Lecturer Tahiti Konttinen, Lead Automation Engineer

---

GE HealthCare specializes in designing and manufacturing medical devices. Among their products are patient monitors. Patient monitors are used in hospitals in many different care areas to monitor patients' vital signs. A patient monitor and the additional devices connected to the monitor form a patient monitoring system.

Several regulations and requirements are defined for medical devices. The manufacturer of a medical device needs to meet these requirements, including strict risk and quality management requirements. Each step in the manufacturing process of the device needs to be traceable.

Software development for medical devices is very similar to ordinary software development. The difference is the amount of documentation needed for medical devices because of the traceability requirements. The phases of software development are modeling, requirement analysis, design, implementation, testing, and deployment.

The purpose of this study was to create software requirement specifications for a new functionality being developed for patient monitors. The process of software requirements specifications was divided into three parts: understanding the functionality, writing the specifications, and approval. The software requirement specifications created were not the final version of the specifications.

In addition, a plan was made for testing the system requirements created for the new functionality. The requirements were investigated one by one. The result was that most of the requirements could be tested using automation.

Keywords: patient monitor, software development, requirement specifications, software testing

Sisällys

Lyhenteet ja käsitteet

DD: *Design Detail*. Ohjelmiston ei-toiminnallinen vaatimus, jolla voidaan esimerkiksi määritellä käyttöliittymän ulkonäköä.

DOORS: *IBM Rational DOORS, Dynamic Object-Oriented Requirements System*. IBM:n vaatimustenhallintatyökalu, joka mahdollistaa myös manuaalitestien kirjoittamisen.

EEG: Elektroenkefalografia eli aivosähkökäyrä kuvaa aivojen sähköistä toimintaa graafisesti.

EKG: Elektrokardiogrammi eli sydänsähkökäyrä kuvaa sydämen sähköisiä impulsseja graafisesti.

FDA: *Food and Drug Administration*. Yhdysvaltain elintarvike- ja lääkevirasto.

Gherkin: Tyyli tai ohjelmointikieli, jolla voidaan esimerkiksi kirjoittaa helppolukuisia vaatimuksia ja automaatiotestejä ”given-when-then”-rakenteella.

Happisaturaatio:
Veren happipitoisuus.

MDR: *Medical Device Regulation*. Euroopan unionin asetus lääkinnällisistä laitteista 2017/745.

Noninvasiivinen verenpaine:
Kehon ulkopuolelta mitattu verenpaine esimerkiksi olkavarsimittarin avulla.

Product Owner:
Tuoteomistaja eli ketterässä ohjelmistokehityksessä tiimin

vastuuhenkilö, jonka tehtävänä on varmistaa tiimin kehitystyön edistyminen.

RUP: *Rational Unified Process*. Yhtenäistetty ohjelmistokehityksen prosessi, iteratiivinen ohjelmistokehitysmalli.

User story: Käyttäjätarina eli ketterässä ohjelmistokehityksessä oleva tapa kuvata toteutettavaa ominaisuutta loppukäyttäjän tai asiakkaan näkökulmasta.

Verifikaatio: Ohjelmiston vaatimuksien todentaminen.

1 Johdanto

Lääkinnällisten laitteiden tuominen markkinoille vaatii niiden valmistajilta paljon työtä, jotta markkinoille päätyy ainoastaan laitteita, joiden käyttö ei vaaranna potilasturvallisuutta. Valmistajan tulee täyttää tiukat regulaatiot ja muut lainsäädännön vaatimukset, jotta valmistajan lääkinnällistä laitetta voidaan käyttää ja myydä. Esimerkiksi Euroopan unionin alueella noudatettava lainsäädäntö on MDR-asetus (Medical Device Regulation) ja Yhdysvalloissa lainsäädännön vaatimukset lääkinnällisille laitteille asettaa elintarvike- ja lääkevirasto FDA (Food and Drug Administration).

Tämä insinöörityö tehtiin GE HealthCare -yritykselle, joka valmistaa useita erilaisia lääkinnällisiä laitteita. Näitä laitteita ovat muun muassa erilaiset kuvantamislaitteet ja potilasmonitorit. Yrityksen toimipisteessä Helsingin Vallilassa valmistetaan potilasmonitoreita ja kehitetään niiden ohjelmistoa.

Tyypillisesti ohjelmistokehityksen alussa täytyy tehdä vaatimusmäärittely, jossa määritellään, miten ohjelmiston tai yhden ohjelmiston toiminnallisuuden tulisi toimia. Ennen tätä vaihetta voidaan myös tehdä mallinnus, joka voi sisältää esimerkiksi erilaisia esitutkimuksia. Seuraavassa vaiheessa suunnitellaan käyttöliittymä. Näiden vaiheiden ollessa valmiita voidaan ohjelmistoa alkaa toteuttamaan. Ohjelmiston ollessa valmis voidaan suorittaa testaus, jonka tarkoituksena on varmistaa, että ohjelmisto täyttää sille laaditut vaatimukset.

Insinöörityössä laadittiin potilasmonitoreihin suunnitellulle uudelle toiminnallisuudelle ohjelmistotason vaatimusmäärittely. Vaatimusmäärittelyn laatiminen sisälsi itse vaatimuksien kirjoittamisen ja niiden luokittelun esimerkiksi eri laitteiden mukaan. Lisäksi kirjoitetut vaatimukset tuli hyväksyttää yrityksen sisällä. Vaatimusmäärittelyssä keskityttiin ainoastaan ohjelmistotason vaatimuksiin, sillä yritys oli jo kirjoittanut ylemmän tason vaatimukset uudelle toiminnallisuudelle. Insinöörityön aikana valmistunut ohjelmistotason vaatimusmäärittely ei ollut lopullinen versio uuden toiminnallisuuden vaatimusmäärittelystä.

Vaatimusmäärittelyn lisäksi insinöörityön tarkoituksena oli selvittää yritykselle paras tapa testata vaatimusmäärittelyssä tulleita vaatimuksia. Mahdolliset testausmenetelmät olivat manuaali- ja automaatiotestaus. Vaatimuksien testausmenetelmän selvittämiseen käytettiin apuna muun muassa olemassa olevia ohjelmistotestejä. Tavoitteena oli, että mahdollisimman moneen voitaisiin hyödyntää automaatiotestausta. Valmiit suositukset vaatimuksien testausmenetelmistä välitettiin eteenpäin yrityksen sisällä. Insinöörityössä ei kirjoitettu testejä eikä kokeiltu automaatiotestauksen toimivuutta.

2 Tausta

2.1 GE HealthCare

GE HealthCare on suuri yhdysvaltalainen yritys, jolla on toimintaa yli 160 maassa. Yrityksen juuret sijoittuvat 1890-luvulle, jolloin kolme sähköalan yhtiötä muodostivat yhden yrityksen nimeltä General Electric eli GE. Yritys kasvoi suureksi, ja se valmisti muun muassa lentokonemoottoreita sekä työskenteli uusiutuvan energian parissa. Vuonna 2023 terveysteknologiaan keskittyvä osio viimeisteli erkaantumisena GE-konsernista ja aloitti itsenäisenä yrityksenä nimeltä GE HealthCare. [1; 2.] Yritys valmistaa pääosin sairaalaympäristöissä käytettäviä laitteita. Näitä laitteita ovat muun muassa

- röntgenlaitteet
- magneettikuvantamislaitteet
- tietokonetomografialaitteet
- ultraäänilaitteet
- potilasmonitorit
- hengityskoneet.

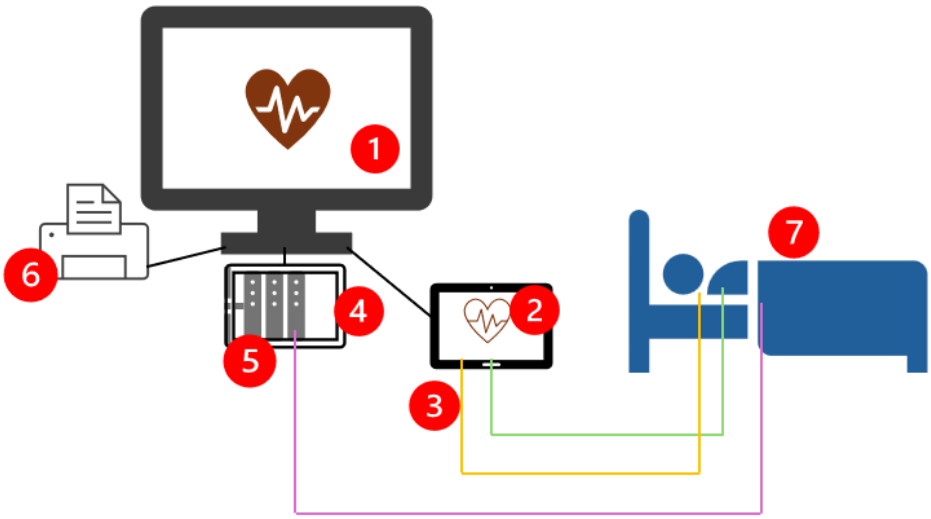
Vuonna 2023 koko GE HealthCaren liikevaihto oli 19,6 miljardia dollaria, josta potilasmonitoreita hallinnoivan Patient Care Solutions -yksikön osuus oli noin 3,1 miljardia dollaria. Nettotuottoa yrityksellä oli koko vuoden aikana 1,6 miljardia dollaria. [3.]

Suomessa yrityksen toiminta keskittyy pääosin potilasmonitorien valmistamiseen ja kehittämiseen. GE HealthCare päätyi Suomeen vuonna 2003, kun he ostivat suomalaisen Instrumentarium Oyj -yrityksen [4]. Instrumentarium Oyj toimi laajasti lääkinnällisten laitteiden parissa, mutta yrityksen tunnetuimpia laitteita olivat anestesialaitteet ja potilasvalvontaan tarkoitetut laitteet. GE HealthCaren toimipaikkoja Suomessa ovat Helsinki ja Kuopio. Potilasmonitorien tuotantoa on Suomessa vain Helsingin toimipisteessä.

2.2 Potilasmonitorit

2.2.1 Potilasmonitorijärjestelmä

Potilasmonitori on lääkinnällinen laite, jolla voidaan monitoroida potilaan elintoimintoja reaaliaikaisesti. Elintoimintojen monitoroinnin lisäksi potilasmonitorit voivat tehdä hälytyksiä, jos potilaan elintoimintojen arvot poikkeavat monitorille määritellyistä hälytysrajoista. Yksinään potilasmonitori ei kuitenkaan kykene näyttämään potilaan tilaa, vaan sen tueksi tarvitaan paljon lisälaitteita. Potilasmonitori ja lisälaitteet muodostavat yhdessä potilasmonitorijärjestelmän. Kuvaan 1 on mallinnettu esimerkki tällaisesta järjestelmästä.



Kuva 1. Potilasmonitorijärjestelmä [5].

Kuvassa 1 on esitetty esimerkki mahdollisesta potilasmonitorijärjestelmästä, joka pohjautuu GE HealthCaren CARESCAPE-potilasmonitorisarjaan. Kuvassa 1 olevat numerot tarkoittavat:

1 = potilasmonitori
2 = parametrilaite tai -moduuli
3 = parametrikaapeli
4 = F2- tai F5-moduulirunko
5 = E-moduuli
6 = tulostin
7 = potilas.

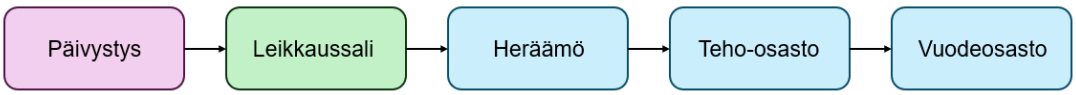
Potilasmonitori (1) voi koostua joko pelkästä monitorista, johon ei kuulu näyttöä, tai monitorista, johon on upotettu näyttö. Potilasmonitorit, joissa ei ole upotettua näyttöä, tarvitsevat erillisen näytön, jotta voidaan nähdä monitoroitavia arvoja. Jotta potilasmonitorin näytöltä voidaan nähdä potilaasta tulevaa dataa, tarvitaan potilasmonitoriin kytkettävä parametrilaite tai -moduuli (2). Näihin laitteisiin kytketään parametri- tai potilaskaapeleita (3) halutuista fysiologisista parametreistä. Parametri- ja potilaskaapelin erona on se, että parametrikaapeli osaa käsitellä sen keräämää fysiologista dataa. Kun käytössä on pelkkä potilaskaapeli, tarvitaan ennen potilasmonitoria väliin moduuli käsittelemään dataa. Kaapeleiden anturit kytketään kiinni potilaaseen (7), jolloin data kulkeutuu näkyviin potilasmonitorin näytölle.

Parametrilaitteet ja -moduulit eivät mahdollista kaikkien fysiologisten parametrien monitorointia. Niihin on mahdollista kytkeä yleisimpiä parametrejä, kuten EKG, happisaturaatio ja noninvasiivinen verenpaine. Jos potilaalta tarvitsee monitoroida esimerkiksi hengityskaasuja tai EEG:tä eli aivosähkökäyrää, tarvitaan näiden parametrien mittaamiseen erillisiä E-moduuleja (5). E-moduulit hoitavat potilaalta kerätyn datan käsittelyn. E-moduulit kytketään potilasmonitorijärjestelmään F2- tai F5-moduulirungoilla (4). Rungon nimessä oleva numero kertoo, kuinka monta E-moduulia runkoon mahtuu. E-moduulit täytyy myös kytkeä kaapeleilla kiinni potilaaseen. Joissakin tapauksissa potilasmonitorijärjestelmään

voidaan kytkeä myös tulostin (6). Tulostimella voidaan tarvittaessa tulostaa esimerkiksi EKG-käyrän poikkeavuuksia, jotka aiheuttavat monitorille hälytyksiä.

2.2.2 Potilasmonitorien käyttöympäristö

Potilasmonitorin pääasiallinen käyttöympäristö on sairaala. Sairaalan sisältä löytyy useita osastoja ja tiloja, joissa tarvitaan potilasmonitoreja. Näitä ovat esimerkiksi vuodeosastot, leikkaussalit, teho-osastot ja päivystys tai ensiapu. Sairaalaan saapunut potilas saattaa hoitonsa ohessa kulkea usean eri osaston läpi. Leikkaushoitoa tarvitseva potilas muodostaa perioperatiivisen polun kulkiessaan sairaalan eri osastojen läpi. Perioperatiivinen polku voidaan jakaa kolmeen vaiheeseen, joita ovat preoperatiivinen, intraoperatiivinen ja postoperatiivinen vaihe. Preoperatiivinen vaihe tapahtuu ennen leikkausta, intraoperatiivinen vaihe on leikkauksen aikana ja postoperatiivinen vaihe kattaa leikkauksen jälkeisen toiminnan. [6.] Kuvassa 2 esitetään yksinkertaistettu perioperatiivinen polku.



Kuva 2. Esimerkki potilaan perioperatiivisesta polusta sairaalassa [6].

Kuvassa 2 on esitetty esimerkki potilaan perioperatiivisesta polusta alkaa preoperatiivisesta vaiheesta, jossa potilas saapuu päivystykseen. Päivystyksessä potilaan todetaan tarvitsevan välittömästi leikkaushoitoa, jolloin potilas siirretään leikkaussaliin. Tällöin alkaa intraoperatiivinen vaihe. Leikkauksen jälkeen siirrytään postoperatiiviseen vaiheeseen, jossa potilas siirretään heräämöön. Potilaan tila voi kuitenkin romahtaa, ja tällöin hänet jouduttaisiin siirtämään sairaalan teho-osastolle. Potilas toipuu teho-osastolla niin hyvin, että hänet voidaan siirtää tavalliselle vuodeosastolle odottamaan kotiutusta.

Kuvan 2 perioperatiivisen polun jokaisella osastolla tarvitaan potilasmonitoreita, jotta potilaan tilaa voidaan seurata ja mahdollisiin muutoksiin pystytään reagoimaan. Kun potilasta joudutaan siirtämään eri paikkoihin sairaalan sisällä, kulkee parametrilaite tai -moduuli monesti koko ajan potilaan mukana. Parametrilaite tai -moduuli voidaan kytkeä uudessa paikassa löytyvään potilasmonitoriin ilman, että potilaaseen kiinnitettyjä kaapeleita joudutaan irrottamaan. Kuvan 2 perioperatiivisessa polussa ei ole otettu huomioon mahdollisia poikkeuksia tai ongelmatilanteita, joiden takia potilas ei kävisi kaikkia osastoja kuvan 2 mukaisessa järjestyksessä läpi.

Perioperatiivisessa polussa kuvatut sairaalan osastot vaativat potilasmonitorilta erilaisia monitoroitavia fysiologisia parametrejä. Päivystyksessä, jossa potilaat vaihtuvat nopeaan tahtiin, tarvitaan parametrejä, jotka ovat helposti mitattavissa kuten happisaturaatiota. Päivystyksessä tulee olla valmius muihinkin parametreihin, kuten sydänsähkökäyrään, mutta kiireinen ympäristö ei aina mahdollista EKG-elektrodien kiinnittämistä tarkasti potilaaseen. Mikäli on tärkeää tietää potilaan pulssi, voidaan tämä tieto saada mittaamalla happisaturaatiota. Teho-osastolla potilaat ovat tyypillisesti sellaisessa tilassa, että pienetkin parametrien muutokset voivat merkitä hengenvaaraa, joten potilasmonitoria tarvitaan monen eri parametrin mittaamiseen. [7.] Mitä enemmän parametrejä käytetään, sitä kattavampi kuva potilaan tilasta saadaan.

3 Lääkinnällisen laitteen ohjelmistokehitys

3.1 Lainsäädännön vaatimukset

Lääkinnällisellä laitteella tarkoitetaan esimerkiksi laitetta, välinettä, implanttia tai muuta tarviketta, jonka tarkoituksena on sairauden tai vamman diagnosointi, lieventäminen, hoitaminen tai ehkäisy. Lisäksi käyttötarkoituksia voivat olla jonkin toiminnon tai anatomisen osan korvaaminen tai muuntaminen. Myös ohjelmistot voivat olla lääkinnällisiä laitteita, jos ne täyttävät määritelmän. Lääkinnällisen laitteen toimintaan vaikuttava ohjelmiston lasketaan olevan osa lääkinnällistä laitetta, joten potilasmonitorin tapauksessa ohjelmistoa ei tarvitse tarkastella

erikseen fyysisestä laitteesta. Lääkinnällistä laitetta käytetään ihmisellä. Käyttötarkoituksen määrittely on laitteen valmistajan vastuulla. [8; 9.]

Käyttötarkoituksen lisäksi valmistajan tulee luokitella lääkinnällinen laite riskiluokkaan. Riskiluokka perustuu laitteen käyttötarkoitukseen ja mahdolliseen riskiin, joka käyttäjälle tai potilaalle voi aiheutua. Käyttäjään tai potilaaseen kohdistuvan riskin suuruuteen voivat vaikuttaa muun muassa laitteen käytön kesto, käyttöalue ja invasiivisuus. [8.] Riskiluokkien määrä vaihtelee eri markkina-alueiden välillä, mutta ne ovat keskenään yhteneväisiä. Esimerkiksi Yhdysvaltojen FDA määrittelee lääkinnällisille laitteille kolme riskiluokkaa, jotka ovat I, II ja III [10]. EU-alueella MDR-asetuksessa on määritelty lääkinnällisille laitteille neljä eri luokkaa, jotka ovat I, IIa, IIb ja III [8]. Molemmissa esimerkeissä riskiluokka I on pienin mahdollinen riskiluokka ja III suurin. Lääkinnällisen laitteen riskiluokka vaikuttaa siihen, millaisia toimenpiteitä valmistajalta vaaditaan, jotta laitetta voidaan myydä eri markkina-alueilla. Nämä toimenpiteet voivat liittyä ennen markkinoille tuomista tai siihen, kun lääkinnällinen laite on jo markkinoilla. [8; 10.]

Kun tarkastellaan potilasmonitoria, luokitellaan se EU-alueella luokkaan IIb ja Yhdysvalloissa luokkaan II. Tämä tarkoittaa laitteen kuuluvan kohtalaisen riskin luokkaan. Potilasmonitorin riskiluokka perustuu siihen, että laitteella monitoroidaan potilaan keskeisiä elintoimintoja, millä voi olla suuri merkitys potilaan saamaan hoitoon ja terveyteen ylipäänsä. FDA vaatii useimmilta luokan II lääkinnällisiltä laitteilta 510k-prosessia, joka laitteen valmistajan tulee tehdä ennen laitteen markkinoille saattamista, kun vastaavia laitteita on jo markkinoilla [10]. Kyseinen prosessi sisältää hyvin paljon samoja osa-alueilta, joita MDR-asetuksen teknisissä asiakirjoissa on vaadittu laitteen valmistajilta. Näitä ovat muun muassa laitteen ja sen käytön tarkka määrittely, muiden vastaavien laitteiden vertailu, vaatimusmäärittely sekä testausasetelma tuloksineen. [8; 11.] Valmistajan on toimitettava mainittuja tietoja, jotta voidaan arvioida laitteen olevan turvallinen käyttää.

3.2 Ohjelmistokehityksen mallit

Ohjelmistokehitykseen liittyy paljon erilaisia vaiheita ennen kuin ohjelmiston voidaan arvioida olevan valmis. Lääkinnällisten laitteiden ohjelmistokehitys ei juurikaan eroa tavallisesta ohjelmistokehityksestä, jossa ei tehdä lääkinnällisiä laitteita tai osia niistä. Lääkinnällisten laitteiden valmistukseen liittyy enemmän regulaatioita ja muita lainsäädännön vaatimuksia. Tämä näkyy tarvittavan dokumentaation määrässä verrattuna ohjelmistoihin, jotka eivät ole lääkinnällisiä laitteita tai osia niistä. Tarvittavassa dokumentaatiossa korostuvat tiukat laatu- ja riskienhallinnan vaatimukset. Lisäksi verifikaation ja jäljitettävyyden merkitys on suuri. Näiden avulla voidaan todeta ohjelmiston toimivan niin kuin valmistaja on määritellyt. Dokumentaation laatiminen on laitteen valmistajan vastuulla.

Ohjelmistokehitystä voidaan kuvata useiden erilaisten kehitysmallien avulla. Yhteistä ohjelmistokehitysmalleille ovat mallien sisältämät vaiheet. Ohjelmistokehityksen vaiheiksi voidaan lukea mukaan mallintaminen, vaatimusmäärittely, suunnittelu, toteutus, testaus ja käyttöönotto. Mallintamisvaiheessa voidaan tehdä esitutkimusta ohjelmistoon liittyen, josta muodostuvat projektin tavoitteet ja asiakkaiden tarpeet. Vaatimusmäärittelyssä kirjataan tulevalle ohjelmistolle vaatimukset siitä, miten sen kuuluisi toimia. Suunnitteluvaiheessa voidaan suunnitella esimerkiksi ohjelmiston käyttöliittymää ja arkkitehtuuria. Toteutusvaiheessa ohjelmisto toteutetaan eli koodataan ja testausvaiheessa ohjelmisto testataan muun muassa sille laadittuja vaatimuksia vasten. Käyttöönotto- tai ylläpitovaiheessa ohjelmisto voidaan julkaista. [12.]

Valittu kehitysmalli riippuu hyvin paljon siitä, millaisesta projektista on kyse. Jotta oikea kehitysmalli voidaan valita, täytyy ensin tunnistaa projektin tyyppi. Projektin tyypit voidaan jakaa kolmeen eri kategoriaan, joita ovat perinteinen (traditional), ketterä (agile) ja äärimmäinen (extreme). [13.]

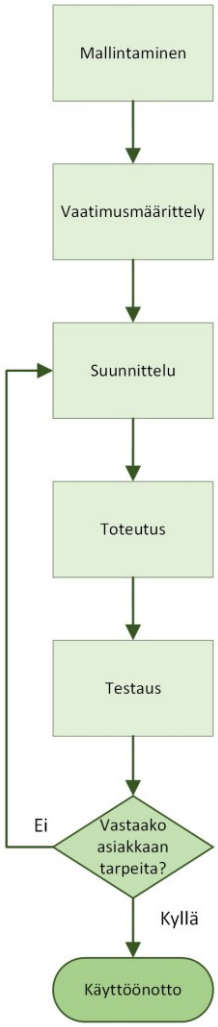
Perinteisissä projekteissa on alusta asti selvää, mitä tavoitellaan ja miten tämä tavoite tullaan saavuttamaan. Näissä projekteissa ei odoteta muutoksia määrittelyyn tai suunnitelmaan eli kehitysmalliksi voidaan valita lineaarinen tai inkrementaalinen kehitysmalli. Lineaarisessa mallissa ohjelmistokehityksen vaiheet

suoritetaan järjestyksessä eikä edelliseen vaiheeseen palata, ellei siihen laadita erikseen prosessia. Perinteisille malleille yhteistä on se, että ohjelmistokehityksen vaiheista mallintaminen, vaatimusmäärittely ja suunnittelu toteutetaan ensin kokonaan. Inkrementaalisessa kehitysmallissa seuraavat ohjelmistokehityksen vaiheet voidaan toteuttaa vaiheittain, jolloin toteutusvaiheeseen voidaan palata uudelleen. Tämä tekee inkrementaalisesta mallista joustavamman mallin verrattuna lineaariseen malliin. [13.]

Ketterissä projekteissa on selvää, mitä tavoitellaan lopputulokselta, mutta lopputuloksen saavuttaminen eli projektin ratkaisut ovat epäselviä. Toisin kuin perinteisissä projekteissa, ketterissä projekteissa odotetaan muutoksia toteutussuunnitelmiin. Ketteriin projekteihin voidaan valita kehitysmalliksi iteratiivinen tai adaptiivinen kehitysmalli. Näiden kahden mallin välillä ei ole suuria eroja. Adaptiivisessa kehitysmallissa projektin ratkaisut ovat epäselvempiä kuin iteratiivisessa mallissa. Lisäksi adaptiivisessa mallissa lopputulokseen voi liittyä joitakin avoimia kohtia. [13.]

Äärimmäisissä projekteissa haluttu lopputulos ja ratkaisut siihen ovat avoimia, jolloin voidaan odottaa paljon muutoksia projektin aikana. Koska haluttu lopputulos on äärimmäisissä projekteissa epäselvä, voi projektin aikana muutoksia tulla mallintamis- ja vaatimusmäärittelyvaiheisiin. Tämä ei ole mahdollista perinteisissä projektityypeissä, ja ketterissä projekteissa tätä pyritään välttämään. [13.]

Potilasmonitorin uuden toiminnallisuuden kohdalla iteratiivisen ketterän kehitysmallin kriteerit täyttyivät, sillä oli selvää, mikä uuden toiminnallisuuden tulisi olla valmistuessaan. Toisaalta keinot päästä haluttuun lopputulokseen olivat avoimia. Iteratiivista ketterää kehitysmallia on kuvattu kuvassa 3.

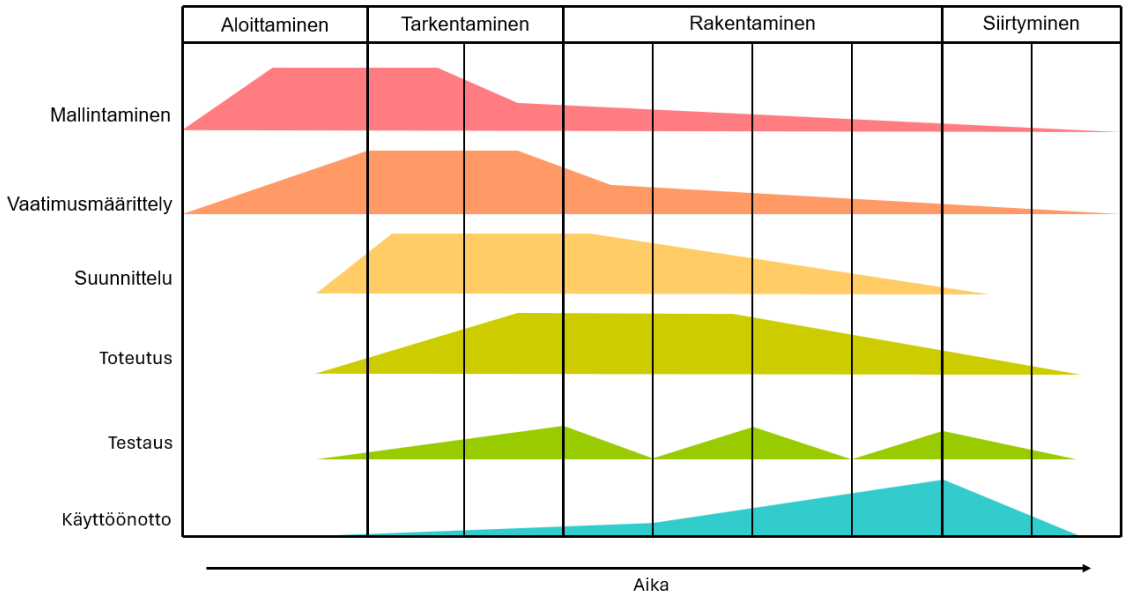


Kuva 3. Iteratiivisen ketterän ohjelmistokehityksen malli [13].

Kuvassa 3 iteratiivinen ketterä kehitysmalli on kuvattu vuokaavion avulla. Ohjelmistokehityksen vaiheet on sijoitettu allekkain järjestykseen ja ennen viimeistä vaihetta, eli käyttöönottoa, on valintakohta. Valintakohdasta voidaan palata tarkentamaan tai korjaamaan suunnitelmaa uudessa iteraatiossa, jos asiakkaan tarpeet eivät täyty. Iteraatiot sisältävät siis ohjelmistokehityksen vaiheet suunnittelun, toteutuksen ja testauksen. Iteraatioita voidaan toteuttaa siihen asti, kun asiakkaan tarpeiden koetaan täyttyvän ja ohjelmiston olevan vaatimusmäärittelyn mukainen. Tämän jälkeen voidaan siirtyä käyttöönottoon.

Kun projektille on valittu kehitysmalli projektin tyypin perusteella, täytyy päättää, millä tavalla malli voidaan toteuttaa. Yksi tapa toteuttaa iteratiivinen kehitysmalli

potilasmonitorin uuden toiminnallisuuden kohdalla on yhtenäistetyn ohjelmistokehitysmallin eli RUP-mallin (Rational Unified Process) avulla. [13.] Tätä mallia on havainnollistettu kuvassa 4.



Kuva 4. Yhtenäistetty ohjelmistokehitysmalli [14].

Kuvan 4 on yhtenäistetty ohjelmistokehitysmalli eli RUP-malli, joka yhdistää ja havainnollistaa, miten ohjelmistokehitysprojekti on iteratiivinen mutta myös inkrementaalinen samanaikaisesti. Mallissa aikaa kuvaa alareunassa oleva vasemmalle osoittava aikanuoli. RUP-mallissa aika on jaettu neljään osaan, jotka ovat aloittaminen, tarkentaminen, rakentaminen ja siirtyminen. Aloittamisvaiheessa luodaan pohja koko projektille eli siinä määritellään, mitä halutaan tehdä ja voidaan jo alkaa hahmottelemaan mahdollisia vaatimuksia. Tarkennusvaiheessa on tarkoitus tarkentaa ohjelmiston vaatimuksia ja hioa loppuun projektin tavoitteita. Lisäksi suunnittelua voidaan aloittaa. Rakentamisvaiheessa on tarkoitus rakentaa ja testata ohjelmisto ennen kuin voidaan aloittaa siirtymisvaihe. Siirtymisvaiheessa voidaan aloittaa ohjelmiston julkaiseminen ja käyttöönotto. [14.] Tarkentamis-, rakentamis- ja siirtymisvaiheet on jaettu kuvassa 4 pienempiin osiin kuvaamaan eri iteraatioita. Iteraatioita voi olla enemmänkin kuin kuvaan 4 on laitettu. Näiden neljän osan sisällä suoritetaan ohjelmistokehityksen vaiheita. Ohjelmistokehityksen vaiheet on lueteltu mallin vasempaan reunaan.

RUP-mallista tulee selkeästi esille se, että ohjelmistokehityksen vaiheita suoritetaan samanaikaisesti, mutta niillä on jokaisessa osassa erilainen painoarvo. Esimerkiksi projektin alkuvaiheessa keskitytään paljon mallintamiseen ja vaatimusmäärittelyyn, mutta näiden vaiheiden painoarvot pienenevät merkittävästi, kun siirrytään ajallisesti rakentamisvaiheeseen. Toisaalta käyttöönoton painoarvo on projektin alussa hyvin pientä, mutta sen merkitys kasvaa projektin puolivälistä eteenpäin aina siirtymisvaiheen alkuun asti.

Kun tarkastellaan yrityksen kehittämää potilasmonitorin uutta toiminnallisuutta, voidaan katsoa projektin olevan tarkentamisvaiheessa. Uudelle toiminnallisuudelle työstetään vaatimusmäärittelyä mutta sille on aloitettu tekemään ensimmäisiä versioita toiminnallisuuden ulkonäöstä ja varsinaisesta toteutuksesta. Lisäksi varsinaista toteutusta on voitu jo hieman aloittaa. RUP-malli kuten muutkin iteratiiviset mallit mahdollistavat nopean reagoinnin ongelmatilanteisiin, ja ne ovat joustavia muutoksien kannalta.

3.3 Vaatimusmäärittely

Minkä tahansa ohjelmiston kehitys alkaa siitä, että määritellään, millainen kehitettävän ohjelmiston tulisi olla. Tätä vaihetta kutsutaan myös vaatimusmäärittelyksi. Vaatimusmäärittelyn vaatimukset voivat määritellä esimerkiksi, miten ohjelmiston tulisi toimia tai millaisia rajoituksia ohjelmistolle tulisi asettaa. Kaikki vaatimukset muodostavat ohjelmiston eri ominaisuudet. Valmis vaatimusmäärittely on selkeä ja ymmärrettävä kaikille ohjelmiston sidosryhmille. Sidosryhmillä tarkoitetaan ryhmiä tai henkilöitä, jotka voivat vaikuttaa kehitettävään ohjelmistoon kuten loppukäyttäjät tai ohjelmistokehittäjät. [15; 16.]

Vaatimusmäärittely aloitetaan selvittämällä ja ymmärtämällä, miksi uusi ohjelmisto tai sen osa tarvitaan, mitä uuden ohjelmiston tai sen osan tulisi tehdä, ja kuka tai ketkä tulevat olemaan tekemisissä ohjelmiston tai sen osan kanssa. Myös ohjelmiston toimintaympäristön ymmärtäminen on tärkeää. Näiden asioiden selvittäminen rajaa ongelman uudelle ohjelmistolle, joka tulee ratkaista. Ongelman ymmärtäminen mahdollistaa, että vaatimusmäärittelyssä saavutetaan

tilanne, jossa ongelmaa ei enää esiinny. Kysymyksiin pyritään saamaan vastauksia keräämällä tietoa ohjelmiston sidosryhmiltä. [15; 16.] Vastaukset voivat tulla esimerkiksi haastatteluiden, kyselyiden tai erilaisten palautteiden kautta. Saadut vastaukset tulee analysoida tarkasti, jotta voidaan saada tarkempia vastauksia aiemmin mainittuihin kysymyksiin. [16.] Tarkemmat vastaukset mahdollistavat tarkemman vaatimusmäärittelyn, jolloin lopputulos vastaa tarkemmin haluttua lopputulosta kaikkien sidosryhmien ja varsinkin asiakkaiden kannalta. Eri sidosryhmiltä voi tulla risteäviä mielipiteitä, joten vaatimusmäärittelyyn täytyy tehdä kompromisseja [15].

Vaatimusmäärittely voidaan jakaa kahteen eri tasoon. Nämä ovat järjestelmä- ja ohjelmistotaso. Järjestelmätaso on ohjelmistotason yläpuolella ja molemmilla tasoilla on omat vaatimuksensa. Eri tasoilla olevat vaatimukset eivät voi kumota toisiaan. Järjestelmätason vaatimukset kattavat koko järjestelmän eli esimerkiksi ohjelmiston, käyttäjien ja lisälaitteiden muodostaman kokonaisuuden vaatimukset. Ohjelmistotason vaatimukset keskittyvät vain itse ohjelmiston toimintaan. Ohjelmiston toiminta voi tarkoittaa taustalla tapahtuvia prosesseja tai ohjelmiston näkyviä osia eli käyttöliittymää. [15; 16.] Järjestelmätason vaatimukset tulee olla selvillä ennen kuin voidaan aloittaa ohjelmistotason vaatimuksien kirjoittaminen.

Ohjelmistotason vaatimukset voidaan jakaa kahteen eri kategoriaan, jotka ovat toiminnalliset ja ei-toiminnalliset vaatimukset. Toiminnalliset vaatimukset ovat vaatimuksia, jotka kertovat, mitä ohjelmisto tekee. Ei-toiminnalliset vaatimukset kertovat, miten ohjelmisto täyttää toiminnalliset vaatimukset. Ei-toiminnallisiin vaatimuksiin liittyy useita eri luokkia kuten

- laatuvaatimukset
- mukautuvuusvaatimukset
- arkkitehtuurivaatimukset
- kehitystyön vaatimukset.

Nämä luokat voidaan jakaa edelleen vielä pienempiin osiin, jotta ohjelmiston jokainen osa tulee käsiteltyä vaatimusmäärittelyssä. Ohjelmistotason

vaatimuksissa täytyy muistaa, että ne ovat myös järjestelmätason vaatimuksia, jolloin turhaa toistoa vaatimuksien välillä on hyvä välttää. [15.]

Vaatimusmäärittely on prosessina iteratiivinen, mikä tarkoittaa, että vaatimuksia voidaan parannella ja lisätä tarvittaessa, kun ohjelmiston tavoitteet tarkentuvat kehitystyön edetessä. Parannukset voivat myös liittyä esimerkiksi vaatimuksien yhdenmukaisuuteen ja ymmärrettävyyteen. [15.] Kaikkien sidosryhmien tulee ymmärtää ohjelmiston toiminta samalla tavalla.

Tärkeä osa vaatimusmäärittelyä on myös vaatimuksien jäljitettävyys. Jäljitettävyydellä tarkoitetaan sitä, että jälkikäteen voidaan helposti selvittää vaatimuksien alkuperä, sekä sitä, miten esimerkiksi eri ohjelmistotason vaatimukset linkittyvät järjestelmätason vaatimuksiin. [15; 16.] Kun vaatimusmäärittelyn koetaan olevan valmis ja kattavan kaiken ohjelmistoon liittyvän, voidaan vaatimusmäärittely vahvistaa [15]. Kun vaatimusmäärittely on vahvistettu, voi kehitystyö alkaa kunnolla, vaikka vaatimuksiin tehtäisiin vielä myöhemmin muutoksia.

3.4 Ohjelmistotestaus ja verifikaatio

Ohjelmistokehitys etenee tyypillisesti vaatimusmäärittelyn kautta käyttöliittymän suunnitteluun, arkkitehtuurin kuvaukseen, suunnitteluun ja lopulta myös itse ohjelmiston toteutukseen. Näiden vaiheiden jälkeen tai samanaikaisesti aloitetaan ohjelmistotestaus ja verifikaatio. Ohjelmistotestauksella tarkoitetaan sen nimen mukaisesti testausta, jonka tarkoituksena on selvittää, että ohjelmisto toimii vaaditulla tavalla.

Verifikaatio on ohjelmistotestausta, jonka tehtävänä on varmistaa, että ohjelmisto toimii sen vaatimusmäärittelyn mukaisesti ja että jokaisella vaatimuksella on olemassa jokin testi [16]. Verifikaatiossa tuotetaan myös virallinen dokumentaatio ja todisteet siitä, että ohjelmisto toimii sille määriteltyjen vaatimuksien mukaisesti. Verifikaatio toteutetaan ohjelmiston ollessa lähestulkoon valmis, kun taas ohjelmistotestausta voidaan tehdä läpi koko kehitystyön iteratiivisesti samoin kuin muitakin ohjelmistokehityksen vaiheita. Toisaalta, jos ohjelmistoon

tehdään verifikaation jälkeen muutoksia, joudutaan verifikaatio toistamaan. Ennen verifikaatioon siirtymistä, täytyy varmistaa, että jokaiselle ohjelmiston vaatimukselle löytyy testi.

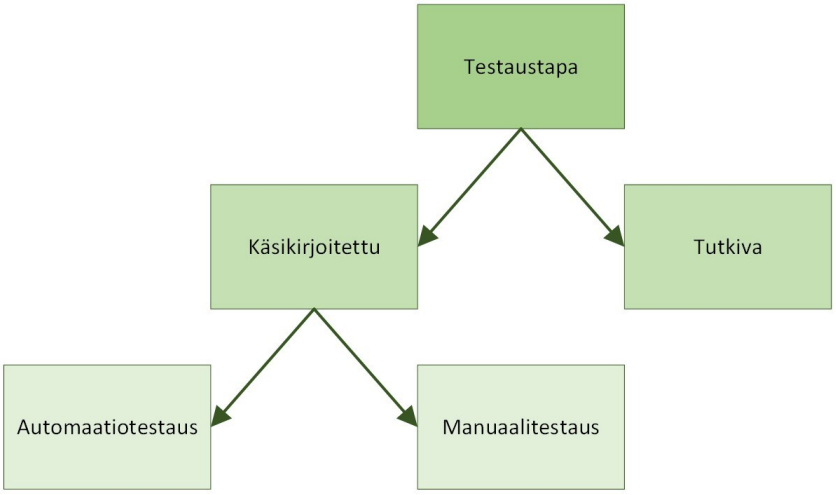
Ohjelmistotestausta on hyvä tehdä koko kehitystyön ajan, jotta mahdollisiin vikoihin voidaan reagoida nopeasti eikä verifikaation aikana jouduta tekemään niin paljon uudelleentestausta. Ennen kuin ohjelmistotestausta voidaan tehdä, täytyy määritellä testausstrategia, johon kuuluvat muun muassa testauksen taso ja tavat sekä testausympäristö ja tarvittavat välineet. [17.] Testauksen tasosta riippuen testausstrategian määrittelyn taso ja kompleksisuus voi vaihdella.

Ohjelmistotestauksella on useita eri tasoja ja niitä käytetään ohjelmistokehityksen eri vaiheissa. Näitä tasoja ovat

- yksikkötestaus
- integraatiotestaus
- järjestelmätestaus
- hyväksymistestaus [17].

Yksikkötestauksessa keskitytään ohjelmiston yksittäisiin osiin. Kyseinen taso sopii hyvin ohjelmistokehityksen alkuvaiheisiin, kun ohjelmistosta on vain pieniä osia testattavana. Integraatiotestauksessa testataan useamman ohjelmiston eri osan toimivuutta yhdessä, jolloin tasoa voidaan hyödyntää hieman myöhemmässä vaiheessa. Järjestelmä- ja hyväksymistestauksessa käydään läpi koko järjestelmän toimivuutta. Erona näiden kahden välillä on se, että järjestelmätestauksessa keskitytään vaatimuksien täyttymiseen, kun taas hyväksymistestaus varmistaa, että asiakkaan tarpeet todellisessa käyttöympäristössä täyttyvät. [18.] Verifikaation voidaan katsoa olevan järjestelmätestausta.

Ohjelmistotestauksella on monia eri tapoja. Näitä tapoja on mallinnettu kuvassa 5.



Kuva 5. Ohjelmistotestaustapojen jakautuminen [17].

Kuvassa 5 ensimmäinen jako eri tapojen välillä voidaan tehdä siihen, onko testaus käsikirjoitettua vai tutkivaa. Käsikirjoitetussa testauksessa suoritetaan ennalta määriteltyjä testitapauksia ja kirjataan saadut tulokset ylös. Tutkivassa testauksessa testaajalle on monesti määritelty etukäteen, mitä ohjelmiston osia tulisi testata mutta testitapauksia ei ole määritelty. Tutkivaan testaukseen voidaan asettaa jokin tarkka aikaraja, jonka jälkeen havaintoja voidaan kirjata ylös. [17.] Käsikirjoitettuun testaukseen ei liity samanlaista aikarajaa.

Kuvan 5 mukaisesti ohjelmistotestauksen eri tapoja voidaan myös jakaa sen perusteella, suoritetaanko testaus manuaalisesti vai käytetäänkö apuna automaatiotyökaluja. Manuaalitestauksessa testaaja käy käsin läpi ohjelmistoa ennalta määritellyllä tavalla joko käsikirjoituksen mukaisesti tai tutkivasti. Manuaalitestauksessa testaajan täytyy käsin kirjata testauksen vaiheet ja tulokset ylös, mikä lisää mahdollisten virhekirjauksien määrää. Manuaalitestaus on tämän takia myös aikaavievää. Toisaalta manuaalitestaus on helppo toteuttaa, sillä siihen ei tarvita lisälaitteita tai -ohjelmistoja, joita automaatiotestaus tarvitsee. Automaatiotestit ovat käsikirjoitettuja testitapauksia, joita automaatiotestaustyökalu suorittaa erillisen ohjelmakoodin avulla. Automaatiotestaus vie vähemmän aikaa kuin manuaalitestaus, koska se ei vaadi työlästä dokumentaatiota, jolloin virheiden määräkin pienenee. Automaatiotestaus ei voi olla tutkivaa, sillä

automaatiotestauksessa käytettävä ohjelmakoodi tulee määritellä ennen testauksen suorittamista toisin kuin manuaalitestauksessa testitapaukset.

4 Uuden toiminnallisuuden vaatimusmäärittely

4.1 Lähtökohdat vaatimusmäärittelylle

Potilasmonitorin uudelle toiminnallisuudelle löytyi ennen ohjelmistotason vaatimusmäärittelyä valmiita dokumentteja ja muita asioita, joita pystyttiin hyödyntämään vaatimusmäärittelyn teon aikana. Uudelle toiminnallisuudelle oli kirjoitettu jo järjestelmätason vaatimukset, joiden perusteella ohjelmistotason vaatimusmäärittely voitiin alun perin aloittaa. Ennen järjestelmätason vaatimuksia uudelle toiminnallisuudelle oli kirjoitettu myös käyttäjävaatimukset sekä toteutettu riskien hallintaa.

Uusi toiminnallisuus tulee korvaamaan nykyisessä ohjelmistossa olevan toiminnallisuuden. Toiminnallisuuksien välillä tulee olemaan yhteneväisyyksiä, sillä niillä pyritään tekemään samoja asioita. Yrityksen sisällä oli myös laadittu uudesta toiminnallisuudesta ja sen eri ominaisuuksista dokumentti, jonka perusteella eroavaisuudet ja yhteneväisyydet toiminnallisuuksien välillä olivat hyvin erotettavissa. Vanhasta toiminallisuudesta oli olemassa ohjelmistotason vaatimusmäärittely, mutta sitä ei juurikaan voitu hyödyntää eroavaisuuksien takia.

Kaikkien edellä mainittujen asioiden lisäksi uudesta toiminnallisuudesta oli jo hahmoteltu mahdollista käyttöliittymää. Käyttöliittymän hahmotelmaa voitiin myös käyttää apuna vaatimusmäärittelyn aikana yhdessä kaikkien edellä mainittujen dokumenttien lisäksi.

Ennen vaatimusmäärittelyn aloittamista tuli varautua uuteen toiminnallisuuteen liittyviin epäselvyyksiin, koska kaikki yksityiskohdat eivät olleet selvillä. Epäselvyyksiin voitiin kysyä vastauksia esimerkiksi uudesta toiminnallisuudesta vastaavan Product Ownerilta tai käyttöliittymän suunnittelijoilta. Näiden henkilöiden kautta vastauksia saatiin nopeasti.

4.2 Prosessin suunnittelu

Ennen varsinaisen vaatimusmäärittelyn aloittamista tehtiin suunnitelma koko vaatimusmäärittelyprosessista. Prosessi jaettiin kolmeen päävaiheeseen, jotka olivat uuden toiminnallisuuden hahmottaminen, ensimmäisten vaatimuksien kirjoittaminen ja lopullisten vaatimuksien kirjoittaminen. Prosessin lopputuloksena tavoiteltiin valmista vaatimusmäärittelyä potilasmonitorien uudelle toiminnallisuudelle. Vaatimusmäärittelyprosessin suunnitelma on esitetty kuvassa 6.

Kuva 6. Vaatimusmäärittelyn prosessikaavio.

Kuvan 6 prosessikaavion alkupisteenä oli edellisessä luvussa esitetyt lähtötiedot ohjelmistotason vaatimusmäärittelylle. Seuraavassa vaiheessa tehtiin

suunnitelma vaatimusmäärittelyn toteutuksesta, joka on nähtävissä kuvan 6 seuraavissa vaiheissa. Varsinainen vaatimusmäärittely alkoi ajatuskartan tekemisestä uudelle toiminnallisuudelle. Ajatuskartan tarkoituksena oli pilkkoa uutta toiminnallisuutta pienempiin osiin auttaen hahmottamaan sen eri ominaisuuksia. Lisäksi ajatuskarttaa voitiin hyödyntää muissa vaatimusmäärittelyprosessin vaiheissa, jotta mitään toiminnallisuuden osa-aluetta ei unohdettaisi. Kun ajatuskartta oli valmis, järjestettiin katselmointitilaisuus yrityksen sisällä, jonka jälkeen tehtiin tarvittavat korjaukset ennen siirtymistä seuraavaan vaiheeseen.

Ajatuskartan ja sen korjauksien jälkeen aloitettiin kirjoittamaan ensimmäisiä versioita vaatimuksista. Kun ensimmäiset versiot vaatimuksista olivat valmiita, järjestettiin jälleen katselmointitilaisuus, jonka jälkeen tehtiin tarvittavat korjaukset. Korjauksien jälkeen vaatimukset voitiin kirjata DOORS-vaatimustenhallintaohjelmaan. Tämän jälkeen siirryttiin vaiheeseen, jossa vaatimukset luokiteltiin niiden ominaisuuksien mukaan.

Luokittelun jälkeen siirryttiin viimeiseen katselmointivaiheeseen, jonka tarkoituksena oli hioa vaatimuksista mahdollisimman valmiita ja hyväksyttää ne yrityksen sisällä. Vaatimukset tuli hyväksyttää uudesta toiminnallisuudesta vastaavan ohjelmistokehitystiimin sisällä ja järjestelmäsuunnittelijoiden kanssa. Katselmoinneista tulevia korjauksia tehtiin sitä mukaa, kuin niitä tuli vastaan.

Edellä kuvattujen vaiheiden lopputuloksena oli järjestelmäsuunnittelijoiden hyväksymä vaatimusmäärittely uudelle potilasmonitorin toiminnallisuudelle. Valmiilla vaatimusmäärittelyllä ei tässä tapauksessa tarkoitettu lopullista vaatimusmäärittelyä. Koska potilasmonitorin uutta toiminnallisuutta kehitetään iteratiivisesti, saattaa vaatimusmäärittelyyn tulla muutoksia kehitystyön myöhemmässä vaiheessa, joita ei käsitellä tämän insinöörityön aikana.

4.3 Ajatuskartta

Ajatuskartta uudesta toiminnallisuudesta tehtiin Miro-alustalla (https://miro.com/). Ajatuskartan tekoon käytettiin apuna alustavia

käyttöliittymän suunnitelmia ja yrityksen sisäistä dokumenttia, johon uusi toiminnallisuus oli hyvin alustavalla tasolla hahmoteltu. Ajatuskartan keskelle kirjoitettiin uuden toiminnallisuuden nimi. Seuraavaksi ajatuskartta jaettiin toiminnallisuuteen liittyvien valikoiden mukaan. Ajatuskarttaa jatkettiin pilkkomalla eri valikoiden sisältöjä esimerkiksi painikkeiden mukaan. Eri painikkeista lisättiin maininnat niiden toiminnallisuuksista ja siitä, miten ne mahdollisesti vaikuttaisivat muiden valikoiden sisältöihin. Kaikista tarkimpia yksityiskohtia toiminnallisuudesta ei avattu ajatuskartassa, koska sen tarkoituksena oli vain selvittää uuden toiminnallisuuden pääpiirteet yleisemmällä tasolla. Tiedossa oli myös, että vaatimuksia kirjoittaessa kaikki tarkemmatkin yksityiskohdat tultaisiin avaamaan.

Ajatuskartan teossa täytyi huomioida GE HealthCaren potilasmonitorien ja parametrilaitteiden eri näyttökokoja. Eri näyttökoot vaikuttivat pääosin uuden toiminnallisuuden ja painikkeiden ulkonäköihin. Perusperiaate säilyi kuitenkin samana näyttökoosta riippumatta. Eroavaisuudet näyttökokojen takia aiheuttivat lisähaaroja ajatuskarttaan.

Kun ajatuskartta oli valmis, järjestettiin katselmointitilaisuus uudesta toiminnallisuudesta vastaavan ohjelmistokehitystiimin sisällä. Tilaisuudessa katselmoitiin tehty ajatuskartta. Tilaisuudessa käytettiin apuna dokumenttia uudesta toiminnallisuudesta. Katselmoinnin jälkeen ajatuskarttaan ei tarvinnut tehdä täydennyksiä. Yrityksen sisäiseen dokumenttiin tehtiin muutamia tarkennuksia, jotta ajatuskartta ja dokumentti vastaisivat tosiaan tarkemmin. Tarkennuksien jälkeen siirryttiin kirjoittamaan ensimmäisiä vaatimuksia.

4.4 Vaatimuksien laatiminen

Vaatimusmäärittelyn toisessa vaiheessa oli tarkoitus kirjoittaa vaatimukset uudelle toiminnallisuudelle aiemmin luodun ajatuskartan ja yrityksen sisäisten dokumenttien avulla. Koska uudelle toiminnallisuudelle oli jo kirjoitettu järjestelmätason vaatimukset, pystyttiin myös niitä hyödyntämään ohjelmistotason vaatimuksissa.

Yrityksessä toiminnalliset ja ei-toiminnalliset vaatimukset kirjataan samaan ohjelmistovaatimusdokumenttiin. Ei-toiminnalliset vaatimukset tunnetaan yrityksessä termillä DD (Design Detail). DD:t erotetaan muista vaatimuksista kirjoitusasun avulla seuraavasti:

- should = DD, ei-toiminnallinen vaatimus, joka määrittelee, miten ohjelmisto toimii
- shall = toiminnallinen vaatimus, joka määrittelee, mitä ohjelmisto tekee.

Yritys on määritellyt, että vaatimuksissa tulisi käyttää Gherkin-tyyliä (https://cucumber.io/docs/gherkin/). Gherkin-tyylissä vaatimukset muodostetaan käyttäen ”given-when-then”-rakennetta. ”Given” määrittelee alkuehdot, ”when” kertoo sen, mitä tapahtuu, ja ”then”-kohdassa kerrotaan lopputulos. Kaikkiin vaatimuksiin ei tarvita alkuehtoja, jolloin vaatimukseen riittävät tyylin kaksi viimeistä kohtaa. Myös ”and”-sanan käyttö vaatimuksissa on mahdollista, jos esimerkiksi yhteen vaatimukseen liittyy useampi alkuehto. Gherkin-tyyliä halutaan hyödyntää, koska se on helposti ymmärrettävissä kaikille sidosryhmille. Yritys on määritellyt DD:ille vapaamman kirjoitusasun, sillä Gherkin-tyyli ei välttämättä sovellu, jos esimerkiksi määritellään, mitä jossakin tietyssä painikkeessa tulisi lukea. Lisäksi yrityksellä on sisäisiä dokumentteja, joissa määritellään, mitä termejä käytetään käyttöliittymän eri komponenteista kuten painikkeista. Tällä tavalla yritys saa kaikista vaatimusdokumenteistaan yhteneväisiä.

Vaatimuksien kirjoittaminen aloitettiin hahmottelemalla uudelle toiminnallisuudelle kappalejakoa. Tämä toteutettiin aluksi jakamalla eri valikot omiin päälukuihinsa. Päälukujen alle tehtiin alalukuja valikoiden sisällä olevien painikkeiden tai ominaisuuksien mukaan. Kappalejaon jälkeen aloitettiin vaatimuksien kirjoittaminen. Kirjoittaminen toteutettiin käymällä läpi ajatuskartan haaroja yksitellen. Tässä vaiheessa ei mietitty, ovatko kirjoitetut asiat toiminnallisia vaatimuksia (shall) vai DD:itä (should). Kirjoitusasu pyrittiin kuitenkin pitämään mahdollisimman lähellä yrityksen ohjeistusta korjausmäärän pienentämiseksi. Kirjoitettuja vaatimuksia jaettiin jo tässä vaiheessa eri CARESCAPE-tuoteperheen laitteisen mukaan.

Koska kirjoitettujen vaatimuksien määrä paisui jo tässä vaiheessa melko suureksi, päätettiin, että vaatimuksien taulukoiminen voisi olla järkevää. Tehty vaatimustaulukko tehtiin taulukon 1 mukaisesti.

Taulukko 1. Vaatimustaulukon pohja.

Tunniste	**Vaatimus**	**Tyyppi**	**Laitteet**	**Muuta**

Taulukon 1 ensimmäiseen sarakkeeseen kirjoitettiin oma tunniste jokaiselle vaatimukselle. Seuraavaan sarakkeeseen kirjoitettiin itse vaatimus. Taulukkoon 1 lisättiin myös Tyyppi-sarake, johon kirjattiin, onko rivillä oleva vaatimus tavallinen vaatimus vai DD. Sarake pyrittiin täyttämään mahdollisimman hyvin, mutta avoimia kohtia jäi silti. Laitteet-sarakkeeseen lisättiin kaikki ne potilasmonitorit tai parametrilaitteet, joita rivillä oleva vaatimus koski. Viimeinen Muuta-sarake lisättiin taulukkoon katselmointia varten. Taulukon 1 sarakkeet valittiin mukailemaan DOORS-ohjelmassa olevaa ohjelmistovaatimusdokumenttipohjaa, jotta luokittelu olisi myöhemmin helpompi toteuttaa.

Ennen vaatimustaulukolle pidettyä katselmointia taulukossa oli 164 kohtaa, joista 15 oli vaatimuksia ja muut DD:itä. Vaatimustaulukko katselmoitiin uudesta toiminnallisuudesta vastaavan tiimin sisällä. Katselmoinnista nousseita kommentteja olivat muun muassa:

- yhdessä vaatimuksessa useampi toiminto
- vaatimuksien siirtäminen toisen alaotsikon alle
- painikkeiden ulkonäkö ja toiminto samassa vaatimuksessa
- ”not”-sanan käyttö
- kielioppi- ja rakennemuutoksia esimerkiksi Gherkin-tyyliin liittyen
- puuttuvat vaatimukset
- vaatimukset kirjoitetaan positiivisen kautta esimerkiksi tilanteessa, jossa jokin painike näkyy vain tietyn valikon ollessa auki, ei tarvitse määritellä erikseen painikkeen näkyvyyttä valikon ollessa kiinni.

- tyylimuutokset ja sanavalinnat yrityksen ohjeistuksen mukaisesti.

Saadut kommentit kirjoitettiin katselmoinnin aikana Taulukon 1 Muuta-sarakkeeseen, jotta jokainen vaatimus olisi helpompi korjata. Pieniä muutoksia jouduttiin tekemään melko paljon. Muutoksien jälkeen vaatimustaulukossa oli 219 kohtaa, joista kaikki olivat DD:itä. Tiimin kanssa sovittiin, että tässä vaiheessa ei tarvitse tarkemmin miettiä, mikä on vaatimus ja mikä on DD. Nämä asiat voitaisiin katselmoida myöhemmin järjestelmäsuunnittelijoiden kanssa ja kattavien järjestelmävaatimuksien takia ohjelmistotasolle ei välttämättä tulisi lainkaan toiminnallisia vaatimuksia. Tiimin kanssa sovittiin, kun katselmoinnista tulleet kommentit olivat korjattu, vaatimukset voitaisiin siirtää DOORS-ohjelmaan. Vaatimusmäärittelyyn liittyi katselmoinnin jälkeen muutamia avoimia kysymyksiä, jotka tuli vielä selvittää sidosryhmiltä, mutta ne eivät estäneet siirtymistä vaatimusmäärittelyn seuraaviin vaiheisiin.

4.5 DOORS

Jokainen potilasmonitorin ohjelmistossa oleva toiminto on jaettu johonkin kategoriaan. Näitä kategorioita ovat muun muassa kaikki potilasmonitorin fysiologiset parametrit ja muut toiminnallisuudet, jotka eivät suoraan liity mihinkään parametriin kuten käyttöliittymän yleiset komponentit ja monitorin hälytyksien yleinen toiminnallisuus. Jokaiselle kategorialle on luotu DOORS-ohjelmaan oma ohjelmistovaatimusdokumentti. Ohjelmistovaatimusdokumenteissa kyseessä oleva kategoria on jaettu pienempiin osiin eri päälukujen alle. Tyypillisesti yhden pääluvun alla on yksi toiminnallisuus tai muu suurempi kokonaisuus. Päälukujen alla voi olla useampia alalukuja ja alalukujen alalukuja riippuen siitä, kuinka monimutkaisesta ohjelmiston osasta on kyse.

Uudelle toiminnallisuudelle oli jo tehty oma pääluku, mutta ennen vaatimuksien siirtämistä vaatimustaulukosta DOORS-ohjelmaan, luotiin alaotsikot samalla tavalla kuin ne olivat taulukossa. Vaatimuksien siirtäminen DOORS-ohjelmaan toteutettiin kopioimalla vaatimukset yksitellen vaatimustaulukosta. Seuraavaksi siirryttiin vaatimuksien luokitteluun. Vaatimusdokumentissa olivat taulukon 2 mukaiset sarakkeet, jotka tuli täydentää jokaisen vaatimuksen osalta.

Taulukko 2. Esimerkki vaatimusdokumentin pohjasta.

Tunniste	Vaatimus	Tyyppi	Laitteet	Ohjelmisto	Projekti	Verifikaatiometodi

Taulukon 2 ensimmäiseen sarakkeeseen DOORS-ohjelma loi jokaiselle vaatimukselle oman tunnisteen, josta selvisi myös kategoria, johon uusi toiminnallisuus liittyy sekä lyhyt numerosarja. Seuraavaan sarakkeeseen kirjoitettiin itse vaatimus. Tyyppi-sarakkeeseen valittiin, onko kyseessä vaatimus vai DD. Tässä vaiheessa kaikki olivat DD:itä. Taulukon 2 Laitteet-sarakkeeseen lisättiin samaan tyyliin kuin taulukossa 1 kaikki rivillä olevaan vaatimukseen liittyvät laitteet. Ohjelmisto-sarakkeeseen kirjattiin jokaisen vaatimuksen kohdalle ohjelmistoversio, jossa uusi toiminnallisuus tullaan julkaisemaan. Projekti-sarakkeeseen täytettiin tieto siitä projektin nimestä, jossa uusi toiminnallisuus tullaan julkaisemaan. Verifikaatiometodi-sarakkeeseen vaihtoehtoina olivat automaatio-, manuaalitestaus sekä koodikatselmointi. Koska tässä vaiheessa kaikki vaatimukset olivat DD:itä, kaikkiin laitettiin verifikaatiometodiksi manuaalitestaus yrityksen käytäntöjen mukaisesti. Luokittelun jälkeen kaikki vaatimukset käytiin vielä kerran läpi ennen katselmointeihin siirtymistä. Vaatimuksiin tehtiin pieniä korjauksia liittyen sanavalintoihin ja vaatimuksien rakenteisiin, jotta vaatimukset olisivat helpommin luettavia ja noudattaisivat tarkemmin yrityksen sisäisiä ohjeita.

4.6 Vaatimusmäärittelyn hyväksyttäminen

Kun vaatimusmäärittely uudelle toiminnallisuudelle oli siirretty DOORS-ohjelmaan luokiteltuna, järjestettiin ensimmäinen katselmointitilaisuus. Tilaisuuteen osallistui vain uudesta toiminnallisuudesta vastaavan ohjelmistokehitystiimin Product Owner. Tilaisuudessa käytiin yleisellä tasolla vaatimusmäärittelyyn tehtyjä muutoksia, jotka tehtiin ennen siirtoa DOORS-ohjelmaan. Lisäksi tilaisuudessa keskusteltiin vaatimusmäärittelyyn liittyvistä epäselvistä asioista, jotka tulisi selvittää sidosryhmien kanssa. Katselmointitilaisuudessa ei käyty läpi kaikkia

vaatimuksia yksitellen, koska tämän ei nähty olevan järkevää tässä vaiheessa eikä aika olisi riittänyt kaikkiin vaatimuksiin.

Osaan asioista, jotka olivat aiemmin olleet epäselviä, oli saatu vastauksia. Vastauksien takia vaatimusmäärittelyyn jouduttiin tekemään muutaman vaatimuksen osalta korjauksia ja lisäyksiä. Ensimmäisen katselmointitilaisuuden päätteeksi sovittiin seuraavasta tilaisuudesta, johon otettaisiin mukaan uudesta toiminnallisuudesta vastaavasta tiimistä ohjelmistokehittäjä ja -testaaja sekä järjestelmäsuunnittelutiimistä yksi suunnittelija. Seuraavaan katselmointitilaisuuteen valmistauduttiin keräämällä valmiiksi kysymyksiä yksittäisiin vaatimuksiin ja yleisempiin avoimiin asioihin liittyen. Kerättyihin kysymyksiin saatiin tilaisuuteen osallistuneilta henkilöitä vastaukset, joiden avulla pystyttiin tekemään tarvittavia korjauksia vaatimuksiin. Lisäksi tilaisuuteen osallistuneelta järjestelmäsuunnittelijalta saatiin vielä myöhemmin muutamia korjausehdotuksia joihinkin vaatimuksiin.

Vaatimusmäärittelyn lopputuloksena dokumentissa oli 231 kohtaa, joista kaikki olivat DD:itä. Siihen, että yhtään toiminnallista vaatimusta ei tullut, vaikuttivat hyvin kattavat järjestelmävaatimukset. Valmistunut vaatimusmäärittely ei kuitenkaan ollut uuden toiminnallisuuden kannalta lopullinen, sillä uuteen toiminnallisuuteen liittyi vielä avoimia osa-alueita, joihin ei voitu vielä kirjoittaa vaatimuksia. Avoimet osa-alueet eivät ole kuitenkaan merkittävässä roolissa, kun katsotaan koko uutta toiminnallisuutta. Lopputuloksessa ovat uuden toiminnallisuuden keskeisimmät osa-alueet ja toiminnot on kuvattu kattavasti yrityksen ohjeistuksien mukaisesti ohjelmistotason vaatimusmäärittelyssä.

5 Testauksen suunnittelu

5.1 Lähtökohtien selvitys

Ohjelmistotason vaatimusmäärittelyn jälkeen siirryttiin insinöörityön toiseen vaiheeseen, jossa oli tarkoitus suunnitella, miten yrityksen tulisi testata uusi toiminnallisuus. Testauksen suunnittelun alussa uuteen toiminnallisuuteen liittyi

testauksen kannalta vielä avoimia kysymyksiä, jotka täytyi selvittää ennen kuin voitiin siirtyä itse vaatimuksien analysointiin. Avoimia kysymyksiä olivat:

- Onko automaatiotestaus uudelle toiminnallisuudelle mahdollista?
- Mitä vaatimuksia tulisi tarkastella?

Ensimmäiseen kysymykseen kysyttiin vastausta uudesta toiminnallisuudesta vastaavan tiimin ohjelmistokehittäjältä ja toisessa ohjelmistokehitystiimissä työskentelevältä ohjelmistotestaajalta, joka on ollut paljon kehittämässä yrityksen automaatiotestausta. Keskustelujen perusteella automaatiotestauksen pitäisi onnistua uudelle toiminnallisuudelle, mutta tällä hetkellä sille ei löytyisi tukea. Tuki olisi kuitenkin mahdollista toteuttaa.

Koska ohjelmistotason vaatimusmäärittelyyn ei tarvittu yhtään toiminnallista vaatimusta, päätettiin toisen kysymyksen osalta tarkastella uuteen toiminnallisuuteen liittyviä järjestelmätason vaatimuksia. Järjestelmätason vaatimuksille ei ollut vielä suunniteltu lainkaan testausta, joten niiden tarkastelu oli insinöörityön kannalta hyvin perusteltua. Todettiin myös, jos ohjelmistotasolle lisättäisiin myöhemmässä kehitystyön iteraatiossa toiminnallisia vaatimuksia, olisi nämä vaatimukset helppo upottaa järjestelmätason testeihin. Lisäksi sovittiin, että uutta toiminnallisuutta olisi hyvä testata mahdollisimman paljon automaatiotesteillä, jotta verifikaatioon kuluvaa aikaa voitaisiin pienentää. Toisaalta olisi hyvä, että uudelle toiminnallisuudelle tulisi vähintään yksi manuaalitesti, jotta toiminnallisuutta testattaisiin mallintamalla oikeiden käyttäjien toimintaa.

5.2 Vaatimuksien analysointi

Vaatimuksien analysointi aloitettiin listaamalla kaikki uuteen toiminnallisuuteen liittyvät vaatimukset järjestelmävaatimuksista. Uudelle toiminnallisuudelle tunnistettiin järjestelmätason vaatimuksista yhteensä 17 vaatimusta. Näistä vaatimuksista 3 olivat samoja kuin vanhassa toiminnallisuudessa. Vanhoille vaatimuksille löytyivät myös manuaalitestit. Nämä manuaalitestit luettiin läpi ja todettiin, että ne eivät sellaisenaan sopisi uuden toiminnallisuuden verifiointiin. Tämän takia myös vanhat vaatimukset otettiin mukaan vaatimuksien analysointiin.

Vaatimuksien analysointi toteutettiin kahdessa iteraatiossa. Ensimmäisessä iteraatiossa pyrittiin arvioimaan jokaisen vaatimuksen kohdalta oikea testausmenetelmä. Analysoinnissa otettiin huomioon muun muassa automaatiotestauksen rajoitukset ja vaatimuksien testaamiseen tarvittava laitteiden määrä. Toiseen iteraatioon otettiin mukaan uutta toiminnallisuutta kehittävästä tiimistä ohjelmistokehittäjä ja -testaaja. Heille esiteltiin ensimmäisen iteraation tulokset ja tehtiin tarvittavat muutokset testausmenetelmiin. Vaatimuksien analysoinnin tulokset kirjattiin taulukkoon 3.

Taulukko 3. Yhteenveto järjestelmävaatimuksien analysoinnista.

Vanhat vaatimukset	**Uudet vaatimukset**	**Vaatimukset yhteensä**	**1. iteraation automaatiotestit**	**2. iteraation automaatiotestit**
3	14	17	5	11

Taulukkoon 3 kirjattiin ylös vaatimuksien jakautuminen vanhoihin ja uusiin vaatimuksiin sekä niiden yhteismäärä. Ensimmäisen iteraation jälkeen arvioitiin, että kaikista vaatimuksista vain 5 voitaisiin testata automaatiolla. Toisen iteraation jälkeen automaatiotestien määrä kasvoi 11 toiminnallisuutta kehittävän tiimin työntekijöiden asiantuntemuksen takia. Automaatiotestien määrä kasvoi myös, koska ensimmäisessä iteraatiossa kaikki vaatimukset, joiden testausmenetelmästä oltiin epävarmoja, määriteltiin manuaalitesteiksi. Nämä epävarmat tapaukset tarkentuivat toisen iteraation aikana, jolloin ne voitiin muuttaa automaatiotesteiksi. Uudelle toiminnallisuudelle saatiin siis automaatio- ja manuaalitestejä, mikä oli haluttu tilanne.

Testauksen suunnittelun viimeisessä vaiheessa uudesta toiminnallisuudesta vastaavalle tiimille tehtiin vielä projektinhallintaohjelmistoon käyttäjätarinat (user story) uusien testien kirjoittamiseen. Jokaiselle vaatimukselle ei tarvinnut luoda omaa tarinaa, sillä joissakin tapauksissa vaatimuksia voitiin laittaa saman tarinan sisälle. Tällaisia tapauksia olivat esimerkiksi tilanteet, joissa eri laitteille oli tehty samankaltaisesta asiasta omat vaatimukset eri laitteisiin kuuluvien rajoituksien takia. Tarinoihin kirjoitettiin valmiiksi, millainen rakenne kussakin

testissä tulisi olla. Koska testien rakennetta mietittiin etukäteen, olisi uutta toiminnallisuutta kehittävän tiimin helpompi aloittaa toiminnallisuuden ohjelmistotestaus.

6 Yhteenveto

Insinöörityön tavoitteena oli kirjoittaa GE HealthCaren potilasmonitoreihin kehitettävälle uudelle toiminnallisuudelle ohjelmistotason vaatimusmäärittely sekä suunnitella vaatimusmäärittelyssä tulleille vaatimuksille ohjelmistotestaus. Vaatimusmäärittely ja ohjelmistotestaus ovat osa ohjelmistokehitystä. Lääkinnällisten laitteiden kohdalla vaatimusmäärittelyn ja ohjelmistotestauksen merkitys korostuu, koska lääkinnällisten laitteiden kehityksen vaiheet tulee olla jäljitettäviä. Potilasmonitorin kohdalla tämä on erityisen tärkeää, sillä monitorien avulla seurataan potilaiden elintoimintoja. Mikäli potilasmonitori ei toimi sille laadittujen vaatimuksien mukaisesti, voi potilasturvallisuus vaarantua. Ohjelmistotestauksen avulla laitteen valmistaja todistaa, että lääkinnällinen laite toimii valmistajan laatimien vaatimuksien määräämällä tavalla ja toteuttaa samalla lainsäädännön vaatimukset.

Ohjelmistotason vaatimusmäärittelyä varten yrityksellä oli pohjamateriaalia, jonka avulla vaatimuksia voitiin kirjoittaa. Näitä olivat muun muassa uudelle toiminnallisuudelle laaditut järjestelmätason vaatimukset sekä vanha toiminnallisuus, jonka uusi toiminnallisuus tulee korvaamaan. Ennen vaatimusmäärittelyn aloitusta, suunniteltiin tätä insinöörityötä varten prosessi, jonka mukaan vaatimukset laadittiin. Suunnitteluvaiheessa määriteltiin, että insinöörityön aikana valmistunut vaatimusmäärittely ei olisi lopullinen versio, sillä uutta toiminnallisuutta työstetään yrityksessä yhtenäistetyn ohjelmistokehitysmallin mukaisesti. Tästä seurasi, että kaikkia uuden toiminnallisuuden osa-alueita ei voitu insinöörityön aikana kirjoittaa vaatimusmäärittelyyn.

Uuden toiminnallisuuden ohjelmistotason vaatimusmäärittelyyn saatiin kirjoitettua 231 ei-toiminnallista vaatimusta eli DD:tä. Toiminnallisten vaatimuksien puuttuminen johtui pääosin siitä, että järjestelmätason vaatimukset olivat niin

kattavia eikä ohjelmistotasolle tarvittu erikseen toiminnallisia vaatimuksia. Insinöörityön alkuperäisenä tavoitteena oli tutkia, voisiko ohjelmistotason vaatimukset testata automaation avulla vai pitäisikö vaatimuksille kirjoittaa manuaalitestejä. Koska ohjelmistotasolle ei tullut lainkaan toiminnallisia vaatimuksia, päätettiin tutkia järjestelmätason vaatimuksia.

Testauksen suunnittelu toteutettiin katselmoimalla kaikki uuteen toiminnallisuuteen liittyvät järjestelmävaatimukset. Jokaisesta vaatimuksesta tehtiin arvio, voitaisiinko vaatimus testata automaation avulla vai tulisiko kirjoittaa manuaalitesti. Arvioita tehtiin kahdessa iteraatiossa, joiden lopputuloksena oli, että automaatiotestejä tulisi enemmän kuin manuaalitestejä. Insinöörityön alkuperäisen suunnitelman ulkopuolelta uudesta toiminnallisuudesta vastaavalle tiimille tehtiin jokaisen vaatimuksen osalta suunnitelma varsinaisten testien rakenteesta.

Insinöörityön tulokset vaatimusmäärittelyn ja ohjelmistotestauksen suunnittelun osalta auttavat yritystä pääsemään lähemmäksi uuden toiminnallisuuden julkaisemista. Seuraavaksi yritys voi keskittyä uuden toiminnallisuuden osalta ohjelmistokehityksen muihin vaiheisiin, joita ovat toiminnallisuuden suunnittelu esimerkiksi käyttöliittymän osalta ja itse toiminnallisuuden toteuttaminen. Samalla yritys voi vähitellen työstää myös ohjelmistotestausta.

Vaatimusmäärittelyn tekeminen ohjelmistokehityksen alussa on tärkeää, jotta lopputulos vastaa asiakkaan ja käyttäjien vaatimuksia. Vaatimusmäärittelyn tekeminen kattavasti auttaa ohjelmiston suunnittelijoita ja kehittäjiä heidän työssään, kun haluttu lopputulos on selkeä. Epäselkeä tai puutteellinen vaatimusmäärittely voi jättää ohjelmistoon avoimia kohtia, jotka voivat muuttaa valmiin ohjelmiston sellaiseksi, joka ei enää vastaa asiakkaan tai käyttäjien vaatimuksia ja toiveita. Vaikka tässä insinöörityössä keskitytään GE HealthCaren potilasmonitoreihin tulevaan uuteen toiminnallisuuteen, voidaan insinöörityössä toteutettua prosessia ohjelmistotason vaatimuksien kirjoittamiseen hyödyntää muutenkin lääkinnällisten laitteiden ohjelmistokehityksessä. Tärkeintä vaatimuksien kirjoittamisessa on se, että vaatimukset ovat ymmärrettäviä ja yksiselitteisiä kaikille ohjelmiston sidosryhmille. Ennen ohjelmistotason vaatimusmäärittelyn

aloittamista on kuitenkin tärkeä ottaa huomioon, että ylemmän tason vaatimukset ovat selvillä ja haluttu lopputulos on tiedossa.

Lähteet

1 GE HealthCare Completes Spin-Off and Begins Trading on Nasdaq. 2023. Verkkoaineisto. GE HealthCare. <https://www.ge-healthcare.com/about/newsroom/press-releases/ge-healthcare-completes-spin-off-and-begins-trading-on-nasdaq>. 4.1.2023. Luettu 23.2.2024.

2 Montevirgen, Karl. 2024. General Electric. Verkkoaineisto. Britannica. <https://www.britannica.com/topic/General-Electric>. Päivitetty 20.2.2024. Luettu 23.2.2024.

3 GE HealthCare Reports Fourth Quarter and Full Year 2023 Financial Results. 2024. Verkkoaineisto. GE HealthCare. <https://www.ge-healthcare.com/about/newsroom/press-releases//inserting-and-replacing-ge-healthcare-reports-fourth-quarter-and-full-year-2023-financial-results>. 6.2.2024. Luettu 23.2.2024.

4 GE Healthcare expands operations in Finland. 2022. Verkkoaineisto. Business Finland. <https://www.businessfinland.com/cases/2022/ge-healthcare-expands-operations-in-finland/>. 26.1.2022. Luettu 23.2.2024.

5 CARESCAPE Canvas. Mounting Options for Clinical Applications. 2022. Verkkoaineisto. GE HealthCare. <https://landing1.ge-healthcare.com/rs/005-SHS-767/images/45351-CARESCAPE-CANVAS-1-2-Quick-Reference-Guide-LP-PM.pdf>. Luettu 19.4.2024.

6 Aura, Suvi & Kinnunen, Tommi. 2022. Perioperatiivinen hoitotyö. 3., uudistettu painos. Helsinki: Sanoma Pro Oy.

7 Beginner Guide. Life Care Solutions. 2016. Yrityksen sisäinen aineisto. GE HealthCare.

8 Euroopan parlamentin ja neuvoston asetus lääkinnällisistä laitteista. 2017. Asetus 2017/745. Verkkoaineisto. Euroopan unionin virallinen lehti 5.5.2017. <https://eur-lex.europa.eu/legal-content/FI/TXT/HTML/?uri=CELEX:32017R0745&from=FI#d1e1046-1-1>. Luettu 2.3.2024.

9 How to Determine if Your Product is a Medical Device. 2022. Verkkoaineisto. U.S. Food & Drug Administration. <https://www.fda.gov/medical-devices/classify-your-medical-device/how-determine-if-your-product-medical-device>. 29.9.2022. Luettu 2.3.2024.

10 Classify Your Medical Device. 2020. Verkkoaineisto. U.S. Food & Drug Administration. <https://www.fda.gov/medical-devices/overview-device-regulation/classify-your-medical-device>. 7.2.2020. Luettu 2.3.2024.

11 Content of a 510(k). 2019. Verkkoaineisto. U.S. Food & Drug Administration. <https://www.fda.gov/medical-devices/premarket-notification-510k/content-510k>. 26.4.2019. Luettu 2.3.2024.

12 Saeed, Soobia; Jhanjhi, NZ; Naqvi, Mehmood & Humayun, Mamoona. 2019. Analysis of Software Development Methodologies. University of Bahrain. DSpace-julkaisuarkisto.

13 Wysocki, Robert K. 2013. Effective Project Management: Traditional, Agile, Extreme. E-kirja. ProQuest Ebook Central.

14 Leffingwell, Dean. 2007. Scaling Software Agility: Best Practices for Large Enterprises. E-kirja. O'Reilly.

15 Paakki, Jukka. 2011. Ohjelmistojen vaatimusmäärittely. Verkkoaineisto. Helsingin yliopisto. <https://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf>. Luettu 25.3.2024.

16 Parviainen, Päivi; Hulkko, Hanna; Kääriäinen, Jukka; Takalo, Juha & Tihinen, Maarit. 2003. Requirements Engineering. VTT Publications 508. Espoo: VTT.

17 ISO/IEC/IEEE 29119-1:2022. Software and systems engineering – Software testing – Part 1: General concepts. Revision of ISO/IEC/IEEE 29119-1:2013. IEEE.

18 Baresi, Luciano & Pezzè, Mauro. 2006. An Introduction to Software Testing. Electronic Notes in Theoretical Computer Science. Vol. 148, s. 89–111.