Salla Karhunen

Potilasmonitorien ohjelmistotestauksen menetelmien vertailu

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Hyvinvointiteknologia

Insinöörityö

11.5.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Salla Karhunen Potilasmonitorien ohjelmistotestauksen menetelmien vertailu 44 sivua 11.5.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Hyvinvointiteknologia
Suuntautumisvaihtoehto	Hyvinvointiteknologia
Ohjaaja(t)	Kehityspäällikkö, Kari Björn, Metropolia Ammattikorkeakoulu Ohjelmistokehittäjä, Ursa Hoikkala

Insinöörityön aiheena oli vertailla, onko potilasmonitorien ohjelmistotestauksessa havaittavissa eroja, kun testit suoritettiin automaatiolla ja manuaalisesti. Tavoitteena työssä oli saada selville, löytyykö näiden kahden eri testimenetelmän väliltä laadullisia eroavaisuuksia, erilaisia virheitä ja lisäksi, kuinka suuri ajallinen ero testien suorittamisessa on.

Insinöörityön teoriaa varten perehdyttiin automaation toimintaan, manuaaliseen testaamiseen, terveydenhuollonlaitteita koskeviin laadullisiin vaatimuksiin ja yrityksen käyttämiin standardeihin. Näiden lisäksi pohdittiin riskinhallintaa, millaisia virheitä ohjelmistotestauksessa syntyy helposti ja miten niitä pystyttäisiin välttämään.

Tässä insinöörityössä toteutettiin kuusi erilaista hälytystestiä, jotka oli jo valmiiksi suoritettu automaatiolla. Tässä työssä ne testattiin manuaalisesti. Tuloksista havaittiin, että täsmälleen samat virheet jäivät kiinni näillä molemmilla testimenetelmillä. Tuloksista ilmeni, että näiden testien suorittamiseen kului aikaa manuaalisella testimenetelmällä noin kolminkertaisesti verrattuna automaatiolla kuluneeseen aikaan.

Insinöörityön tuloksia voidaan hyödyntää, jos halutaan jatkaa näiden kahden eri testimenetelmän eroavaisuuksien selvittämistä. Tämä työ antaa tietoa terveydenhuollonlaitteita koskevista vaatimuksista. Sen avulla voidaan myös varautua mahdollisten virheiden varalle. Lisäksi tätä työtä pystytään mahdollisesti hyödyntämään uusien työntekijöiden koulutuksen tukena.

Avainsanat	Ohjelmistotestaus, automaatio, manuaalitestaus, laatu, virheet

Author(s) Title Number of Pages Date	Salla Karhunen Comparison of Methods for Testing Patient Monitor Software 44 pages 11 May 2016
Degree	Bachelor of Engineering
Degree Programme	Health Informatics
Specialisation option	Health Informatics
Instructor(s)	Kari Björn, Development Director, Metropolia University of Applied Sciences Ursa Hoikkala, Test Development Engineer

The subject of the study was to compare two different testing methods, i.e. automation and manual used in patient monitor software testing. The purpose of the study was to find out if there are some quality differences or different types of errors and to determine the time difference between these two testing methods.

The theoretical section explores automation and manual testing, qualitative requirements of medical devices and the standards the company is using. In addition, it covers risk-based testing, often-occurring errors and how to avoid errors.

In this study six different alarm tests were executed. The same tests had been earlier executed in automation and, in this study, they were now executed by manual testing. The findings show that the exact same errors were detected by these two different testing methods. The results also show that in manual testing the time to execute these six tests was triple compared to automation.

The results of this study can be used to continue the research to compare these two different testing methods. Additionally, this project gives information of the quality requirements for medical devices and how to prepare for possible errors. Moreover, it is possible to use this study as support in the training of new employees.

Keywords	Software testing, automation, manual testing, quality, errors

Sisällys

Lyhenteet

CD	Compact disc. Optinen digitaalisen datan ja äänen tallennusformaatti.
CE	Conformité Européenne. Vakuuttaa tuotteen täyttävän sitä koskevien direktiivien vaatimukset.
CEN	European Committee for Standardization. Eurooppalainen standardisoimisjärjestö, jonka tehtävänä on tuottaa eurooppalaisia standardeja.
DOORS	Rational Dynamic Object Oriented Requirements System. Vaatimusten hallintasovellus.
EKG	Elektrokardiogrammi. Sydänsähkökäyrä, joka on sydämen toimintaa kuvaava käyrä.
EN	European Standard. Eurooppalainen standardi.
EU	Euroopan unioni.
FDA	Food and Drug Administration. Yhdysvaltain elintarvike- ja lääkevirasto.
HR	Heart rate. Sydämen syke.
ID	Object identifier. Objektin tunniste.
ISO	International Organization for Standardization. Kansainvälinen standardisoimisjärjestö, jonka tehtävänä on tuottaa kansainvälisiä standardeja.
SpO_2	Happisaturaatio. Happikyllästeisyys, joka kertoo kudoksen happipitoisuuden suhteen sen suurimpaan mahdolliseen arvoon.
SPR	Software Problem Report. Ohjelmiston ongelmaraportti ClearQuest -työkalussa, luomalla SPR saadaan virheelle oma henkilökohtainen koodi.

1 Johdanto

Työn tilaaja on globaali yritys, joka vastaa sairaalalaitteiden tuotekehityksestä. Lisäksi yrityksessä kehitetään ja tuotetaan muun muassa potilasmonitoreja sekä niiden ohjelmistoja ja tarvikkeita. Potilasmonitorien avulla sairaaloissa, esimerkiksi tehohoidossa pystytään seuraamaan jatkuvalla mittauksella potilaiden EKG:tä (elektrokardiogrammi), verenpainetta ja muita parametreja. Näiden eri mittauksista saatujen tulosten perusteella pystytään tarkkailemaan ja tutkimaan potilaan tilaa sekä tilan mahdollisia muutoksia. Tämän insinöörityön molemmissa testiprosesseissa käytetään kohdeyrityksen valmistamia samaan tuoteperheeseen kuuluvia potilasmonitoreja, joista on olemassa kolme erilaista versiota. Näiden kolmen potilasmonitorin eroja ovat muun muassa niiden koko ja moduulipaikkojen määrät, moduulien avulla saadaan potilasmonitorin näytölle näkyviin erilaisia mitattavia parametreja.

Tämän insinöörityön tavoitteena on selvittää, miten paljon laadullisia eroavaisuuksia automaatiotestauksella ja manuaalitestauksella on potilasmonitorien ohjelmistotestauksessa. Testitapauksina tässä tutkimuksessa ovat riskien lieventämiseen liittyvät hälytystestit, joissa mitataan, aktivoituuko ja/tai eskaloituuko, sekä yhdessä testissä poistuuko, hälytys vaatimuksissa määritetyn ajan puitteissa. Testit ovat jo valmiiksi testattu automaattisella menetelmällä. Tässä tutkimuksessa ne testataan manuaalisesti, jotta voidaan samalla myös varmentaa, ettei manuaalisesti pystytä parempiin tuloksiin. Näissä molemmissa testiprosesseissa mitataan testaukseen kuluva kokonaisaika sekä tutkitaan, millaisia virheitä, näillä kahdella eri menetelmällä syntyy. Saatujen testituloksien avulla verrataan, saadaanko jommallakummalla menetelmällä laadukkaampia tuloksia. Testit suoritetaan yrityksen uusimmalla testeissä käytössä olevalla ohjelmistoversiolla, sillä sille on tehty myös automaatiotestit mahdolliseksi. Potilasmonitorin lisäksi testeissä käytetään potilassimulaattoria, EKG-kaapelia, SpO_2-parametria (happisaturaatio) ja sekuntikelloja.

Tämän insinöörityön tilannut yritys pysyy salaisena (Yritys X), koska kyseessä on tuotekehitykseen liittyvä tutkimus. Tämän takia mitään luottamuksellista tietoa ei julkaista eli esimerkiksi testeistä saatuja tarkkoja tuloksia ei voida julkistaa tässä tutkimuksessa.

2 Ohjelmistotestaus

Tässä luvussa käsitellään ohjelmistotestausta, jossa tarkastellaan, miksi ohjelmistoja testataan, sekä esitellään kolme ohjelmistokehitysmenetelmää, jotka on kehitetty suunnittelun ja toteutuksen tueksi. Nämä luvut toimivat perustana manuaali- ja automaatiotestaukselle, jotka käydään läpi kahdessa viimeisessä osassa tätä lukua. Manuaali- ja automaatiotestauksista tutkitaan niiden toimintaperiaatteita ja miten ne kohdeyrityksessä toimivat.

2.1 Ohjelmistotestaus yleisesti

Potilasmonitorien ohjelmistojen toiminnallisuuden testaamista vaatimuksien suhteen voidaan suorittaa sekä manuaalisesti että automatisoidusti. Lisäksi on olemassa sellaisia testitapauksia, joita voidaan suorittaa näillä molemmilla testimenetelmillä. Testien toteuttamisessa sekä manuaalisessa että automatisoidussa testiprosessissa apuna käytetään erilaisia moduuleja ja simulaattoreita, joiden avulla saadaan potilasmonitorin näytölle näkyviin testitapauksissa tarvittavia parametreja. Esimerkiksi EKG saadaan potilasmonitorin näytölle näkymään, kun potilasmonitoriin laitetaan kiinni oikeanlainen moduuli, johon kiinnitetään EKG-kaapeli, jonka mittauselektrodit puolestaan kiinnitetään potilassimulaattoriin, joka toimii testeissä oikean potilaan korvaajana. EKG:n ansiosta saadaan tietoja potilaan sydämen tilasta. Sillä nähdään esimerkiksi, mikä on potilaan sydämen syke. EKG-kaapeleita on olemassa erimääräisillä kytkennöillä, joita ovat esimerkiksi kolmikytkentäinen ja kymmenkytkentäinen versio. Kymmenkytkentäisessä versiossa sydänsähkökäyrällä on laskennallisesti havaittavissa 12 kytkentää, ja sillä saadaan tarkempia tuloksia sydämen toiminnasta verrattuna kolmikytkentäiseen mittaukseen. Kytkentöjen määrän valinta perustuu siihen, millaisessa tilanteessa EKG:tä tarvitsee mitata.

Ohjelmistoja testaan, seuraavista syistä:

- Löydetään mahdolliset virheet sekä poikkeavuudet ohjelmistoista ja testiproseduureista.
- Saadaan varmuus, että kaikki sen ominaisuudet varmasti toimivat niin kuin niiden on tarkoitus toimia.
- Saadaan varmistetuksi, että toteutettavasta ohjelmistotuotteesta tulee toivotun kaltainen.

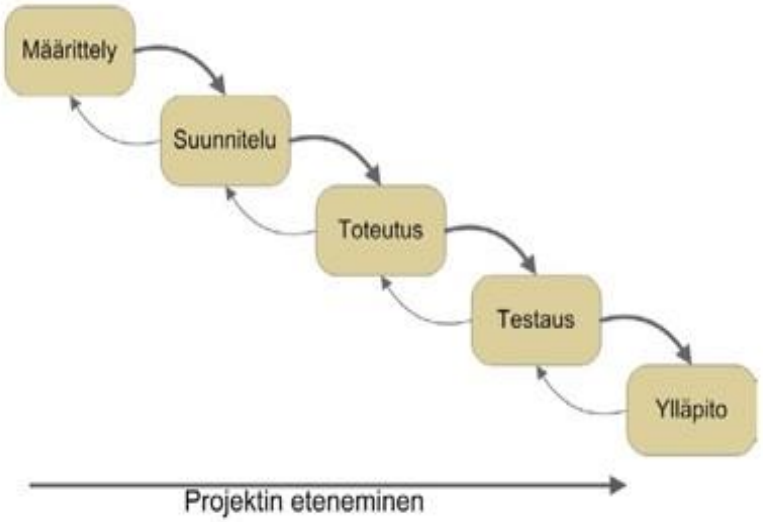
- Varmistetaan, että toteutettu ohjelmisto noudattaa sille tehtyjä määräyksiä.
- Varmistetaan, että ohjelmiston laatu vastaa sitä mitä sen pitää olla.
- Saadaan varmennettua ohjelmiston yhteensopivuus muiden järjestelmien kanssa.
- Saadaan estettyä ohjelmiston liian aikainen toimitus.

Periaatteessa testaustyötä voidaan pitää myös eräänlaisena vertailutehtävänä, sillä testauksen tarkoituksena on tarkastaa, mitä on ollut tarkoituksena tehdä, vastaa sitä, mitä on saatu tehdyksi. Lisäksi testeissä tulee tunnistaa sellaiset toiminnot, joissa saatu tulos poikkeaa suunnitelmista. Kaikki tulokset, joita testiprosesseista saadaan, kirjataan saadun tuloksen mukaisesti. Mikäli testattu testitapaus on ollut vaatimuksien mukainen, saadaan tulokseksi ”Pass”, mutta mikäli tulos hylätään eli jos se poikkeaa halutusta tuloksesta, kirjataan silloin tulokseksi ”Fail”. [1, s. 10; 2, s. 37-38.]

Kohdeyrityksellä on ohjelmistotestauksen vaatimustenhallintasovelluksena käytössä Rational DOORS -niminen työkalu. Virheiden seurantajärjestelmänä yrityksessä käytetään Rational ClearQuestia ja Rallya, jonne testien ohjelmistoversiosta riippuen kumpaan näistä kahdesta yllä mainitusta työkalusta, merkitään testeissä havaitut virheet. Periaatteessa jokaiselle löydetylle virheelle tulisi tehdä oma vikailmoituksensa, mutta esimerkiksi kaikki testin protokollassa olevat kirjoitusvirheet voidaan koota yhdeksi yhteiseksi virheilmoitukseksi. Kun nämä löydetyt virheet on saatu korjattua, niin testiprosessi suoritetaan jommankumman virheiden seurantasovelluksen kautta uudelleen, ja kirjataan sinne uudet testitulokset.

2.2 Ohjelmistotuotannon menetelmät

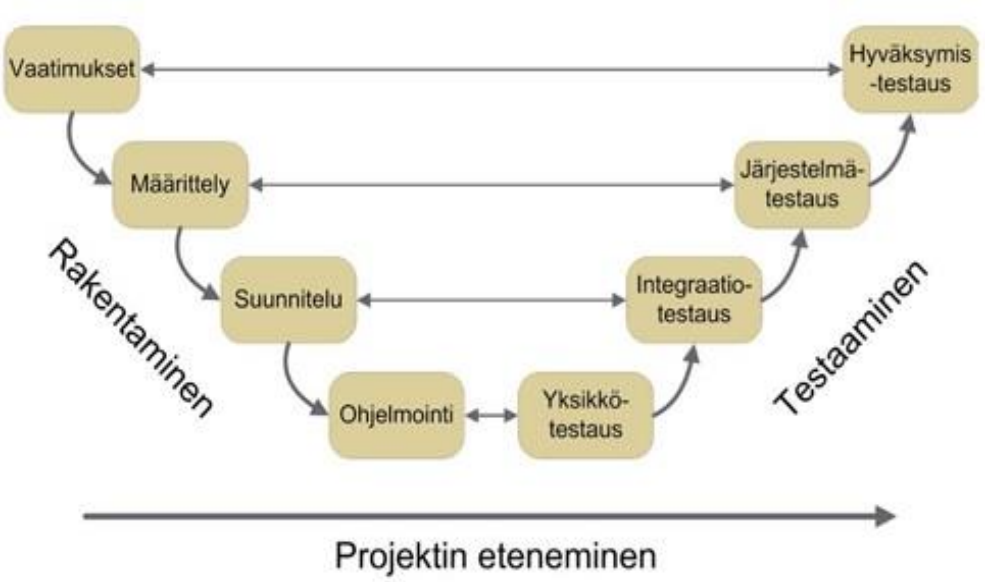
Tässä työssä esitellään kolme ohjelmistokehityksen prosessimallia, jotka ovat vesiputousmalli, V-malli ja Scrum-malli. Näistä kolmesta mallista on ensimmäisenä kehitetty vesiputousmalli (kuva 1), jonka ideana on se, että projekti etenee vaihe vaiheelta ja edelliseen työvaiheeseen palaamista on tarkoitus käyttää vain harvoin. Kuvassa 1 on esitetty, miten vesiputousmallin mukaan projekti etenee.



Kuva 1. Vesiputousmalli [1, s. 13.]

Kuvan 1 mukaan ohjelmiston tekeminen aloitetaan määrittelemällä sille tarvittavat vaatimukset. Kaikki saatavilla oleva tieto esimerkiksi kannattavuuslaskelmat, markkina-analyysi ja asiakastarpeiden alustavat tiedot kootaan taustatutkimukseksi. Tämän jälkeen määriteltyjä vaatimuksia käyttäen tehdään suunnitelma laitteen tai ohjelmiston rakenteista, niiden toiminnallisista vaatimuksista ja projektisuunnitelma. Projektisuunnitelmaa hyödyntäen ohjelmisto, sekä laitteisto kootaan ja sen jälkeen ohjelmisto toteutetaan. Kun toiminnallinen kokonaisuus on valmis laitteistoa, ja sen ohjelmistoa päästään testaamaan. Ohjelmiston tai laitteen vastatessa sitä mitä suunniteltiin, eli kun se täyttää asiakastarpeet, eikä sisällä enää virheitä, on testivaihe silloin valmis. Tämän jälkeen ohjelmisto on myös valmis käyttöönotettavaksi. [1, s. 12-13, 23.]

V-mallin ideassa (kuva 2) rakentamispuoli toimii oikeastaan samalla tavalla kuin vesiputousmallissa, mutta siinä toteutetaan enemmän testauksia. V-malli on esitetty vaihe vaiheelta kuvassa 2.



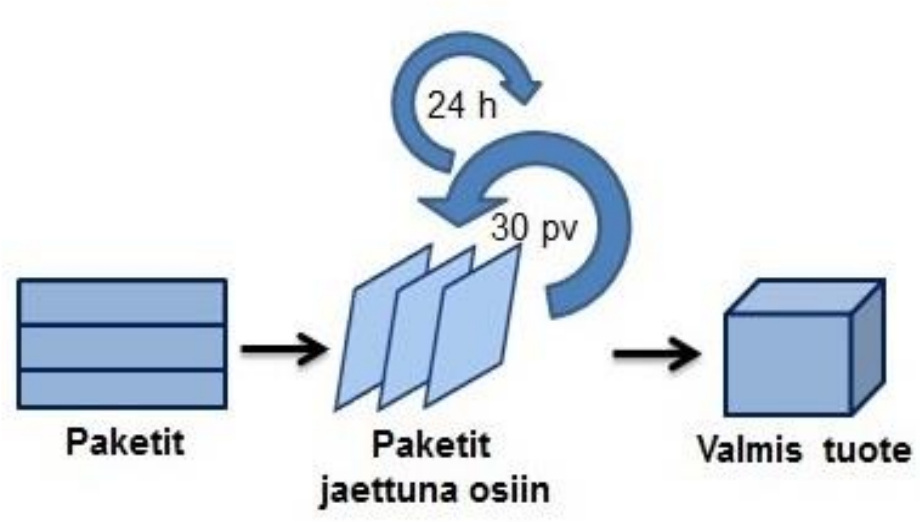
Kuva 2. V-malli [1, s. 14.]

V-mallin jokaisessa rakennusvaiheessa on myös oma testikategoriansa, eli ohjelmointityötä tarkistetaan tätä menetelmää käytettäessä erilaisilla testaustasoilla. Näitä tasotestejä ovat yksikkötestaus, integraatiotestaus, järjestelmätestaus ja viimeisenä hyväksymistestaus. Sen jälkeen, kun ohjelmisto läpäisee nämä kaikki tasot, on se valmis käyttöönotettavaksi. [1, s. 14; 3, s. 11.]

Yksikkötesteissä tarkastellaan yhden yksittäisen osan toimintaa samantien, kun se on toteutettu. Sen tarkoituksena on varmistaa, että kehitetty toiminto tai tehty muutos varmasti toimivat, eivätkä ne ole aiheuttaneet uusia virheitä ohjelmistoon. Integraatiotesteissä aletaan yksikkötesteissä testattuja eri osia sovittaa yhteen, jotta saataisiin ohjelmisto lopulta toimimaan yhtenä kokonaisuutena. Integraatiotesteissä testataan sellaisia testitapauksia, jotka ovat huomattavasti laajempia kuin yksikkötesteissä. Järjestelmätesteissä testataan ohjelmistoa, joka on koottu yhdeksi toiminnalliseksi kokonaisuudeksi. Tässä siis testataan, että järjestelmä varmasti toimii kokonaisuutena, ja se toteuttaa sille määritellyt tavoitteet. Nämä järjestelmätestien vaiheeseen kuuluvat testit suoritetaan testausympäristössä. Viimeisenä olevaa hyväksymistestausta suoritetaan ohjelmiston oikeassa kohdeympäristössä. Siinä järjestelmä tarkistetaan virallisesti, eli asiakas mahdollisesti hyväksyy tuotteen. Testaustyön jakamisella näihin eri osioihin saadaan muodostettua tehokkaampi ja tarkempi testiprosessi, sillä jokaisella tasolla voidaan perehtyä kyseisen alueen ominaisten virheiden löytämiseen. Samoin kuin useat muut ohjelmistotuotannon mallit, niin myös V-malli sisältää saman ongelman eli tes-

taaminen aloitetaan usein liian myöhään. Jos kyseessä on esimerkiksi vaatimusten-määrittely, niin luodaan se huomattavan paljon aiemmin kuin testaukset aloitetaan. [1, s. 14-15, 51.]

Yritys X:ssä ohjelmistotestauksen peruspohjana käytetään muun muassa juuri näitä molempia yllä esiteltyjä malleja. Silloin kun kehitetään kokonaan uusi ohjelmisto, niin sen koko toteutusprosessi tapahtuu osittain näiden yllä esiteltyjen mallien perusteella. Näitä automatisoituja järjestelmätestejä ajetaan yrityksessä läpi kuitenkin jatkuvasti koko kehityksen ajan, jo siis ennen varsinaista järjestelmätestausta. Nämä yllä esitellyt kaksi mallia ovat kuitenkin jo vanhahtavia malleja, ja yrityksessä käytetäänkin näiden lisäksi hyödyksi myös uudempia malleja, joista tässä tutkimuksessa esitellään Scrum-malli. Yritys X:ssä tätä mallia käytetään avuksi kehitysprosesseissa, mutta sen huonona puolena on se, ettei sillä kuitenkaan pystytä suorittamaan lopullista verifikaatiota. Tässä vaiheessa voidaankin siis hyödyntää V-mallia, koska yrityksessä tulee ennen tuotteen markkinoille saattamista olla verifikaatio, käytössä olevan laatujärjestelmän vuoksi. Tämän verifikaation avulla saadaan varmistettua potilasturvallisuus. Kuvassa 3 on esiteltynä Scrum-mallin toimintaperiaate.



Kuva 3. Scrum-malli [1, s.28.]

Scrum-malli on ketterä ohjelmistotuotannon menetelmä, jolla yritetään reagoida mahdollisimman nopeasti ilmestyviin ongelmatilanteisiin. Koska tämä malli on suhteellisen

yksinkertainen, sekä luonteeltaan helposti hallittava, se on myös yleistynyt hyvin yritysmaailmassa. Scrum-mallissa vaatimustenmäärittely tehdään esimerkiksi haastattelemalla asiakkaita. Sen jälkeen, kun toteutettavalle tuotteelle on tehty suunnitelma ja vaatimustenmäärittely, jaetaan tuotteen toteutus yhden iteraation eli lyhyiden työskentelyjaksojen kokoisiksi paketeiksi, jotka sitten toteutetaan pieni osa kerrallaan. Ohjelmistoa kehitetään tämän mallin mukaan aina muutaman viikon tai 30 päivän jaksoissa, jossa ohjelmistolle tehdään uusi määrä ominaisuuksia, sekä toimintoja, joita testataan ja samalla saadaan varmistettua, että kaikki varmasti myös toimivat. Kun kaikki paketit ovat valmiita, on tuote mallin mukaan silloin valmis, ja se voidaan toimittaa julkaistavaksi tai käyttöönotettavaksi. Scrum-mallin mukaan työryhmän keskeinen kommunikaatio on tärkeää, joten joka päivä tulee pitää palaveri, jossa käydään läpi tilanteita eli selvitetään mitä kukakin on saanut aikaiseksi, onko ilmennyt joitakin ongelmia, mitä pitää selvittää, sekä sovitaan ja jaetaan esimerkiksi viikon sisällä suoritettavat uudet tehtävät. [1, s. 27-29.]

2.3 Manuaalitestaus

Manuaalitestaus on testimenetelmä, jossa testin suorittajana toimii ihminen. Testaaja testaa ohjelmiston toimivuutta manuaalisesti, ilman automaattisesti toimivia välineitä. Testaajan on ymmärrettävä, millaisia vaatimuksia testillä on ja miten se tulee suorittaa. Manuaalitestauksen tärkeimpänä tavoitteena on varmistaa, että testattava ohjelmisto on mahdollisimman virheetön, ja se toimii niin kuin dokumentin vaatimuksissa sen sanotaan toimivan. Mikäli testissä havaitaan joitakin virheitä tai muita poikkeavuuksia saadun tuloksen ja odotetun tuloksen välillä, ne tulee merkitä virheiksi. Kun nämä raportoidut virheet ovat korjattu, testi suoritetaan uudelleen, jotta saadaan varmistettua, että aikaisemmin löydetyt virheet on varmasti saatu korjattua. Testimenetelmistä manuaalitestausta käytetään esimerkiksi silloin, kun testi tarvitsee suorittaa vain kerran tai kaksi. Kun ohjelmistoja testataan manuaalisesti testitapauksia ei välttämättä ole mahdollista suorittaa samalla tarkkuudella useaan kertaan. [4.]

Yritys X:ssä testin suorittajan tulee tarkistaa ennen testien aloittamista, että kaikkien laitteiden huoltopäiväykset ovat sallittujen rajojen sisällä. Yrityksen jokaisessa kalibroitavassa laitteistossa on tarra, joka kertoo laitteen laitetunnuksen, huoltopäivän ja sen päivän, milloin laite on viimeistään huollettava uudelleen. Manuaalitestauksessa testaajan tulee kirjoittaa testiraporttiin saatujen testitulosten lisäksi myös testissä käytettyjen

laitteiden tietoja, joita ovat muun muassa kaikkien käytössä olleiden laitteiden sarjanumerot, huoltopäivät ja testaamiseen kulunut aika. Potilasmonitorista tulee kirjata vielä näiden parin yllä mainitun kirjattavan asian lisäksi muutamia muita tietoja, kuten mikä ohjelmistoversio potilasmonitorissa on ollut käytössä testiä suorittaessa ja mikä ohjelmistopaketti on ollut käytössä. Ohjelmistopakettien avulla testejä pystytään suorittamaan eri testiympäristöissä, vaihtamalla kyseinen ohjelmistopaketti vastaamaan haluttua toimintaympäristöä. Potilasmonitorissa olevaa ohjelmistopakettia pystytään vaihtamaan siten, että se vastaa esimerkiksi teho-osastoa tai leikkaussalia. Suoritettavien testien testiympäristöjen on hyvin tärkeää vastata tulevien käyttäjien käyttöympäristöä, jotta suoritetut testit olisivat mahdollisimman totuudenmukaisia. Näiden erilaisten ohjelmistopakettien toimivuutta voidaan testata myös automaattisesti suoritettavissa testeissä.

Useat yrityksessä testattavista testeistä sijaitsevat DOORS-työkalussa, jonne kirjataan sekä käytössä ollut laitteisto, että saadut testitulokset. Työkalussa on muutamia kohtia, jotka kertovat testaajalle, mitä hänen tulee seuraavassa testikohdassa tehdä, sekä niiden lisäksi on olemassa testaajan kirjausta vaativia osioita. Tällaisia kirjattavia kohtia ovat ”Tester input”- ja ”Result” -kohdat. ”Tester input” eli testaajan syöte on kohta, johon testaajaa pyydetään kirjaamaan esimerkiksi, kuinka kauan koko testin tekemiseen on kulunut aikaa tai miten monijohtoinen EKG-kaapeli on käytössä. Toisena kirjattavana kohtana oleva ”Result” eli tulos on kohta, johon valitaan viidestä siinä olevasta eri vaihtoehdosta testin tila, esimerkiksi onko se hyväksytty vai hylätty (Pass/Fail) ja testaajan havaitsema tulos. Jokaisella kohdalla on DOORS-työkalussa myös oma ID-numeronsa (Object identifier), joiden avulla pystytään testeistä etsimään jotakin tiettyä testikohtaa. Yrityksessä suoritetaan myös sellaisia testejä, joista ei kuitenkaan suoriteta koko testiä, vaan vain esimerkiksi jokin tietty sen sisältämä testikappale. Tällöin ID-numero helpottaa testaajaa oikean kohdan löytämisessä.

2.4 Automatisoitu testaus

Automaatiotestaus on manuaalitestauksen automatisoitu muoto. Siinä ohjelmistojen testaukset suoritetaan koneellisesti jonkin testiohjelman avulla. Automaation avulla testiprosessit saadaan tehtyä nopeammin kuin manuaalisesti testaamalla ja testitapauksia voidaan helposti toistaa. Tällaista jatkuvaa uudelleentestaamista kutsutaan myös nimellä regressiotestaus, jonka avulla saadaan varmistetuksi, että kaikki tehdyt

muutokset tai korjaukset ovat poistaneet aikaisemmin havaitut virheet, mutta eivät kuitenkaan ole aiheuttaneet ohjelmistoon uusia virheitä. Oikeastaan isoimpana ajatuksena käyttää automaatiota hyödyksi testeissä onkin vapauttaa manuaalitestejä suorittavat testaajat muihin tehtäviin. Esimerkiksi mikäli jostain ohjelmistosta tulee jokaisena vuorokautena uusi versio, ei kustannusten ja ajankäytön kannalta ole järkevää käyttää testaajien aikaa täsmälleen samojen perustestien tekemiseen. Nämä automatisoidut testit voivat olla myös eräänlaisia tarkistuksia, joita tietokone jätetään yön yli tekemään. Tällaisissa tapauksissa ohjelmistokehittäjät pystyvät sitten seuraavana työpäivänään käymään läpi näiden tarkastuksien tulokset ja korjaamaan mahdollisesti havaitut virheet uudempaan ohjelmistoversioon. [1, s. 76; 3, s. 19.]

Testausautomaatiota voidaan monesti pitää manuaalitestauksen korvaajana, mutta todellisuudessa se vain täydentää sitä. Automaatio ei poista manuaalitestauksen tarvetta. Esimerkiksi kun testaaja tekee testiprosessia manuaalisesti, testin suorittamista voidaan muuttaa missä vaiheessa tahansa ja tarkkailla miten ohjelmisto siinä tapauksessa toimii, kun taas automaatiotestauksella automaatio-työkalu tekee vain sen, mitä testin läpiajaja on laittanut sen tekemään. Toisaalta testin tekeminen automatisoituun muotoon on kalliimpaa kuin sen testaaminen manuaalisesti, mutta puolestaan automaatiolla testien toistaminen on edullisempaa. Testi kannattaa siis tehdä automatisoituun muotoon, mikäli sitä tarvitsee toistaa useamman kerran. [1, s. 76-77; 2, s. 366.]

Yritys X:ssä voisi automaatiolla suoritettavissa testeissä olla käytössä esimerkiksi jonkinlainen robottisormi, joka painaisi potilasmonitorin näyttöä aina sen perusteella, miten se on ohjelmoitu potilasmonitorin ohjelmistoa testaamaan. Tällaisen robottisormen avulla yrityksessä suoritetuissa automaatiotesteissä saataisiin testattua potilasmonitoreita konkreettisemmin. Kun potilasmonitoreita testataan vain käskyjen perusteella, niin näytöllä olevat valikot avautuvat/sulkeutuvat silloin vain käskyjen mukaisesti, ilman että niihin oikeasti kosketaan.

Automaatiotestauksessa Yritys X:llä on käytössä Robot Framework -työkalu, jonka avulla voidaan suorittaa hyväksyntätason testejä ja hyväksyntätestausohjattua kehittämistyötä. Työkalu mahdollistaa käyttäjän luomaan uusia korkean tason avainsanoja jo olemassa olevista avainsanoista. Testit saadaan rakennetuksi, kun näitä avainsanoja yhdistellään, jolloin saadaan luotua hyvin laajoja avainsanasarjoja. Robot Framework on avoimen lähdekoodin ohjelma, joka on julkaistu Apache Licence 2.0 -lisenssin alaisena, ja sen tekijänoikeudet omistaa Nokia Networks. Sen testiominaisuuksia voidaan

toteuttaa joko Python- tai Java -tekstikirjastoilla. Yrityksellä on Robot Frameworkin apuna käytössä Jenkins-niminen jatkuvan integroinnin työkalu, joka toimii testien suorittamisen hallinnassa ja tulosten julkaisussa. Tämän Jenkins-työkalun avulla saadaan siis Robot Framework ajamaan testit läpi. [5.]

Yritys X:ssä automaatiotestauksessa testiin ilmoitettavaksi tarvittavat laitteistojen tiedot saadaan haettua automaattisesti erilaisista jo valmiiksi olemassa olevista kalibraatiotietokannoista. Tämä mahdollistaa sen, ettei testin läpiajajan tarvitse kirjoittaa niitä käsin testin alkuun. Mikäli tietokannasta kuitenkin puuttuu jonkin laitteen tiedot, testaajan tulee siinä tapauksessa lisätä ne sinne itse. Useimmiten nämä tarvittavat tiedot voidaan lisätä jo olemassa olevaan tietokantaan, jolloin kirjausprosessi myös nopeutuu jonkin verran.

3 Laadulliset vaatimukset

Luvussa 3 käsitellään laadullisia vaatimuksia, joita lääkintälaitteille on määritelty. Ensin tarkastellaan, mitä käsite laatu tarkoittaa, jonka jälkeen käsitellään standardeja, joita Yritys X käyttää. Viimeisessä osassa tätä lukua tarkastellaan virheitä, joista ensin käsitellään yleisiä ohjelmistotestauksessa tapahtuvia virhetilanteita, jonka jälkeen keskitytään siihen, miten näitä mahdollisia virheitä voitaisiin välttää. Näiden lisäksi tarkastellaan riskienhallintaa.

Laadunhallintajärjestelmäkseen Yritys X on valinnut ISO 9001:2008 -laatustandardin, sekä terveydenhuollon laitteille ja tarvikkeille määritetyt ISO 13485:2003- ja EN ISO 13485:2012 -standardit. Näiden standardien lisäksi yritys noudattaa lääkinnällisille laitteille määritettyä direktiiviä 93/42/EEC. Jokainen yrityksen käyttämä laatujärjestelmä vaatii hyvien dokumentointitapojen noudattamista, jotta asiakirjojen aitous ja oikeellisuus voidaan varmistaa.

3.1 Laatu

Yrityksen omassa dokumentissa määritelmä käsitteelle laatu on yhtä kuin potilasturvallisuus. Yritys X:n laatupolitiikan mukaan yritys on sitoutunut varmistamaan asiakkaiden tyytyväisyyden, parantamaan jatkuvasti toimintaa sekä noudattamaan lakeja ja määräyksiä. Yrityksessä ollaan sitouduttu noudattamaan laatua ja vaatimuksienmukaisuutta

jokaisessa päätöksessä, joita yrityksessä tehdään. Uskotaan, että se vaikuttaa potilaisiin, asiakkaisiin ja itse yritykseen. Jokaisen työntekijän tuleekin siis tavoitella tinkimätöntä asiakastyytyväisyyttä, jokaisessa kehitettävässä ja valmistettavassa tuotteessa sekä palvelussa. Ohjelmistotuotannossa laadulla tarkoitetaan periaatteessa täysin samaa asiaa kuin missä tahansa muussakin tapauksessa, eli lopputuloksen tulee täyttää toivotut odotukset ja sille asetetut vaatimukset. [1, s. 132.]

Terminä laatua on kuitenkin suhteellisen vaikea määritellä. Se voi muun muassa tarkoittaa tuotteen teknistä laatua (erinomainen, hyvä, huono), tuotteen hinnan asettamaa odotettua laatua tai asiakkaan tarvitsemaa laatua. Yritys X:n ohjelmistotestauksessa tulee aina pyrkiä parhaaseen laatuun, aluksi kehitystyövaiheessa toteuttamalla paras mahdollinen ohjelmisto ja sen jälkeen varmistamalla, että ohjelmisto toimii oikein, etenkin riskianalyysin perusteella potilasturvallisuuteen vaikuttavien toimintojen osalta. Lisäksi ohjelmistoa testaamalla pystytään varmistamaan, että ohjelmisto on tasalaatuinen. Tällä periaatteella saadaan varmistettua, että potilasmonitorit varmasti toimivat niin kuin niiden kuuluukin, jolloin saadaan myös taattua potilasturvallisuus potilasmonitorien käyttöympäristöissä. Potilasturvallisuuden edistäminen on yleisestikin ajatellen yksi tärkeistä laadunhallinnan osa-alueista. Se on myös sen olennaisena tavoitteena. Jotta potilasturvallisuutta saadaan kehitettyä entisestään, tulee inhimillisten virheiden mahdollisuus ymmärtää ja pyrkiä niiden minimoimiseen, lisäksi tehdyistä virheistä oppiminen ja järjestelmälähtöinen ajattelu vaikuttavat myönteisesti potilasturvallisuuteen. [1, s. 132-133; 6.]

Valmistettaville ohjelmistotuotteille on luokiteltu viisi yleistä laatuattribuuttia, jotka ovat sen verran yleisiä, että ne sopivat kaikenlaisille ohjelmistotuotteille. Määritykset ovat luotettavuus, tehokkuus, turvallisuus, ylläpidettävyys ja koko. Luotettavuudella tarkoitetaan, että ohjelmisto toimii ilman katkoksia, ongelmia tai riskitekijöitä, jolloin se myös antaa asiakkaille edustavan kuvan laitteistosta. Tehokkuudella tarkoitetaan tässä tapauksessa, että ohjelmisto pystyy suorittamaan sille määritetyt tehtävät mahdollisimman lyhyessä ajassa tai se pystyy käsittelemään suuria määriä sille annettuja syötteitä annetussa ajassa. Puolestaan turvallisuudella tarkoitetaan, että järjestelmällä on kyky torjua mahdollisia hyökkäyksiä tai murtautumisyrityksiä, jotka hyväksikäyttävät esimerkiksi laitteiston tunnettuja heikkouksia. Ylläpidettävyydellä tarkoitetaan, että ohjelmistoa pystytään siirtämään järjestelmältä toiselle ja se mukautuu uusiin tilanteisiin ja tapauksiin, joissa sitä käytetään. Koko tarkoittaa, että ohjelmiston täytyy olla kooltaan sen

tarpeisiin nähden järkevä ja sen pitää olla kehitetty sellaisessa projektissa, joka vastaa kokoluokaltaan tuotetun ohjelmiston realistisia tarpeita. [1, s. 137.]

Laadukkaan tuotantoprosessin tuloksena lopputuotteen tulee valmistua sille määritellyn aikataulun ja budjetin mukaisesti. Jotta lopputuotteelle asetetut tavoitteet saavutetaan käytettävissä olevien resurssien puitteissa, tulee prosessin vaiheita seurata, arvioida ja mitata. Laadunvarmistuksen yhteydessä käytetään termejä verifiointi ja validointi, joiden avulla kuvataan menettelytapoja tuotteen ja tuotantoprosessin laadun arvioinnissa. Periaatteessa nämä termit vastaavat myös kahteen kysymykseen: onko ohjelmisto tehty oikein ja onko tehty oikea ohjelmisto. Verifiointi tarkoittaa sellaisia toimenpiteitä, joilla yritetään vahvistaa, että tuote on tehty suunnitelmien mukaisesti ja se täyttää sille määritetyt toiminnot oikein. Puolestaan validoinnin avulla pyritään varmistamaan, että rakennettu ohjelmisto on oikeanlainen, joka täyttää sille ennalta määritellyt vaatimukset ja täyttää myös asiakkaan tarpeet. On äärimmäisen tärkeää, että kaikki toiminnot laitteissa toimivat kuten niiden pitäisikin toimia, sillä terveydenhuollossa asiakas joutuu luottamaan esimerkiksi leikkaukseen mennessään, että siellä olevat laitteet toimivat. Tämän takia Yritys X:ssä painotetaankin riskilähtöisyyteen. [1, s. 135; 3, s. 6.]

Yritys X:n tulee noudattaa standardien lisäksi myös lääkinnällisille laitteille määriteltyä 93/42/EEC -direktiiviä, joka pakottaa Euroopan talousalueen jäsenmaat säätämään paikalliset lait yhdenmukaisiksi, jotka sitten takaavat 93/42/EEC-direktiivin mukaisten CE-merkittyjen (Conformité Européenne) tuotteiden vapaan liikkumisen jäsenvaltioiden alueella. Kun laite täyttää sitä koskevan direktiivin asettamat vaatimukset, myönnetään laitteelle CE-merkintä, jonka on oltava laitteessa kiinnitettynä ennen kuin laite voidaan viedä markkinoille ja ennen sen käyttöönottamista. CE-merkintä on ainoa merkintä, jonka avulla nähdään, että valmistaja on varmistanut laitteen täyttävän kaikkien sitä koskevien direktiivien vaatimukset ja käynyt läpi tarpeelliset arviointimenettelyt. Yritys X voi kiinnittää vaatimuksenmukaisuuden merkiksi tuotteisiinsa CE-merkin, sillä sen kehittämät ja valmistamat tuotteet on sekä suunniteltu että testattu täyttämään turvallisuuden ja suorituskyvyn minimikriteerejä määräämät standardit. [7, s. 38, 55; 8, s. 4, 10, 15.]

Yrityksen omassa dokumentissa kerrotaan, että tämä yllä mainittu 93/42/EEC-direktiivi sisältää lääkintälaitteille yhteiset luokittelukriteerit sekä olennaiset vaatimukset. Se jakaa lääkinnälliset laitteet neljään eri luokkaan, jotka ovat I, IIa, IIb ja III. Tämä luokitus perustuu siihen, että mitä korkeampi luokka laitteella on, sitä riskialttiimpi myös tuote

on. Yritys X:n tuottamien potilasmonitorien riskiluokitus on toiseksi vaativin luokka eli IIb, koska niillä monitoroidaan myös elintoimintojen kannalta kriittisiä parametreja. [7, s. 38.]

Yrityksen täytyy olla varma, että heidän työntekijänsä ovat tietoisia siitä, miten tärkeää ja merkityksellistä heidän työskentely- ja toimintatapansa ovat parhaan mahdollisen laadun saavuttamiseksi. Kun yrityksen jokainen työntekijä sitoutuu joka päivä ylläpitämään laatua, niin se antaa silloin yrityksen asiakkaille ja potilaille varmuuden siitä, että yrityksen kehittämät ja valmistamat tuotteet sekä palvelut ovat turvallisia ja tehokkaita käyttää. Laatuongelmia voi ilmaantua, jos ollaan huolimattomia tai mikäli työntekijä on tietämätön oman työnsä vaatimuksista.

3.2 Standardit

Standardien tarkoituksena on helpottaa elinkeinoelämän, viranomaisten sekä kuluttajien elämää. Standardien avulla saadaan parannettua tuotteiden turvallisuutta ja suojella niiden käyttäjiä sekä varmistettua, että tuotteet ja ohjelmistot toimivat yhdessä. Lisäksi niillä varmistetaan, että tuotteet sopivat siihen käyttöön ja käyttöympäristöön, joihin ne on tarkoitettu. Niissä olevien kirjainyhdistelmien perusteella selviää, missä organisaatiossa standardi on vahvistettu. CEN (European Committee for Standardization) on eurooppalaisen standardisoimisjärjestön lanseeraaman standardin tunnus. Itse standardin tunnus on kuitenkin EN (European Standard). Puolestaan ISO (International Organization for Standardization) on kansainvälisessä standardisoimisjärjestössä julkaistun standardin tunnus. Esimerkiksi jos standardin tunnus olisi EN ISO, niin tarkoittaisi se silloin, että sama standardi on voimassa sekä Euroopassa että kansainvälisesti. [9; 10.]

3.2.1 ISO ja FDA

Yrityksen oman dokumentin mukaan kaikkien lääkinnällisten laitteiden valmistajien, kuten siis myös Yritys X:n on kehitettävä oma laatujärjestelmänsä kansainvälisen standardijärjestö ISO:n tuottamien standardien pohjalta. Koska Yritys X on globaalisti toimiva yritys, vaatimuksia laadulle määrittelee ISO:n lisäksi FDA (Food and Drug Administration), joka säätelee lääketieteellisten laitteiden markkinoita Yhdysvalloissa. FDA:n tavoitteena on suojella amerikkalaisia potilaita sekä kuluttajia. Yrityksen toiminnasta

FDA haluaa tarkistaa laatujärjestelmän toimivuutta ja olla varma siitä, että sitä myös noudatetaan. Lisäksi se seuraa, miten tehokkaita, aikataulussa pysyviä ja valmiita prosesseja yrityksellä on. Nämä sisältävät esimerkiksi valitukset, testeissä havaitut virheet ja tuotannossa ilmaantuneet viat. Mikäli esimerkiksi jostain testistä saatu tulos on jäänyt dokumentoimatta, sitä ei FDA:n mukaan ole edes tehty. Viranomaisten mielestä huolellisesti suoritettu dokumentointi on siis tärkeässä osassa tuotteen konkreettista laatua ajatellen. FDA:lla on lisäksi oikeus tulla tarkastamaan, missä ja milloin tahansa rekisteröityneet yritykset: tällaiset tarkastukset voidaan joko ilmoittaa etukäteen, tai ne voivat olla yllätyskäyntejä.

Valmistajille FDA:sta on tullut viranomainen, jota saatetaan jopa pelätä. Tällä viranomaisella on erittäin paljon valtaa valmistajiin, jotka vievät Yhdysvaltoihin laitteita, sillä siellä on harvinaisen vaativat lakisääteiset määräykset. Lääkinnällisten laitteiden vaatimuksiin on Yhdysvalloilla omia standardointiorganisaatioita, vaikka FDA hyväksyy myös omien standardien lisäksi monia kansainvälisiä standardeja. Esimerkiksi standardi ISO 13485:2003 on Yhdysvalloissa hyväksytty standardi, mutta pelkkä sen mukainen laadunhallintajärjestelmä ei kuitenkaan ole riittävä. [7, s. 83, 85.]

Yritys voi saada ISO-laatujärjestelmäsertifikaatin auditoinnin perusteella, joita yrityksen tulee järjestää säännöllisin aikavälein. Sertifioija on ulkopuolinen, pätevä ja puolueeton taho, joka varmistaa, että yritys toimii sertifioitavan standardin mukaisesti ja yrityksen laadunhallinta vastaa oikean standardin määrittelemiä vaatimuksia. Sertifioidun laatujärjestelmän avulla yritys pystyy osoittamaan asiakkailleen perusasioiden olevan kunnossa ja tuotteidensa olevan korkealaatuisia. Tulosten oikeellisuutta laatusertifikaatin avulla ei kuitenkaan pystytä takaamaan. [1, s. 135; 7, s. 75; 11, s. 96.]

3.2.2 ISO 9001 ja ISO 13485

Yritys X:n käyttämien kahden standardin ISO 9001 ja ISO 13485 keskeisenä erona on se, että ISO 13485 -standardin päämääränä on vaatimustenmukaisuuden varmistaminen. Yrityksen on siis pystyttävä luotettavasti osoittamaan, että heidän kehittämät ja tuottamat laitteet varmasti täyttävät niille määritetyt vaatimukset. Puolestaan ISO 9001 -standardi korostaa jatkuvaa parantamista. Tämän lisäksi näiden standardien erona on se, että standardi ISO 13485 on tarkoitettu käytettäväksi terveydenhuollon laitteille ja tuotteille, jonka takia siinä on jotain lisäyksiä ja jonkin verran poistettuja asioita ISO 9001 -standardiin verrattuna. [7, s. 70.]

Kansainvälinen ISO 9001 -laadunhallintastandardi asettaa organisaation laadunhallintajärjestelmälle vaatimukset. Standardi on maailman tunnetuin työkalu laadunhallintajärjestelmän rakentamiseen ja myös sen kehittämiseen. Sen avulla saadaan lisättyä luottamusta tuotteen ja palvelun vaatimustenmukaisuuteen sekä parannettua asiakastyytyväisyyttä. Laadunhallintajärjestelmä sisältää kaikki toiminnot, joita yritys tarvitsee saavuttaakseen tavoitteensa, esimerkiksi prosessit ja resurssit. ISO 9001 -standardi on terveydenhuollossa laajasti käytössä, mutta yksinään tämä standardi ei kuitenkaan riitä terveydenhuollon laitteiden ja tarvikkeiden laadunhallintajärjestelmän vaatimustasoksi. [7, s. 66; 12.]

ISO 9001:2015 on syksyllä 2015 uudistettu versio standardista ISO 9001:2008. Yrityksillä on kolme vuotta aikaa muuttaa laadunhallintajärjestelmänsä vastaamaan tätä uutta standardia. Uudistuneessa standardissa painotetaan laadunhallintajärjestelmän suunnittelun ja käytön kokonaisvaltaisuutta suhteessa toimintaympäristöön ja liiketoimintaan. Siinä myös korostetaan, että johdon tulee sitoutua ja osallistua laatujohtamiseen sekä ottaa siitä vastuu. Lisäksi uusitussa standardissa lähtökohdaksi on otettu riskilähtöisyys, jonka avulla yritetään välttää epätoivottuja lopputuloksia. [13.]

Laadunhallintastandardi ISO 13485 on tuotettu terveydenhuollon laitteiden ja tarvikkeiden laatujärjestelmää varten, ja se on suunnattu etenkin laitevalmistajille. ISO 13485 -standardi perustuu ISO 9001 -standardiin, mutta se on kuitenkin itsenäinen standardi. Kyseisen standardin vaatimustaso on korkeampi kuin ISO 9001 -standardin, koska sillä varmistetaan terveydenhuollon laitteissa olevat erityispiirteet, kun rakennetaan ja ylläpidetään laadunhallintajärjestelmää, sekä huomioidaan viranomaismääräykset, joita terveydenhuollon laitteilla on. Toisaalta koska ISO 13485 -standardista puuttuu joitakin piirteitä mitä ISO 9001 -standardissa on, joten yritys, joka täyttää ISO 13485 -standardin vaatimukset, ei välttämättä kuitenkaan omaa ISO 9001:n mukaista laatujärjestelmää. ISO 13485 -standardista on tällä hetkellä kansainvälisesti voimassa oleva ISO 13485:2003 ja EU:ssa EN ISO 13485:2012. Näiden kahden version erona on vain se, että EU-versio sisältää liitteen Z, jossa kerrotaan, millä tavoin standardi kattaa direktiivien laadunhallintajärjestelmävaatimukset. Järkevintä on käyttää terveydenhuollon laitteiden omaa laadunhallintajärjestelmä EN ISO 13485:2012 -standardia, silloin kun tuoteluokka vaatii laadunhallintajärjestelmää. [7, s. 32, 66-67, 70.]

Näiden kahden voimassa olevan version lisäksi ISO 13485 -standardista on nyt juuri julkaistu maaliskuussa 2016 uusi versio, joka on ISO 13485:2016. Se on tällä hetkellä

saatavissa englanninkielisenä, ja sen suomennos tullaan julkaisemaan kesäkuussa 2016. ISO 13485:2016 on kehitetty vastaamaan viimeisimmän laatujärjestelmän käytäntöjä, joka sisältää muutokset teknologian ja lainsäädännön vaatimuksissa sekä odotuksissa. Lisäksi tässä uudistuneessa versiossa on painotettu enemmän riskienhallintaan ja riskeihin perustuvaa päätöksentekoa. Koska Yritys X:n kehittämät potilasmonitorit kuuluvat riskiluokitukseltaan toiseksi korkeimpaan luokkaan (IIb), on standardin ISO 13485 mahdollistama perusteellinen laadunhallintajärjestelmä yritykselle pakollinen. [14; 15.]

ISO 13485 -standardia voidaan hyödyntää tuotteiden ja niihin mahdollisesti liittyvien erilaisten palveluiden suunnittelussa sekä kehittämisessä. Tämän lisäksi sitä on mahdollista käyttää tuotannossa, laitteiden asennuksessa ja myös sen jälkeen, kun tuote on saatu markkinoille. Tuotekehitykseen tämä standardi vaatii tarkkoja menettelyohjeita ja tuotteen kehittämistä määriteltyjen ohjeiden mukaisesti. Jokaisen tuotekehitystyövaiheen tulisi sisältää erilaisia katselmuksia, joissa seurataan muun muassa laitteen verifioinnin ja validoinnin tuloksia. Erittäin tärkeänä katselmuksena pidetään etenkin loppukatselmusta, joka kattaa tuotekehityksen kaikki tulokset. [7, s. 70, 73.]

3.3 Virheet

3.3.1 Virheet testauksissa

Ohjelmistotestauksessa vastaan voi tulla monia erilaisia virhetilanteita missä vaiheessa tahansa ohjelmistokehitystä. Yleisiä ohjelmistotestauksessa havaittuja virheitä ovat seuraavat tilanteet:

- Ohjelmistossa on havaittavissa jokin virhe, jolloin testiprosessi ei toimi niin kuin vaatimuksien mukaan sen kuuluisi toimia.
- Käynnistymisen keskeytyminen on virhetila, jossa laitetta ei saa kunnolla käyntiin. Jossain tapauksissa voi näyttää siltä, että ohjelmisto käynnistyy normaalisti ilman virheilmoituksia, mutta jokin sen toiminnoista jää silti puutteelliseksi.
- Pysähtymisvika on tilanne, jossa laitetta ei saa sammutettua tai pysäytettyä tarvittaessa ja kaatumisesta puhutaan silloin, kun ohjelmisto ei pysy päällä vaan sammuu yllättäen.
- Datavika on tapahtuma, jossa jokin tieto näytöllä on vääränlainen.

- Tilaviassa laite toimii välillä oikein, mutta välillä väärin.
- Katkos on tilanne, jossa kahden laitteen välillä tieto ei kulje, esimerkiksi Yritys X:llä tämä voisi olla tilanne, jossa potilasmonitoriin ollaan vaihtamassa toista ohjelmistoversiota ja se ei päivity.
- Virhesyötteitä ovat kirjoitusvirheet, puuttuva data ja väärinkäsitykset.

Yllä lueteltujen virhetilanteiden lisäksi virheitä voi aiheutua, jos testaajalla on paineita ajankäytön suhteen. Lisäksi virheitä syntyy, koska testaaja on ihminen ja ihmiset tekevät virheitä. Yrityksen kaikkien työntekijöiden onkin tärkeää ymmärtää, että yksi potentiaalinen mahdollisuus virheisiin syntyy, silloin jos oma työskentelytapa on väärä. [16, s. 23-24; 17; 18.]

Ohjelmistoista löytyvät virheet voidaan jakaa erilaisiin luokkiin niiden alkuperän, vaikutusten ja vakavuuden perusteella. Tietoja voidaan myöhemmin hyödyntää sekä testauksen että korjaustoimenpiteiden priorisoinnissa ja uusien testitapausten suunnittelussa. Ohjelmistoissa olevien virheiden vakavuus riippuu siitä, kuinka usein virhe esiintyy, miten paljon sen korjauskustannukset tulevat olemaan ja mitä seuraamuksia se on aiheuttanut. Pieniksi ja vähäisiksi virheiksi luokitellaan sellaiset, joiden vaikutukset ovat paikallisia ja niitä on helppo korjata: tällaisia ovat esimerkiksi kirjoitusvirheet tai tarpeettomat tulosteet. Vakavien virheiden aiheuttamat seuraamukset eivät rajoitu yksittäisiin tapauksiin. Monesti virhetilanteista aiheutuvat ongelmat ovat lisääntyviä, jolloin niiden seuraukset ja kustannukset myös suurenevat. Yritys X:ssä tällainen kriittinen virhetilanne voisi pahimmassa tapauksessa aiheuttaa vakavan uhan potilaan turvallisuudelle, mikäli se tapahtuisi potilasmonitorien käyttöympäristöissä. Testaajalle virheiden vaikutukset eivät aina tule näkyviin, sillä jonkin toisen toiminnon tai virheen seurauksena virhe saattaa peittyä tai kumoutua. [3, s. 8, 27.]

Virheet voidaan jakaa myös toisella tavalla kahteen eri luokkaan, näkyviin virheisiin ja näkymättömiin virheisiin. Yrityksen omassa dokumentissa mainitaan, että näkyviä virheitä ovat sellaiset viat, joita on helppo havaita ja ne sisältävät poikkeamia virheiden kirjauksessa tai menettelytavoissa. Näkyviä virheitä ovat muun muassa tilanteet, joissa testilaitetta käytetään, vaikka sitä ei olisi huollettu määräaikaan mennessä tai suoritetusta testitapauksesta puuttuu tietoja, kuten esimerkiksi SPR (Software Problem Report) -numero väärintoimineen testikohdan havaittu testitulos kohdasta. Näkymättömiä virheitä ovat sellaiset, jotka eivät ole yhdenmukaisia eivätkä ne ole helposti nähtävissä, mutta joilla on kuitenkin merkittävä laatua säätelevä vaikutus. Tällaisia näkymättömiä

virheitä ovat esimerkiksi tilanteet, joissa testaaja ei havaitse laitteessa poikkeavaa ominaisuutta tai verifikaation ja validaation tuloksia tulkitaan väärin.

Testauksessa löytynyt virhe kertoo ohjelmiston ongelmasta, mutta virheen alkuperäistä syytä tai tarkkaa sijaintia ei välttämättä saada selville. Testaajan tulee dokumentoida havaittu virhe tai poikkeavuus, jonka jälkeen vastuu löydetyn virheen korjauksesta ja paikantamisesta siirtyy ohjelmistokehittäjälle. Mitä aikaisemmin mahdolliset virheet havaitaan, sitä helpompi ohjelmistokehittäjän on löytää ja korjata ne. [3, s. 9.]

3.3.2 Virheiden välttäminen

Yritys X:ssä pyritään saamaan työntekijät etukäteen tietoisiksi mahdollisten näkyvien ja näkymättömien virheiden varalle pakollisten koulutuksien avulla. Jokaisen työntekijän on suoritettava virhetietoisuuskoulutus, jonka avulla työntekijät saadaan tietoisiksi laitteissa ja ohjelmistoissa mahdollisesti ilmenevien erilaisten vikojen ja poikkeavuuksien varalle. Kuitenkin testaajan itse aiheuttamia virheitä saattaa syntyä esimerkiksi sellaisissa tilanteissa, kun testaaja käyttää testitapauksessa kokonaan vääriä työkaluja tai ei osaa käyttää jotain testauksessa käytössä olevaa työkalua oikein. Tällaisten virheiden ehkäiseminen on mahdollista, kun testaaja vain muistaa olla testiä tehdessään tarkkaavainen, huolellinen ja varma siitä, mitä on tekemässä. Testaajan on myös äärimmäisen tärkeää muistaa aina tarkistaa, että testauksessa käytössä olevat laitteet ovat varmasti käyttökelpoisia eli ettei niiden huoltopäiväykset ole menneet umpeen. Lisäksi virheiden syntymistä voidaan minimoida välttämällä psykososiaalisten tekijöiden aiheuttamia virhetilanteita ohjelmistotestauksen testitapauksissa. Tällaisia psykososiaalisista tekijöistä johtuvia virhetilanteita voivat aiheuttaa muun muassa epäselvät työtehtävät, jatkuva kiire, pitkät työpäivät, toistuvat ylityöt ja määräaikaiset/osa-aikaiset työsuhteet. Ihmisten tekemiä inhimillisiä virheitä voidaan välttää, kun testaajien työtaakat pysyvät sopivina, jonka ansiosta testaajat eivät ole pahasti stressaantuneita eivätkä väsyneitä. Stressin sekä väsymyksen on nimittäin havaittu johtavan usein työn tuottavuuden vähenemisen lisäksi ihmisen tekemän suorituksen ja toiminnan heikentymiseen, joita ovat muun muassa tarkkaavaisuuden, työmuistin, arvioinnin, reaktionopeuden ja päätöksenteon heikkeneminen. [6; 19, s. 151, 154; 20, s. 44.]

Työkseen testaamista tekevien lisäksi näistä yllä mainituista syistä johtuen myös testiproseduurien eli suoritettavien testien kirjoittajilla voi sattua testiä kirjoittaessaan inhimillisiä virheitä, jotka sitten vaikuttavat omalta osaltaan testien läpi menemiseen. Yritys

X:ssä kaikki sellaiset verifioitavat tulos -kohdat (Result), jotka on merkitty lainausmerkkien sisään tulee olla havaittuna tuloksena täsmälleen samanlaisia. Jos testaajan pitää verifioida, että potilasmonitorin viestikentässä näkyvä hälytysteksti on ”HR high”, mutta havaittavissa olevassa tuloksessa olisikin ”HR High”. Tämän seurauksena tulokseksi tulisi merkitä hylätty (Fail).

Järjestelmälähtöisen ajattelutavan mukaan yksilön tekemän virheen taustalla on kuitenkin useimmiten järjestelmässä oleva virhe. Muuttamalla ohjelmistoa siten, että havaittu virhe saadaan poistetuksi, vältytään silloin virheen toistumiselta. Voi olla, että testaajat joskus epäröivät virheiden ilmoittamista, mutta niistä ilmoittamisen pitäisi oikeastaan tuntua lähinnä kunnia-asialta, koska kun virheitä saadaan minimoitua; tällöin myös potilasturvallisuus kehittyy. [6.]

Riskinhallinnan avulla pystytään myös varautumaan mahdollisten virheiden varalle. Sen apuna käytetään riskien lieventämistä, jolla tarkoitetaan sitä, että testaamalla laitteistoa saadaan mahdollisuuksia vähentää virheiden todennäköisyyttä. Riskinhallintaa käytetään hyväksi etenkin sellaisiin virhetilanteisiin, joilla olisi korkea vaikutus laitteen toimintaan. Riskinhallinta aloitetaan projektien aikaisessa vaiheessa tunnistamalla riskit. Siinä myös kartoitetaan järjestelmän laatuun liittyvät riskit. Tietoa voidaan käyttää apuna testien suunnittelussa, valmistelussa sekä toteuttamisessa. Riskinhallinta on oikeastaan jatkuvaa tiedonkeräystä, tilanteiden seuraamista ja mahdollisista aiheutuneista vahingoista oppimista. Sen avulla koitetaan välttää virheitä, vahinkoja ja onnettomuuksia sekä pyritään selvittämään myös tapahtumien perussyyt. Kun riski on tunnistettu, tehdään riskin arviointi eli sellaisten haittojen analysointi, joita voi aiheutua, mikäli riski toteutuu. Lisäksi analysoidaan, missä määrin ja millä keinoilla riskien aiheuttamia haittoja pystytään hallitsemaan. [20, s. 52; 21.]

Ohjelmiston määrittelyvaiheessa tehdään usein riskianalyysi ja yritetään kartoittaa riskien toteutumistodennäköisyyttä. Lisäksi pyritään selvittämään mahdolliset riskit, jotka uhkaavat ohjelmiston valmistumista, käytön kannattavuutta tai toimintaa. Riskianalyysistä voi olla hyötyä, kun esimerkiksi testattavien ominaisuuksien valinta perustuu jo aiemmin tunnistettuihin riskeihin. Mikäli riskit toteutuvat, voivat ne aiheuttaa jopa katastrofaalisia onnettomuuksia, mutta jos virheet havaitaan ja niihin reagoidaan ajoissa, voidaan silloin välttyä tällaisilta vakavilta onnettomuuksilta. [1, s. 121; 20, s. 53.]

4 Testaukset

Koska Yritys X:n kehittämät ja valmistamat tuotteet ovat lääkinnällisiä laitteita, tulee testaajien työskennellä tarkkaavaisesti potilasmonitorien tulevien käyttöympäristöjen takia. Mikäli potilasmonitorin julkaisun jälkeen ohjelmistossa olisi esimerkiksi testeissä huomaamatta jäänyt vika ja se ei kriittisessä tilanteessa hälyttäisikään, niin silloin potilasturvallisuus vaarautuisi. Nämä tässä tutkimuksessa esitellyt potilasmonitorien ohjelmistotestaukset toimivat yrityksessä osana laadunvarmistusta. Laitteistoja tulee siis testata, jotta voidaan pienentää vakavien virheiden todennäköisyyttä, jolloin myös pystytään minimoimaan vakavien virhetilanteiden riski.

4.1 Testimenetelmät

Yritys X:ssä uuden ohjelmistoversion testejä suoritetaan useissa eri Scrum-tiimeissä, jotka koostuvat noin kymmenestä henkilöstä. Jokaisella pienellä tiimillä on oma mittaus-parametri, josta he ovat vastuussa. Tiimit kehittävät ja testaavat ohjelmistotestauksissa parametriansa, sekä manuaalisesti että automaattisesti. Tässä työssä tarkasteltiin lähemmin parametria EKG, jonka testeistä ja ohjelmistokehityksestä on vastuussa Leijona-tiimi, joka on siis yksi yrityksen Scrum-tiimeistä. Kyseinen parametri valittiin tähän tutkimukseen, koska se oli tekijälle jo entuudestaan tuttu, joten sen testaaminen oli selkeää ja testien suorittaminen oli mahdollista itsenäisesti.

Tässä tutkimuksessa suoritetaan yksi testipaketti, joka sisältää suoritettavia testejä yhteensä kuusi kappaletta. Näistä jokainen sisältää kahdesta neljään erilaista testitapausta. Lisäksi jokaisessa testitapauksesta on useita erilaisia verifioitavia kohtia. Tällaisia verifioitavia kohtia ovat muun muassa tarkastaa, että hälytyksen ääni on oikeanlainen, hälytys on havaittavissa määritellyissä kohdissa potilasmonitorilla ja hälytys on prioriteetiltaan oikea. Suoritettavat hälytystestit testaavat potilasmonitoria suhteellisen kattavasti, sillä aina yhdestä hälytyksestä aiheutuvat tapahtumat sijoittuvat potilasmonitorissa moniin eri kohtiin potilasmonitorin näytölle, sekä siinä oleviin erilaisiin valikkoihin.

Kaikki testattavat kuusi testiä ovat samasta testipaketista, ja ne ovat jo valmiiksi testattu automaattisesti. Automaatiotestien jälkeen nämä kaikki testit suoritetaan uudelleen testaamalla ne manuaalisesti. Testipaketti valikoitui Leijona-tiimin suorittamista auto-

maatiotesteistä manuaalisesti testattavaksi, koska automaatiolla testattuna puolet sen sisältämistä testitapauksista oli jäänyt kiinni virheistä ohjelmistoversiolla, joka sillä hetkellä oli ollut testattavana. Tässä tutkimuksessa haluttiin saada selville, löytyykö näistä jo testatuista testitapauksista manuaalisesti testattuna näiden automaatiolla havaittujen virheiden lisäksi jotain muitakin virheitä, kun käytössä oli täsmälleen sama ohjelmistoversio. Lisäksi haluttiin selvittää, miten suuri ajallinen ero näiden kahden testimenetelmän välillä on sekä syntyykö jommalla kummalla tavalla laadukkaampia tuloksia.

Kaikki kuusi testiä olivat hälytystestejä, joissa testien ideana oli se, että niistä tarkistettiin, aktivoituuko, eskaloituuko ja poistuuko hälytys potilasmonitorista vaatimuksissa määritetyn ajan puitteissa. Nämä potilasmonitorien ohjelmistotestaukseen luokitellut määritykset, joita vaatimuksissa on mainittuina, ovat järjestelmäsuunnittelijan vastuulla. Valitut hälytystestit on luokiteltu kriittisiksi testeiksi, ja ne ovat myös riskien lieventämiseen liittyviä testejä. Riski-analyysin perusteella hälytyksen myöhästyminen tai sen puuttuminen ovat riski potilaalle. Valitut testit liittyvät siis riskinhallintaan, joka on myös molempien uudistettujen standardien ISO 9001:2015 ja ISO 13485:2016 mukaan erittäin tärkeässä osassa, kun pohditaan tuotteen laatua. Lisäksi syynä näiden hälytystestien valitsemiseen tähän tutkimukseen oli se, että näiden vertaileminen oli mahdollista. Nimittäin kaikista testeistä, joita yrityksessä suoritetaan ei välttämättä löydy vertailuun tarvittavia parametreja. Jos testissä ei tarvitse ottaa sekuntikellolla aikaa siitä, kuinka kauan hälytyksen aktivoitumiseen menee, vaan tarvitsee ainoastaan varmistaa jonkin tietyn parametrin näkyminen potilasmonitorilla, niin sellaisesta testistä ei silloin saada kunnollista vertailua aikaiseksi. Tietysti edellytetään, että molemmissa testimenetelmissä kyseisen parametrin todetaan olevan oikeassa kohdassa.

Hälytyksille on näissä suoritetuissa testeissä käytössä olleissa potilasmonitoreissa määritelty neljä erilaista prioriteettia, jotka ovat matala, keskitaso, korkea ja eskaloituva. Nämä eri prioriteetin hälytykset poikkeavat toisistaan muun muassa siten, että jokaisella hälytyksellä on oma tunnusvärinsä ja erilainen hälytysääni. Matalan prioriteetin hälytyksessä värinä on syaani, ja sen hälytysääni on yhdestä piippauksesta koostuva sarja. Tämän hälytyksen hälytysääni on vaihtoehtoisesti, joko yksi ei toistuva äänimerkki tai toistuva yhden piippauksen äänimerkki, jolloin hälytysääni toistuu noin 25 sekunnin välein. Testaaja voi itse määritellä tämän matalan tason hälytyksen toistuvuuden potilasmonitorilta salasanan takana olevasta valikosta. Puolestaan keskitason prioriteetin hälytyksen väri on keltainen, ja sen hälytysääni koostuu kolmen piippauksen sarjasta, joka toistuu noin 19 sekunnin välein. Korkean prioriteetin hälytyksen värinä on

punainen, ja sen hälytysääni on kaksi kertaa viidestä piippauksesta koostuva sarja, joka toistuu noin viiden sekunnin välein. Prioriteetiltaan eskaloituva hälytys on sellainen, joka näissä tutkimuksessa testatuissa testeissä alkaa keskitason hälytyksestä ja tietyn vaatimuksissa määritellyn ajan jälkeen se muuttuu korkean prioriteetin hälytykseksi, eli se siis eskaloituu.

Suoritetut testit testaavat järjestelmävaatimuksia, joten ne liittyvät kappaleessa 2.2 esitellyn V-mallin järjestelmätestaamisen vaiheeseen. Yritys X:n testiprosessi kuitenkin eroaa vesiputous- ja V-mallista siten, että näitä järjestelmätestauksia suoritetaan myös jo ohjelmistokehityksen aikana, eikä vasta silloin, kun ohjelmisto on kokonaisuudessaan valmis. Yrityksen testiprosesseissa uusien ohjelmistojen kehittäminen tapahtuu näiden molempien mallien perusteella, sillä nämä mallit kuvaavat testeissä olevia eri tason vaiheita. Kun yrityksessä kehitetään jo olemassa olevaa ohjelmistoversiota, niin aina tehtyjen muutosten jälkeen täytyy varmistaa testaamalla ohjelmistoa monella eri tavalla, ettei tehdyllä uudistuksella ole ollut haitallisia sivuvaikutuksia. Tätä jatkuvasti toistuvaa varmistustestausta kutsutaan siis myös nimellä regressiotestaus, jota suoritetaan automaatiotestauksen avulla.

Tarkastellessa vesiputousmallia ja yrityksessä tapahtuvaa ohjelmistojen kehittämisprosessia, niin sekä vesiputousmallissa olevat vaiheet että yrityksen ohjelmistokehityksessä tapahtuvat vaiheet ohjelmiston valmiiksi saattamisessa täsmäävät keskenään. Ensimmäisenä kehitettäville tuotteille tehdään tietynlaiset vaatimukset, joita niiden täytyy ehdottomasti noudattaa, jonka jälkeen päästään sitten suunnittelemaan itse ohjelmistoa. Näiden vaiheiden jälkeen voidaan aloittaa toteutusosio, ja kun ohjelmistokehittäjät ovat saaneet uuden ohjelmistoversion valmiiksi, pystytään sen jälkeen aloittamaan sen testiprosessit. Viimeisenä vaiheena on ylläpito, jossa tarkistetaan aina pienenkin muutoksen jälkeen, että tuotteen julkaisun jälkeen havaitut virheet on varmasti korjattu ja etteivät nämä korjaustoimenpiteet ole aiheuttaneet ohjelmistoon uusia virheitä. Periaatteessa tämä koko malli kulkee koko ajan käsi kädessä edellisen vaiheen kanssa, sillä jokaisessa vaiheessa tulee ohjelmiston noudattaa sille alussa määriteltyjä vaatimuksia.

V-malliin verrattaessa pätee myös sen ja Yritys X:ssä tapahtuvan ohjelmistokehitysprosessin yhtäläisyys. V-mallissa testit on jaettu neljään eri testivaiheen tasoon, joista Yritys X:ssä suoritettavissa testauksissa sovelletaan yksikkötestausta, integraatiotestausta ja järjestelmätestausta. Nämä suoritettavat eri tason testit suoritetaan siis peräkkäin,

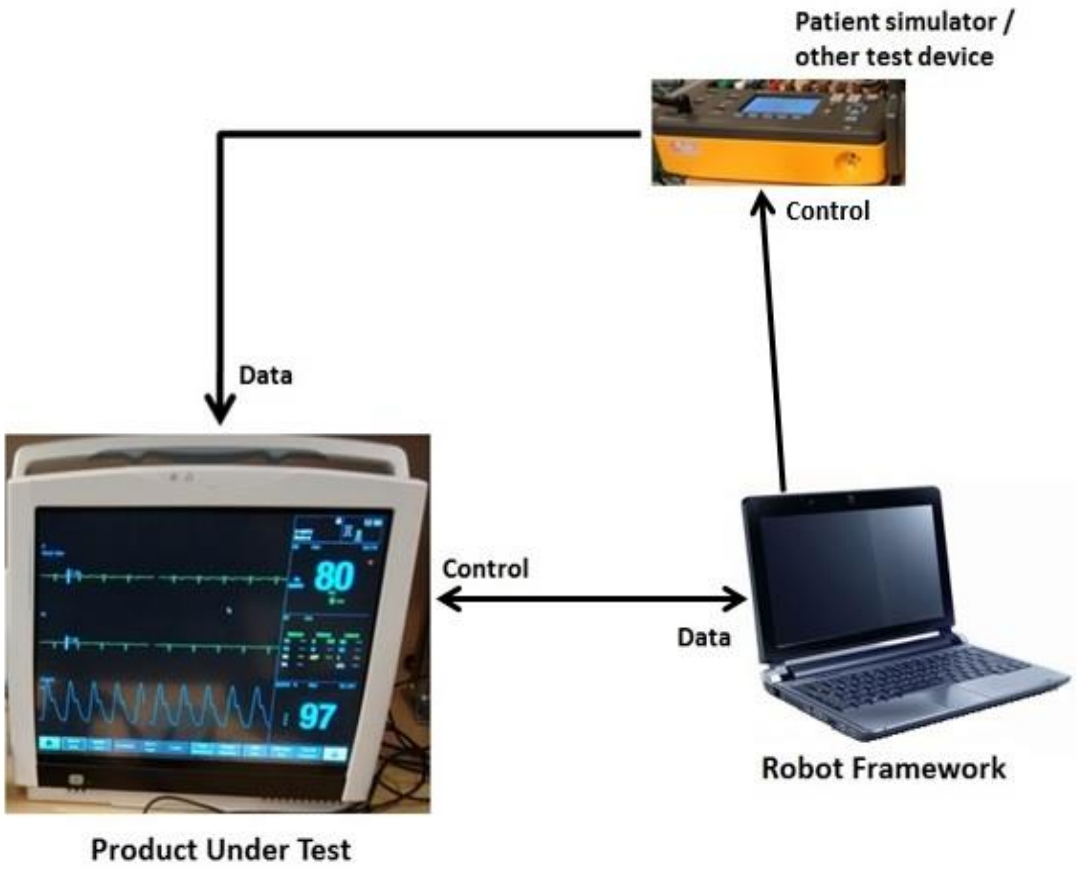
eli tämä lopputuloksena saatava valmis kokonainen ohjelmisto koostuu testatuista osista, jotka sitten kootaan yhdeksi isommaksi kokonaisuudeksi.

Scrum-malliin verrattaessa myös Yritys X:ssä uuden ohjelmistoversion kehityksessä ja sen testaamisessa käytetään tämän mallin periaatteita. Scrum-tiimeillä on aina viikoittaiset kokoukset, joissa käydään läpi senhetkiset tilanteet. Yrityksen toiminnan ja Scrum-mallin erona on se, että mallin mukaan tuotteen ollessa valmis se voidaan toimittaa markkinoille, mutta Yritys X:ssä näin ei kuitenkaan voida tehdä. Tämä johtuu siitä, että Scrum-mallissa ei suoriteta ollenkaan verifikaatiota, mutta yrityksen laatujärjestelmä kuitenkin vaatii sen suorittamisen laitteille potilasturvallisuuden vuoksi.

4.2 Automaatio

4.2.1 Automaattisesti ohjattavat testilaitteet

Automaattisesti tehtävät järjestelmä- ja integraatiotestit suoritetaan Yritys X:ssä ajamalla testitapaukset läpi Robot Frameworkin kautta ja sieltä saadut tulokset kirjautuvat automaattisesti Jenkins -ohjelmistoon. Tässä Jenkins-ohjelmistossa samaan testiin liittyviä testitapauksia voi kuitenkin olla tallennettuina vain rajallinen määrä, joten aina kun uusi testi on suoritettu, niin vanhin testi poistuu sieltä automaattisesti. Mikäli jokin tietyllä ohjelmistoversiolla suoritettu testitapahtuma on jossain määrin testin läpiajajan mielestä tärkeä, se tulee tallentaa toiseen ohjelmistoon. Testiautomaatiota ajava laitteisto on esiteltynä kuvassa 4.



Kuva 4. Automaatiotestauksen toiminta. [Yritys X:n materiaali]

Tietokoneelle asennettu Robot Framework -työkalu ajaa testitapauksia läpi ja testattavana olevaa potilasmonitoria (Product Under Test) valvotaan (Control) yrityksen sisäisen verkkoyhteyden kautta. Tietokoneella käytössä olevasta Robot Frameworkista lähtee tiedot halutuista simuloitavista arvoista potilassimulaattorille tai muulle testauksessa käytössä olevalle testilaitteelle (Patient simulator / other test device), josta haluttu data puolestaan siirtyy potilasmonitoriin. Tämän sisäisen verkkoyhteyden kautta kaikki testauksista saadut tulokset ja mahdolliset muut ilmoitukset (Data) siirtyvät automaattisesti tietokoneelle, jonka jälkeen testaaja voi aloittaa testeistä saatujen tuloksien tarkistamisen.

Automaattisesti suoritettavassa testauksessa ohjelmistoversion vaihtaminen potilasmonitoriin tapahtuu niin, että käyttäjä syöttää testiautomaatiolle tietyn komennon. Tämän jälkeen syötetty komento aloittaa suorittamaan sille määritellyn listan mukaisesti eri testivaiheita. Ensin se tarkistaa yrityksen sisäiseltä palvelimelta mikä on uusin käytettävissä oleva ohjelmistoversio, jonka jälkeen se tarkistaa mikä ohjelmistoversio poti-

lasmonitorissa on jo valmiiksi asennettuna. Mikäli uusin mahdollinen ohjelmistoversio on jo asennettuna, silloin automaatiotyökaluun syötetyn komennon alainen suoritus loppuu. Jos potilasmonitorissa valmiina oleva ohjelmistoversio on eri kuin uusin mahdollinen saatavilla oleva ohjelmistoversio on, niin automaatiotyökalu aloittaa silloin kyseisen uusimman ohjelmistoversion latauksen. Tämän jälkeen automaatiotyökalu aktivoi uuden ohjelmistoversion potilasmonitorilla ja lopettaa sillä hetkellä käynnissä olevan potilastapauksen, jonka ansiosta uuden ohjelmistoversion päivitys käynnistyy. Seuraavaksi automaatiotyökalu odottaa, että potilasmonitori käynnistyy uudelleen ja viimeisenä se tarkistaa, että tämä uusi ladattu ohjelmistoversio on varmasti käytössä.

Automaatiolla suoritettavissa testiprosesseissa on apuna erilaisia testikirjastoja, jotka säätävät erilaisia määrityksiä testeille. Näissä testeissä käytössä olevista eri kirjastoista esimerkiksi yhdellä voidaan ohjata potilassimulaattoria simuloimaan arvoja potilasmonitorille, ja toisen kirjaston avulla saadaan kirjatuksi havaittu tulos ja päätös tulokselle (Pass/Fail). Automaation avulla voidaan ohjata myös useita muita erilaisia tapahtumia, joita Yritys X:ssä tarvitaan testien suorittamiseen. Tällaisia tilanteita ovat esimerkiksi sellaiset, joissa verkkovirtakytkimen avulla saadaan potilasmonitori päälle tai pois päältä, mikäli siinä ei ole paristoa ja puolestaan verkkokytkimen avulla saadaan valittua käyttöön testissä tarvittavia moduuleja.

4.2.2 Testien suorittaminen

Automaattisesti suoritettavassa eskaloituvassa hälytystestissä käytössä oleva työkalu ajaa ensin läpi laitteistolle tehtävät asetukset. Työkalu tyhjentää potilasmonitorin kaikki lokitiedostot ja määrittää hälytykselle valitun prioriteetin. Työssä tarkastelluissa testitapauksissa automaattisesti suoritetuissa testeissä hälytyksien prioriteettina oli viidessä testissä eskaloituva, ja yhdessä testissä prioriteetin valinta oli ohjelmoitu määritettäväksi satunnaisella valinnalla. Tässä tapauksessa hälytyksen prioriteetiksi valikoitui korkea. Tämän jälkeen automaatiotyökalu testaa valmiiksi ohjelmoituja testivaiheita. Aluksi se asentaa potilasmonitorin näytölle näkyviin tarvittavat parametrit, jonka jälkeen se tarkistaa muun muassa, ettei mitään hälytyksiä ole päällä, kirjaa ylös sen ohjelmistopaketin nimen, joka on käytössä, ja lisäksi ilmoittaa minimi- ja maksimiarvot, joita potilassimulaattorilla voidaan EKG-parametrille simuloida. Tämän jälkeen automaatiotyökalu aiheuttaa hälytyksen ja laittaa sekuntikellon päälle. Kun hälytyksen tila on saavutettu, automaatio tarkastaa, onko hälytys aktiivinen ja aktivoituiko se vaatimuksissa määritetyn ajan sisällä sekä ovatko hälytyksen audiovisuaaliset eli nähtävät ja kuultavat

hälytyksen ilmoitukset näkyvissä määritellyissä kohdissa ja ovatko ne oikean prioriteetin mukaiset. Seuraavaksi automaatio-työkalu ottaa sekuntikellolla aikaa siitä, kun hälytyksen tila on saavutettu varmistaakseen, eskaloituuko hälytys korkeamman tason hälytykseksi vaatimuksissa määritetyn ajan sisällä. Seuraavaksi se tarkistaa, onko hälytyksen tila eskaloitunut sekä ovatko audiovisuaaliset ilmoitukset havaittavissa määritellyissä kohdissa ja ovatko ne oikean prioriteetin mukaiset. Tämän jälkeen automaatio-työkalu suorittaa seuraavat testit, mikäli sille on sellaisia määritelty vielä toteutettavaksi.

Edellä kuvattu testikokonaisuus pyörii jatkuvasti regressiotestinä. Toisin sanoen aina, kun ohjelmoijat kehittävät uuden ohjelmistoversion, tämä uusi ohjelmistoversio testataan näillä tähän testipakettiin kuuluvilla testitapauksilla. Regressiotestauksen ansiosta voidaan olla varmoja, etteivät tehdyt muutokset ole aiheuttaneet uusia virheitä testattavana olevaan ohjelmistoversioon. Automaatiotestaus sopii regressiotestaukseen paremmin kuin manuaalinen testimenetelmä, koska automaatiolla testaaminen on huomattavasti nopeampaa, eikä se vaadi niin paljon testaajien työaikaa.

Tähän tutkimukseen valittu uutta ohjelmistoversiota testaava testikokonaisuus on siis yhden yrityksessä olevan Scrum-tiimin vastuulla. Testikokonaisuutta kehitetään sekä testataan pienissä osissa. Mikäli testistä löydetään jokin virhe ja se liittyy testiproseduuriin, Scrum-tiimi hoitaa itse sen korjaamisen, mutta jos virhe on testikirjastoissa, sen korjaamisesta huolehtii Robot-tiimi. Yhteisenä työkaluna näillä eri tiimeillä on Perforce, joka on versionhallintatyökalu. Kyseisessä työkalussa vastuut on jaettu kahteen eri kansioon, joista toinen kansio on Scrum-tiimien ja toinen Robot-tiimin. Tiimien vastuita on tasapainotettu sen mukaisesti, miten tiimeillä on ollut resursseja ja tällä kansiojaolla vastuualueet pystytään pitämään selkeinä. Robot-tiimi ei käytä hyödykseen Scrumia, vaan se käyttää Kanban boardia, sillä siinä ei ole aikaan sidottuja työskentelyjaksoja, sillä se on jatkuva palvelumalli. Tiimin on vaikea suunnitella töitä jaksoihin, sillä se saa jatkuvasti uusia töitä muilta tiimeiltä. Kun muut tiimit lähettävät Robot-tiimille jonkin työn, voivat he määrittää sille haluamansa prioriteetin, joiden perusteella Robot-tiimi valitsee, mitkä työt tulee saada ensimmäisenä valmiiksi.

4.3 Manuaalinen testaus

4.3.1 Testien valmisteleminen

Ennen kuin manuaalisella testimenetelmällä suoritettavia testejä pystyttiin tässä tutkimuksessa suorittamaan, täytyi automaatiolla tehdyt testit ensin kirjata automaatiossa käytössä olevasta Jenkins-ohjelmistosta Microsoft Word -tekstinkäsittelyohjelmaan. Testien suorittaminen manuaalisesti oli tämän jälkeen selkeämpää, kun tekstinkäsittelyohjelmaan siirrettiin ainoastaan sellaiset kohdat, jotka olivat manuaalisiin testeihin liittyen tarpeellisia. Tällaisia kohtia olivat testeissä käytössä olleet laitteet, aiheutettavan hälytystilan tiedot ja verifioitavat testikohdat, joista kirjattiin lisäksi ylös automaatiolla saadut tulokset, ja tulosten vertailua varten myös testeihin kuluneet ajat, sekä sen lisäksi testien suorittamiseen mennyt kokonaisaika.

Yleisesti ajatellen tämä testien kirjoitusprosessi sujui suhteellisen helposti, kun automaatiossa käytössä olevan ohjelmiston käyttöön ensin hieman tutustui. Kuitenkin kolmannen suoritettavan testin kanssa oli jonkin verran epäselvyyksiä, sillä testin sisältämien kahden testitapauksen otsikointi oli suhteellisen epäselvästi ilmaistu. Koko testin otsikossa mainittiin, että kyseessä on eskaloituvan hälytyksen testaaminen ja kahdessa sen sisältämän testitapauksen otsikossa puhuttiin selkeästi, että testataan sykehälytyksen eskaloitumista. Puolestaan kahden muun testitapauksen otsikoinnista sai aluksi sellaisen kuvan, että kyseessä olisikin korkean prioriteetin hälytyksen verifioiminen. Ristiriita näiden kahden testin oikein suorittamisen suhteen tuli, kun automaattisesti suoritetuissa testeissä tuloksissa oli kuitenkin määritelty hälytyksen prioriteetiksi eskaloituva. Tämän lisäksi automaattisesti suoritettujen testien tuloksissa oli aluksi saatu keskitason hälytys, joka oli sitten eskaloitunut korkean prioriteetin hälytykseksi eli kyseessä ei siis ollutkaan pelkän korkean prioriteetin hälytyksen verifioiminen. Näissä kahdessa testissä kyseessä olikin nopean eskaloitumisen testaaminen, joka tarkoittaa sitä, että hälytystila aktivoituu suoraan eskaloituneeseen vaiheeseen eli näissä testitapauksissa korkean prioriteetin tilaan. Tämä nopea eskaloituminen tapahtui kuitenkin keskitason prioriteetista korkeaksi, syynä tähän oli se, että näissä kahdessa testitapauksessa EKG:n tilaksi oli määriteltynä lihasartefakta. EKG-lihasartefakta eli EKG-häiriö on lihasjännitystila, jossa potilasmonitorilla näkyvissä olevassa EKG:n käyrässä on havaittavissa vapinaa.

Kun kaikki automaatiotestit oli saatu purettua Jenkins-työkalusta, pystyi manuaalitestien suorittamisen aloittaa. Aluksi piti varmistaa, että ennen testien aloittamista kaikki käytössä olevat laitteet ovat samanlaisia kuin mitä automaatiotestauksissa on ollut käytössä testejä suoritettaessa. Näissä molemmissa testeissä oli käytössä potilasmonitori, johon oli asennettuna sama ohjelmistoversio ja ohjelmistopaketti, joka näissä suoritetuissa testeissä vastasi teho-osastoa. Lisäksi molemmissa testeissä käytettiin potilassimulaattoria, sekä moduuleja, joihin oli kiinnitettyinä vastaavanlaiset kaapelit (EKG ja SpO_2). Testeissä käytettiin näitä täsmälleen samanlaisia laitteita, jotta saadut tulokset olisivat mahdollisimman samankaltaisia.

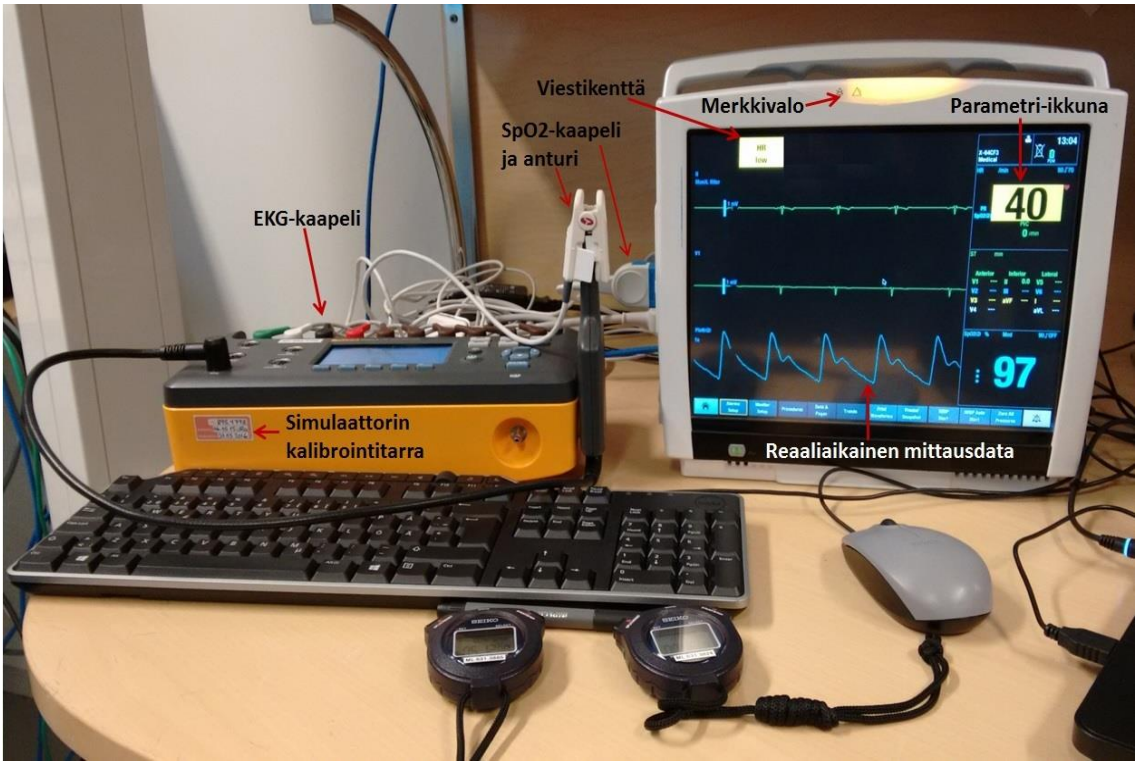
4.3.2 Testien suorittaminen

Manuaalinen testimenetelmä etenee sillä tavalla, että ensimmäisenä valitaan testiproseduurissa ilmoitetun testin suorittamiseen tarvittavan potilasmonitorin, johon asennetaan tarvittava ohjelmistoversio sisäisessä verkkoyhteydessä kiinni olevan tietokoneen avulla. Ohjelmistoversion vaihtaminen tapahtuu niin, että valitaan yrityksen sisäisessä verkossa olevalta palvelimelta testattavan ohjelmiston version, joka ladataan tietokoneelle. Tämän jälkeen avataan tietokoneelle asennettuna oleva DAEMON Tools Lite -niminen työkalu, jonka avulla ohjelmistoversion muuttaminen potilasmonitoriin tapahtuu. Työkalun avulla pystyy luomaan virtuaalisen CD -aseman tietokoneelle, jonka kautta ohjelmistoversion muuttamisprosessi potilasmonitoriin tapahtuu. Seuraavaksi ohjelmistoversio aktivoidaan potilasmonitorilla, ja lopetetaan sillä hetkellä toimissa meneillään oleva potilastapaus ja käynnistetään potilasmonitori uudelleen. Tämän jälkeen tarkistetaan, että vaihdettu ohjelmistoversio on varmasti päivittynyt oikeaksi.

Ennen testien aloittamista täytyy testiproseduurista lukea, mitä eri laitteita testin suorittamisessa tarvitaan, ja kun kaikki testiproseduurissa ilmoitetut testin alussa tarvittavat laitteet on yhdistetty oikean ohjelmistoversion, sekä ohjelmistopaketin omaavaan potilasmonitoriin voi testin suorittamisen aloittaa. Testin alussa potilasmonitorille määritellään testiproseduurin mukaisesti erilaisia asetuksia, joita testin suorittamiseen oikealla tavalla tarvitaan. Tämän jälkeen voi aloittaa testaamisen, jossa varmistetaan, toimiiko testattavana oleva ohjelmistoversio käytössä olevassa laitteistossa niin kuin sen pitäisi toimia.

Kolmessa tässä tutkimuksessa suoritetussa manuaalisessa testissä oli testilaitteistona käytössä potilasmonitori, yksi moduuli, 10-johtoinen EKG-kaapeli, potilassimulaattori

sekä näiden lisäksi kaksi sekuntikelloa, joista toisella mitattiin testeihin kuluvaa kokonaisaikaa ja toisella hälytysten aktivoitumista, eskaloitumista sekä niiden poistumista. Muissa kolmessa testissä oli yllä mainittujen laitteiden lisäksi käytössä toinen moduuli, SpO_2-kaapeli, sekä potilassimulaattorissa kiinnitettynä oleva SpO_2-simulaattori. Kaikissa manuaalisesti suoritetuissa testitapauksissa hälytyksien prioriteeteiksi valittiin täsmälleen samat kuin automaatiotestauksissa. Kuvassa 5 on esitelty kolmessa manuaalitestauksessa käytössä ollut laitteisto sekä potilasmonitorissa päällä oleva hälytystila.



Kuva 5. Manuaalitestaus, keskitason prioriteetin hälytys.

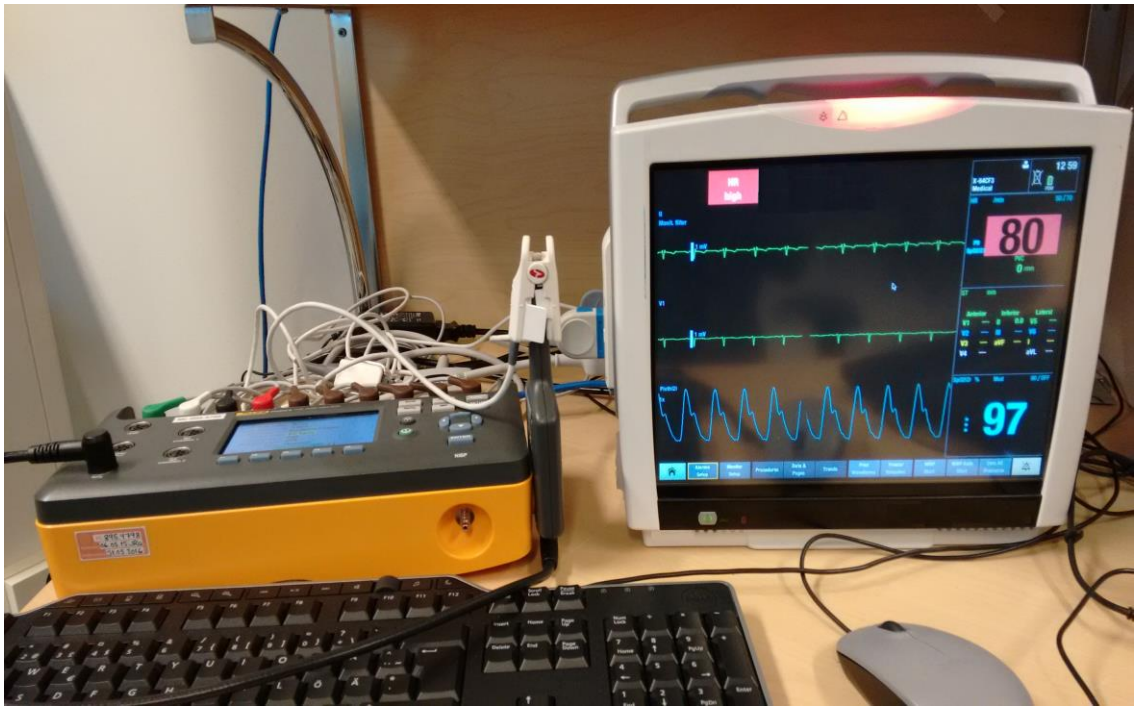
Kuvassa 5 vasemmalla olevassa potilassimulaattorissa on oranssireunainen tarra, josta käy ilmi laitteen kalibrointipäivät eli milloin se on kalibroitu viimeksi ja mihin mennessä se on viimeistään kalibroitava seuraavan kerran, sekä sen henkilön nimikirjaimet, joka on suorittanut laitteen kalibroinnin edellisellä kerralla. Potilassimulaattoriin on kiinnitetty EKG-kaapelin mittauselektrodit ja potilassimulaattorissa kiinni olevassa SpO_2-simulaattorissa on kiinni SpO_2-kaapelin toisessa päässä oleva anturi. EKG- ja SpO_2 -kaapelit ovat molemmat kiinnitetty omissa moduuleissaan, jotka ovat puolestaan yhdistettyinä potilasmonitoriin. Kuvassa olevasta potilasmonitorista on havaittavissa kaksi erilaista mitattavaa parametria, jotka ovat nähtävissä sekä potilasmonitorin oikeassa laidassa olevassa parametri-ikkunassa, että reaaliaikaista mittausdataa näyttävissä

käyrissä, jotka ovat vasemmasta laidasta oikeaan laitaan reaaliajassa liikkuvia. Nämä parametrit ovat SpO_2(2), joka on myös valittuna sydämen sykkeen lähteeksi, ja EKG, josta on valittu näytöllä näkyväksi kaksi mittausta, EKG1 ja EKG2. Happisaturaation mittauksessa on käytössä SpO_2(2), joka johtuu siitä, että monitorissa kiinni olevissa kahdessa moduulissa on molemmissa mahdollisuus SpO_2-mittaukseen. Näistä kahdesta mahdollisesta SpO_2-mittauksesta potilasmonitori ilmoittaa testattavana olevan parametrin nimeksi SpO_2(2), koska SpO_2(2) on sekundaarinen, sillä testeissä käytössä olevilla moduuleilla on oma arvojärjestyksensä.

Kuvassa 5 näkyvä hälytys on prioriteetiltaan keskitason hälytys, jonka tunnistaa siitä, että potilasmonitorin merkkivalo vilkkuu keltaisena, parametri-ikkunassa oleva HR-arvo (sydämen syke) 40 ja viestikentässä näkyvä hälytyksen teksti ”HR low” ovat molemmat ympäröity keltaisella värillä. Kuvassa näkyvässä testitilanteessa on aluksi potilasmonitorin näytölle asetettu näkyväksi EKG ja SpO_2(2) -parametrit. Tämän jälkeen näille parametreille on määritelty halutut raja-arvot eli alhaisimmat ja korkeimmat arvot, jotka kyseisille parametreille on tässä testitilanteessa ollut hyväksyttävissä. HR:lle on tässä testissä määritelty alhaisimmaksi arvoksi 50 1/min (sydämen lyönti per minuutti) ja puolestaan korkeimmaksi arvoksi 70 1/min. Seuraavaksi testissä on varmistettu, ettei potilasmonitorissa ole mitään hälytyksiä päällä, jonka jälkeen on aiheutettu ”HR low” hälytys simuloimalla potilassimulaattorilla HR:lle arvoksi 40 1/min. Kun HR:n arvo alkoi potilasmonitorin näytöllä muuttua, käynnistettiin sekuntikello ja mitattiin aikaa, kuinka kauan siitä hetkestä menee, että potilasmonitorilla on havaittavissa kyseinen hälytystila. Kun keskitason hälytystila on saatu aiheutettua, verifioidaan, aktivoituuko hälytys vaatimuksissa määritetyn ajan sisällä, onko hälytyksen ilmoitukset näkyvissä määritellyissä kohdissa potilasmonitorilla sekä vilkkuuko merkkivalo ja kuuluuko hälytysääni.

Tässä suoritetussa testissä hälytys aktivoitui 4,68 sekunnissa, joka ei kuitenkaan ole vaatimuksissa määritetyn ajan sisällä: 0 s ± toleranssi (2 s), joten tämä testikohta hylättiin (Fail). Puolestaan hälytyksestä verifioitavat audiovisuaaliset testikohdat olivat havaittavissa oikealla prioriteetilla, joten nämä testikohdat menivät hyväksytysti läpi (Pass). Näiden jälkeen mitattiin, kuinka kauan aikaa kestää, että potilasmonitorilla oleva keskitason hälytys eskaloituu korkean prioriteetin hälytykseksi, ja jälleen verifioida audiovisuaalisten ilmoitusten näkyvyys oikealla prioriteetilla. Näistä verifioitavista kohdista molemmat menivät hyväksytysti läpi (Pass). Aikaa hälytyksen eskaloitumiseen meni 69,17 sekuntia, joka on vaatimuksien mukainen: 69 s ± toleranssi (2.0 s), ja lisäksi audiovisuaaliset kohdat olivat oikean prioriteetin mukaisia.

Kun hälytys on eskaloitunut keskitason hälytyksestä korkean prioriteetin hälytykseksi, kaikki potilasmonitorissa näkyvissä olevat hälytykseen viittaavat ilmoitukset ovat väriltään punaisia. Kuvassa 6 on esiteltynä korkean prioriteetin hälytys.



Kuva 6. Manuaalitestaus, korkean prioriteetin hälytys.

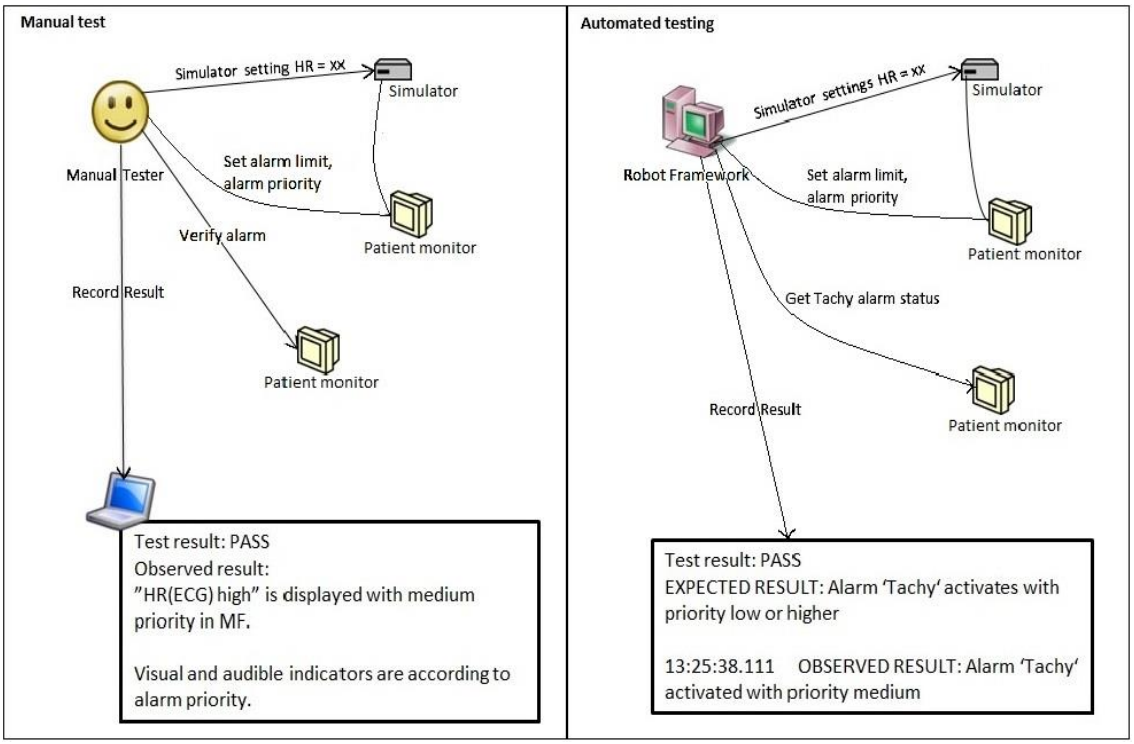
Kuvassa 6 testitapauksena on ”HR high” -hälytys, joka on testattu muuten samalla tavalla kuin kuvassa 5 oleva hälytys, mutta tässä testitapauksessa on simuloitu potilassimulaattorilla korkeimman raja-arvon ylittävä arvo 80 1/min. Aikaisempaan kuvaan 5 verrattuna tästä yllä olevasta kuvasta 6 on nähtävissä, että kaikki hälytykseen viittaavat ilmoitukset ovat muuttuneet väriltään punaisiksi. Samoin myös hälytyksen äänimerkki oli muuttunut keskitason hälytyksessä olevasta kolmen piippauksen sarjasta viiden piippauksen mittaiseksi sarjaksi. Koska hälytyksen prioriteetti on muuttunut korkeammaksi, näiden mainittujen muutosten lisäksi ovat hälytysäänen ja potilasmonitorin merkkivalon toistuvuus myös nopeutuneet, eli hälytysäänen välissä on lyhempi hiljainen tauko ja sen merkkivalo vilkkuu nopeammin.

Molempien kuvissa 5 ja 6 näkyvien hälytystilojen lisäksi hälytyksien tietoja on nähtävissä muun muassa parametrivalikosta, jossa hälytyksen tilaa kuvaava teksti on ympäröitynä hälytyksen prioriteetin mukaisella värillä. Parametrivalikossa on näkyvissä hälytykseen liittyvä ikoni, joka on kolmion muotoinen ja väriltään se on myös sen hetkisen

prioriteetin mukainen. Lisäksi hälytyksen tiedot tallentuvat potilasmonitorissa olevaan lokiin, josta niitä pystyy tarkastelemaan myös jälkikäteen.

4.4 Tulosten kirjaaminen

Yllä esiteltyjen testimenetelmien automaattisen- ja manuaalisen testauksen tuloksien kirjauksessa on jonkin verran eroavaisuuksia. Kuvasta 7 on nähtävissä tulosten kirjausprosessien erot.



Kuva 7. Automaation ja manuaalitestauksen tulosten kirjaus. [Yritys X:n materiaali]

Kuvassa 7 vasemmalla puolella olevassa manuaalisessa testimenetelmässä (Manual test) testaaja (Manual Tester) valitsee potilasmonitorista (Patient monitor) aiheutettavalle hälytykselle haluamansa raja-arvot sekä hälytyksen prioriteetin (Set alarm limit, alarm priority). Tämän jälkeen testaaja simuloi potilassimulaattorilla (Simulator) parametrin arvon täsmälleen samaksi tai korkeammaksi kuin mitä raja-arvon yläraja on äsken asetettu olevan (Simulator setting HR = xx), jonka jälkeen testaaja verifioi näkemänsä hälytyksen potilasmonitorissa (Verify alarm) ja kirjaa tulokset tietokoneella olevaan ohjelmistoon. Testaaja vertaa testiproseduurissa ilmoitettua odotettua tulosta (Expected result) omaan havaitsemaansa tulokseen (Observed result), ja tekee pää-

töksen tuloksesta sen mukaisesti. Kuvassa olevassa esimerkissä tulos on ollut hyväksyttävä (PASS), ja testaaja on kirjoittanut havainneensa oikean hälytyksen potilasmonitorin viestikentässä (MF = message field). Kyseinen hälytys on ollut prioriteetiltaan keskitasoinen (medium priority). Tämän lisäksi testaaja on myös verifioinut, että näkyvät ja kuuluvat hälytykseen liittyvät havainnot ovat olleet oikean prioriteetin mukaiset (Visual and audible indicators are according to alarm priority).

Vastaavasti oikeanpuoleisessa kuvassa automaattisesti testattaessa (Automated testing) tietokoneella oleva ohjelmiston ajama testiskripti määrittää potilasmonitorille etukäteen ohjelmoidut hälytyksen raja-arvot ja hälytyksen prioriteetin, jonka jälkeen se simuloi potilassimulaattorista lähtevän datan täsmälleen samaksi tai yli ylärajan arvon. Tämän jälkeen se saa tiedon hälytyksen tilasta ja kirjaa tulokset ohjelmistoon ylös. Automaattisesti kirjatuista tuloksista on nähtävissä, että testi on mennyt hyväksytysti läpi (PASS). Automaatio kirjoittaa havaitun tuloksen (Observed Result) lisäksi odotetun tuloksen (Expected Result), joiden tulee täsmätä keskenään, jotta testitapaus menee hyväksytysti läpi.

Kuten kuvasta 7 näkyy, tulosten kirjausprosessi on täysin erilainen näillä kahdella eri testimenetelmällä. Automaattisesti testattaessa tuloksissa huomioidaan havaitun tuloksen lisäksi odotettu tulos ja tuloksiin kirjautuu automaattisesti myös kellonaika, jolloin tulos on saatu. Puolestaan manuaalisesti suoritetuissa testeissä testaaja kirjoittaa testituloksiin vain havaitsemansa tuloksen. Automaattisesti testattaessa tulosten kirjausprosessi on aina samaa testiä suoritettaessa samanlainen, kun taas manuaalisesti testatuissa testeissä jokainen kirjattu tulos on erilainen, sillä jokainen testaaja kirjoittaa havaintonsa tuloksiin omalla hyväksi havaitsemalla tavallaan.

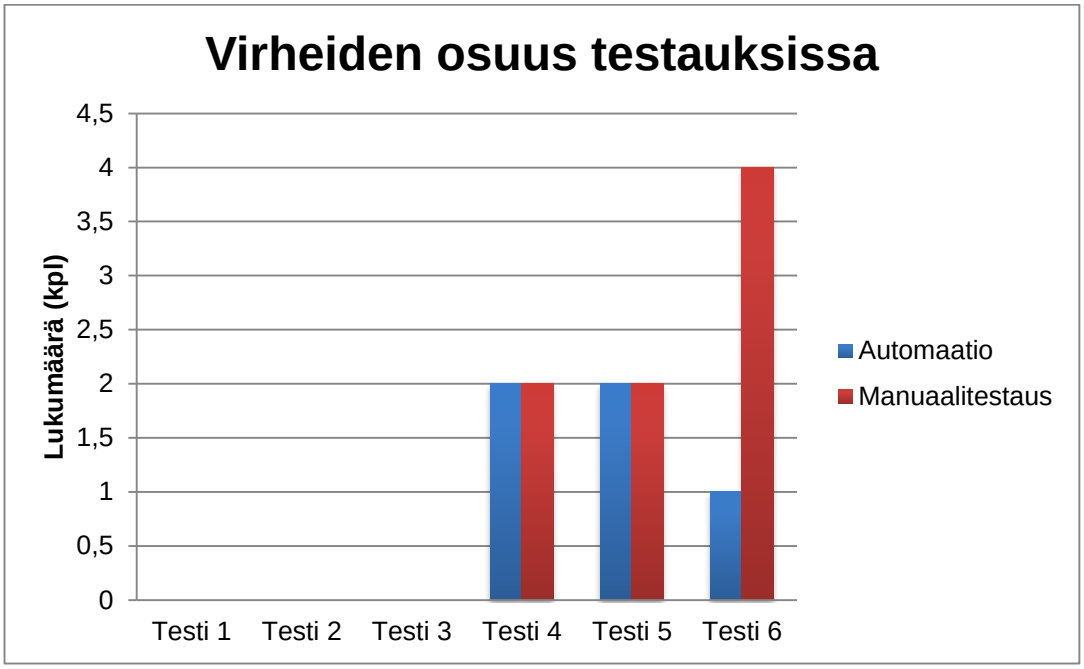
Tulosten katselmointi eli ohjelmiston toiminnan tarkastus suoritetaan manuaalisesti testatuissa testeissä vasta sen jälkeen, kun testit on suoritettu. Automaatiossa testitulosten kirjaaminen katselmoidaan jo ennen testin suoritusta. Nämä automaatiossa olevien tulosten kirjaukset testien ja eri testikirjoittajien välillä ovat yhdenmukaisia, sillä suurin osa kirjaamisesta tehdään uudelleen käytettävissä kirjastofunktioissa, jotka ovat jo etukäteen määriteltyjä. Kuten kappaleessa 3.2.1 on jo mainittu, niin Yritys X:ssä dokumentaation laatu on merkittävänä osana tuotteen konkreettista laatua, etenkin viranomaisten kannalta. Kirjattujen tulosten tulee olla kirjoitettuina näillä molemmilla testimenetelmillä huolellisesti ja selkeästi jokaiseen sellaiseen kohtaan, mihin niitä vaaditaan.

5 Tulokset

5.1 Laadulliset eroavaisuudet

Laadullisiin eroavaisuuksiin sijoittuu tilanne, jossa automaattisesti suoritetuissa testeissä havaittiin hylätty testitulos, jonka seurauksena automaatio lopetti testin tekemisen kesken. Esimerkiksi kuudennessa testissä on automaatiolla suoritettu sen sisältämistä neljästä testitapauksesta vain yksi, kun taas manuaalisesti testattaessa on suoritettu nämä kaikki neljä testitapausta. Tämä johtuu siitä, että mikäli jokin kohta automaatiolla suoritettavassa testissä saa tulokseksi hylätty, automaatio olettaa, että kyseinen testijärjestelmä ei ole toimiva, jonka takia loppuja testikohtia ei enää suoriteta. Syynä tähän toimintaan oli lisäksi se, että automaatiotestin kirjoittaja oli laittanut nämä testit ajettavaksi niin, että jokainen testi oli yksi kokonainen taulukko. Tästä johtuen koko testi siis keskeytyi sen seurauksena, että yksi testikohdista sai tulokseksi hylätty. Kun testitapauksia testataan automaatiolla, testin saadessa tuloksen hylätty on mahdollista laittaa testitapaus joko uudelleen ajettavaksi tai vaihtoehtoisesti rakentaa testitapaukset siten, että jokainen testikohta on oma rivinsä. Tämä tarkoittaa sitä, että testin pystyisi kirjoittamaan myös sellaiseen muotoon, että yksi rivi on aina yksi testitapaus, jolloin jokainen testikohta tulisi suoritetuksi, vaikka jokin edellä olleista testikohdista olisi hylätty. Kun kyseessä on manuaalinen testimenetelmä, testaaja voi suorittaa jokaisen testitapauksen välittämättä siitä, onko joku aikaisemmista testikohdista hylätty. Lisäksi manuaalitestauksessa testaaja pystyy käyttämään omaa harkintakykyään testin suorittamisen suhteen.

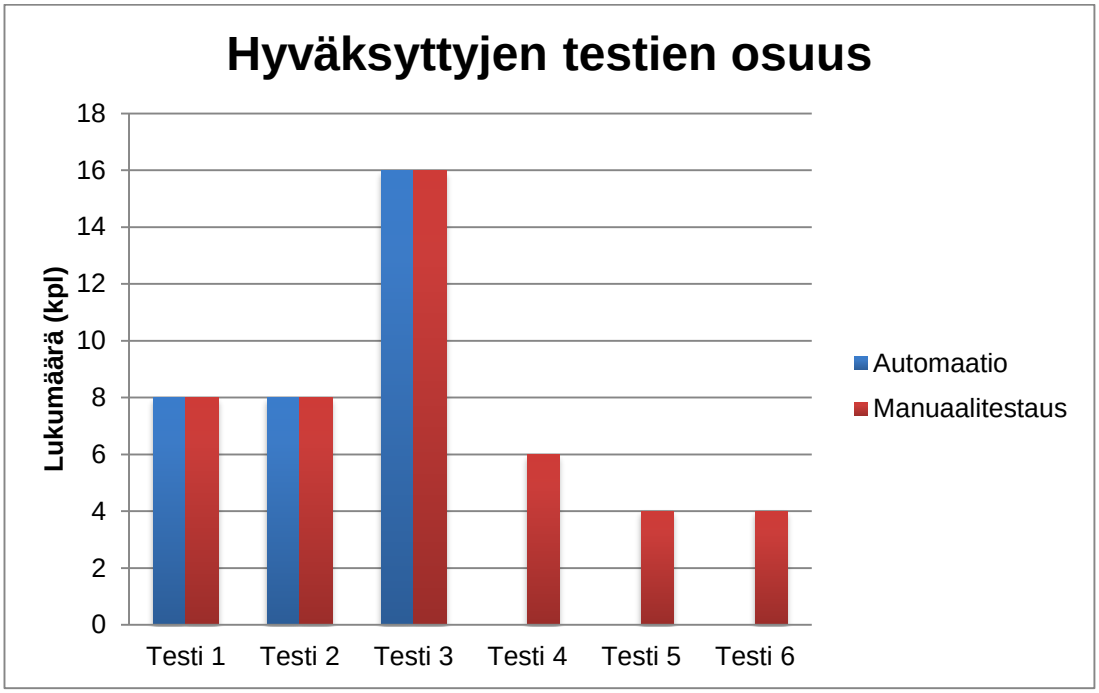
Luvussa 2.1 on mainittuna, että ohjelmistoja testataan useiden eri syistä. Tällaisia syitä ovat muun muassa mahdollisten virheiden ja poikkeavuuksien löytäminen ohjelmistoista sekä testiproseduureista. Virheiden löytymisen ansiosta saadaan estettyä ohjelmistojen liian aikainen käyttöönotto. Näillä kaikilla syillä on myös suoranainen vaikutus tuotettavan laitteen laatuun, sillä on erittäin tärkeää löytää virheet ja saada ne korjatuksi ennen tuotteen markkinoille viemistä. Kuvan 8 pylväsdiagrammista näkyy virheiden osuus molemmilla testimenetelmillä testattaessa.



Kuva 8. Testauksissa havaittujen virheiden osuus.

Kuvasta 8 on nähtävissä, että testit 1, 2 ja 3 eivät löytäneet lainkaan virheitä, kun taas testit 4, 5 ja 6 löysivät jonkin verran virheitä. Näistä testit 4 ja 5 löysivät molemmilla testimenetelmillä samat virheet. Näistä molemmista testeistä löydettiin kaksi virhettä. Testissä 6 oli automaattisesti testattaessa virheitä havaittu vain yksi, kun taas manuaalisesti testattuna virheitä löytyi neljä. Eroavaisuus virheiden löytymisessä johtui siitä, että automaatio lopetti testauksen suorittamisen kesken, havaitessaan siellä yhden virheen ja merkitsi koko testitapaukselle tulokseksi hylätyn. Näiden testauksista löydetyiden virheiden syyt on kerrottu tarkemmin kappaleessa 5.3.

Mahdollisten virheiden löytymisen lisäksi teorian kappaleessa 2.1 on kerrottu, että ohjelmistoja testataan, jotta saadaan varmuus siitä, että ohjelmistotuotteesta tulee toivotun kaltainen, sen ominaisuudet toimivat niin kuin niiden on tarkoitus toimia, ja se toteuttaa sille ennalta määriteltyjä ominaisuuksia. Näiden testauksissa läpimenneiden testien osuudet on esitelty kuvassa 9.



Kuva 9. Hyväksyttyjen testien osuus suoritetuissa testauksissa.

Kuvan 9 pylväsdiagrammista on havaittavissa, että jokainen suoritetuista testeistä sisälsi hyväksytysti läpi menneitä testikohtia. Testeistä testit 1, 2 ja 3 sisälsivät kaikki saman verran hyväksyttyjä testikohtia, kun taas testeissä 4, 5 ja 6 oli havaittavissa virheiden lisäksi myös hyväksyttyjä testikohtia vain manuaalisesti testattuna.

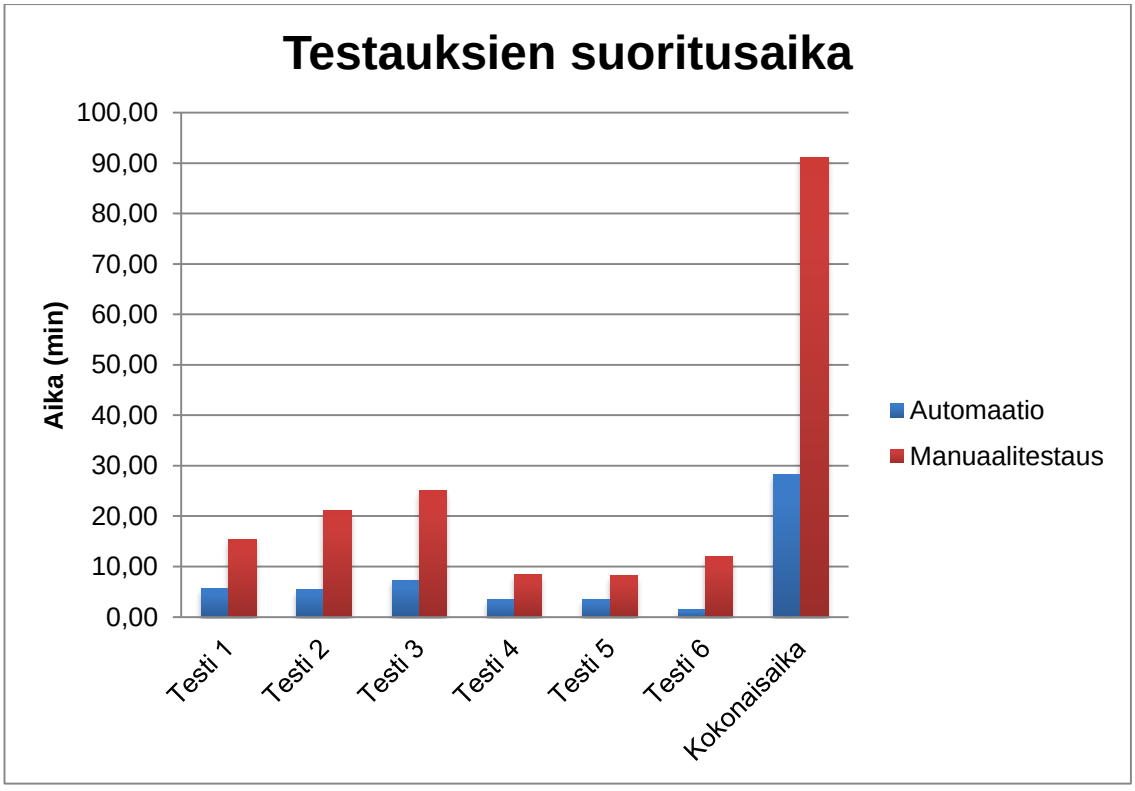
Näissä kahdessa pylväsdiagrammissa (kuvat 8 ja 9) tulosten vertailussa on testikohtien osuudet määritelty niin, että hälytyksien aktivoituminen, eskaloituminen ja poistuminen ovat kaikki omia kohtiansa, mutta audiovisuaaliset kohdat on luokiteltu tässä tapauksessa yhdeksi yhteiseksi kohdaksi. Tällaisia audiovisuaalisia eli nähtäviä ja kuultavia verifiointia tarvitsevia seikkoja ovat muun muassa hälytysten havaitseminen potilasmonitorin viestikentässä, parametrivalikossa, sekä hälytyksen äänen kuuleminen ja merkkivalon näkeminen. Luokittelun avulla saatiin rajattua tulosten vertailua varten näitä verifioitavia audiovisuaalisia kohtia, jotka kaikki olivat näissä testeissä saman isomman hyväksynnän tai mahdollisen hylkääntymisen kohteena.

5.2 Aikataululliset eroavaisuudet

Testeissä aikataulullisia eroavaisuuksia oli havaittavissa jonkin verran, esimerkiksi hälytyksien aktivoitumiseen mennyt aika oli automaatiotestauksissa mitattuna huomatta-

vasti tarkempi kuin manuaalisesti tehdyissä testeissä on saatu mitattua. Esimerkiksi testissä 1 on mitattu, kuinka kauan menee aikaa siitä, kun potilassimulaattorilla simuloitu HR raja-arvon yli menevä arvo aiheuttaa HR high hälytyksen potilasmonitorissa. Toisin sanoen miten nopeasti hälytys aktivoituu siitä, kun potilassimulaattorilla on simuloitu HR:lle uusi arvo ja sen arvo alkaa potilasmonitorin näytöllä muuttua. Automaatiolla mitattiin, että hälytyksen aktivoitumiseen oli tässä testitapauksessa mennyt aikaa 3 millisekuntia, puolestaan manuaalisesti mitattuna tämä viive oli ollut 46 sekuntia. Havaittavissa on siis, että tämä automaatiolla mitattu aika on huomattavan paljon tarkempi kuin manuaalisesti mitattu. Syynä tähän on se, että silloin kun testit tehdään manuaalisesti, menee testaajalla aina jonkin verran aikaa ennen kuin hän on havainnut hälytyksen potilasmonitorissa ja samalla hetkellä painanut sekuntikellon pois päällä. Tämä suhteellisen pieni viive johtuu ihmisen reaktionopeudesta.

Testauksiin kulunutta suoritusaikaa vertaillessa on nähtävissä, että automaatiolla suoritetut testitapaukset saatiin tehtyä huomattavan paljon nopeammin kuin manuaalisesti testattuna. Kuvassa 10 on esiteltynä kaikkiin testeihin kulunut aika, sekä lisäksi kaikkien kuuden testin suorittamisen kokonaisaika.



Kuva 10. Testauksien suoritusaika.

Pylväsdiagrammista (kuva 10) on nähtävissä, että automaatiotestauksessa näiden kaikkien yksittäisen testin suorittamiseen on kulunut aikaa alle kymmenen minuuttia, kun taas manuaalisesti testattuna vain kahden testin suorittaminen on sujunut alle kymmenessä minuutissa. Yleisesti katsoen jokaisessa kuudessa testissä on manuaalisesti testatessa aikaa kulunut vähintään kaksinkertaisesti verrattuna automaatiolla suoritettuihin testiaikoihin.

Kokonaisaikaa tarkastellessa on huomattavissa, että automaattisesti testatessa näiden kaikkien kuuden testin suorittamiseen oli kulunut aikaa noin 28 minuuttia, kun taas manuaalisesti testien tekemiseen aikaa kului yhteensä noin 91 minuuttia. Manuaalisesti testattuna aikaa näiden kaikkien testien suorittamiseen on mennyt yli kolminkertaisesti verrattuna siihen mitä automaatiolla on aikaa kulunut niiden suorittamiseen. Suhteellisen eron näiden testien suoritusajoista pystyy laskemaan kaavan 1 perusteella.

$$x = \frac{t_m}{t_a} \tag{1}$$

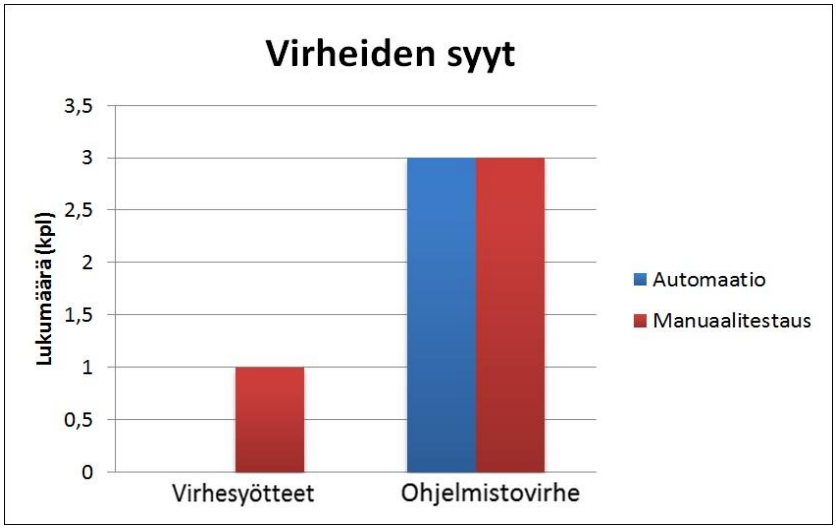
Manuaalitestauksen ja automaattisesti testattujen testien suhteellinen ero voidaan laskea kaavan mukaisesti: kaavassa x kuvaa menetelmien suhteellista eroa kun, t_m on manuaaliseen testaamiseen kulunut aika, ja t_a on automaatiotestiin kulunut aika.

Kaikkein suurin ajallinen ero testien suoritusajoissa on testi 3:ssa, sillä automaattisesti tämän koko testin suorittamiseen on kulunut aikaa 7,34 minuuttia, kun taas manuaalisesti aikaa on mennyt 25,06 minuuttia. Suhteellinen ero testatussa testissä näillä kahdella eri menetelmällä on 3,4-kertainen. Puolestaan pienin ajallinen suoritusero on testi 5:n suorittamisessa, automaattisesti testattuna aikaa on mennyt 3,43 minuuttia ja manuaalisesti testattuna aikaa on kulunut 8,22 minuuttia. Suhteellinen ero on testin suorittamisessa noin 2,4-kertainen. Koska testiautomaatio suoriutui näistä kaikista suoritetuista testeistä näin paljon nopeammin, on näiden tulosten perusteella kannattavaa vapauttaa manuaalitestaaja muihin tehtäviin.

Testauksien suoritusaikoihin vaikuttaa myös se, että testitapauksia testattaessa tulee kaikki havaitut tulokset kirjoittaa jokaiseen verifioitavaan testikohtaan. Kuten kappaleessa 4.4 on esitelty, manuaalisesti tämä kirjaaminen sujuu huomattavasti hitaammin, sillä testaajan tulee jokaisessa verifioitavassa kohdassa itse miettiä, mitkä asiat ovat tärkeitä kirjoittaa tuloksiin ja mitkä olisivat turhia kommentteja.

5.3 Testauksissa havaitut virheet

Kuten aiemmin esitellystä kuvasta 8 on nähtävissä, sekä testiautomaatiolla että manuaalisesti suoritetuista testeistä samat kolme testitapausta testit 4, 5 ja 6 eivät menneet hyväksytysti läpi. Luvussa 3.3.1 on mainittu, että ohjelmistotestauksessa voi tulla vastaan useita erilaisia virhetilanteita missä tahansa vaiheessa ohjelmistokehitystä. Tällaisia syitä voivat siis olla esimerkiksi ohjelmistosta löytyvät virheet, datavika tai virhesyötteet. Kuvasta 11 näkyy, millaisia virheitä tässä tutkimuksessa suoritetuista testeistä löydettiin.



Kuva 11. Testauksissa havaittujen virheiden syyt.

Kuvan 11 pylväsdiagrammista on nähtävissä, näistä testatuista testeistä löytyi vain kaksi syytä näihin löytyneisiin virheisiin. Ohjelmistovirheitä sisältäneitä testejä löytyi yhteensä kolme kappaletta, sekä automaatiolla että manuaalisesti suoritetuista testeistä. Näissä kaikissa kolmessa hylätyssä testissä syynä tulokseen hylätty oli se, että testin sisältämissä testitapauksissa aiheutettu hälytys ei aktivoitunut vaatimuksissa määritetyn ajan puitteissa. Kyseisten testien vaatimuksissa on sanottu, että hälytyksen tulee aktivoitua 0 s ± toleranssi (2 s) kuluessa siitä, kun potilassimulaattorilla parametrille simuloitu uusi arvo alkaa muuttua potilasmonitorissa, mutta näissä kaikissa kolmessa hylätyssä testissä aikaa hälytyksen aktivoitumiseen meni neljästä sekunnista viiteen sekuntiin. Lisäksi automaatiolla testattaessa myös kahdessa testitapauksessa hälytyksen eskaloituminen ja yhdessä testitapauksessa hälytyksen poistuminen hylättiin, koska automaatio jätti testin tekemisen kesken eli tulokseksi näihin kaikkiin loppuihin kohtiin tuli hylätty, vaikka niitä ei todellisuudessa oltu edes testattu. Nämä ohjelmistosta

löydetyt ohjelmistovirheet ovat riski-analyysin mukaan riskejä potilaalle, joten niiden löytäminen ja korjaaminen on erittäin tärkeää.

Löydettyjen ohjelmistovirheiden lisäksi suoritetuista testauksista löytyi yhdestä testistä, testi 3:sta virhesyötteeseen liittyvä virhetilanne, joka oli havaittavissa vain manuaalisesti suoritetussa testissä. Testissä testitapausten otsikoinnit olivat suunniteltu huonosti, sillä ne aiheuttivat väärinymmärryksen manuaaliseen testimenetelmään. Nämä väärinymmärryksen aiheuttaneet testitapaukset selvisivät tarkemmin vasta sen jälkeen, kun kysyi automaation läpi ajaneelta henkilöltä lisätietoja testin suoritustavasta.

6 Yhteenveto ja johtopäätökset

Tässä insinöörityössä teoriaosuudessa käytiin läpi ohjelmistotestauksen perusteita, kolme erilaista ohjelmistokehityksen prosessimallia. Työssä perehdyttiin manuaaliseen ja automatisoituun testaukseen, sekä lääkinnällisiä laitteita koskeviin laadullisiin vaatimuksiin, ja pohdittiin, millaisia virheitä testauksissa helposti löytyy ja miten niitä voitaisiin välttää. Näiden teoriassa läpi käytyjen asioiden ansiosta saatiin pohjatietoa työn toteutukselle.

Tässä insinöörityössä tavoitteena oli selvittää, millaisia eroavaisuuksia oli havaittavissa, kun testejä suoritetiin automaattisesti ja manuaalisesti testattuna. Tutkimuksessa selvitettiin, kuinka suuri ajallinen ero näillä kahdella eri testimenetelmällä on, onko havaittavissa laadullisia eroavaisuuksia, löytyykö niillä erilaisia virheitä testatuista testeistä ja lisäksi perehdyttiin riskeihin. Koska automaatiolla suoritetut testaukset olivat jo etukäteen suoritettu, niin tässä tutkimuksessa osittain pyrittiin myös varmentamaan, ettei manuaalisella testimenetelmällä pystytä parempaan.

Tässä työssä läpikäydyissä testimenetelmissä, automatisoidussa ja manuaalitestauksessa havaittiin kaikissa suoritetuissa testeissä täsmälleen samat virheet, joten pelkästään näitä tutkimustuloksia tarkastellessa automaatiolla testaaminen on yritykselle kannattavampaa. Saatujen tuloksien perusteella automaatiolla testattaessa testiprosessi on huomattavasti nopeampi, jolloin se myös vähentää yrityksen kustannuksia, sillä testaaja voi tehdä samaan aikaan jotain muita testejä manuaalisesti, esimerkiksi sellaisia, joita ei ole vielä saatu automatisoituun muotoon. Koska tutkimuksessa suoritettuja testejä on vain kuusi kappaletta, siitä johtuen ei voida vielä kuitenkaan sanoa luotettavia

tuloksia näiden kahden menetelmän eroavaisuuksista. Tämän tutkimuksen perusteella voidaankin siis oikeastaan luottaa vain tässä työssä testattujen testien automaatiolla suoritettavaan regressiotestaukseen ja siitä saataviin tuloksiin.

Kuitenkin nämä molemmat testimenetelmät, sekä manuaalitestaus, että automaatiolla suoritettavat testit ovat yritykselle arvokkaita. Molemmilla testimenetelmillä saadaan hieman erilaisia tuloksia verifioitaville testitapauksille. Automaatiolla pystytään suorittamaan testejä huomattavan paljon nopeammin kuin mitä manuaalitestauksessa samojen testien suorittamiseen aikaa kuluisi. Kuitenkin silloin, kun jotain testiä suoritetaan automaatiolla, automaatio pystyy verifioimaan vain tiettyä asiaa kerrallaan, sillä se ei kykene rinnakkaiseen toimintaan. Toisaalta manuaalitestauksessa testaaja voi havaita testiä suorittaessaan useita erilaisia poikkeavuuksia, kuten jos parametrin reaaliaikaista mittausdataa näyttävä käyrä potilasmonitorin näytöllä katkeaa tai jos potilasmonitoriin tulee jokin yllättävä ylimääräinen hälytystila. Tällaisissa tilanteissa testaaja pystyy tekemään niistä tarvittavat virhetilailmoitukset, jolloin ohjelmistokehittäjät voivat aloittaa selvittämään, mistä löydetty ongelma mahdollisesti johtuu. Lisäksi kaikille testitapauksille ei välttämättä ole edes kannattavaa tehdä automatisoitua testimuotoa, sillä jos jokin testi suoritetaan vain muutamaan otteeseen, niin silloin sen automatisointi veisi vain turhaan aikaa.

Laajempaa tutkimusta automaation ja manuaalitestauksen eroista pystyy halutessa helposti jatkamaan tästä insinöörityöstä. Sen voisi tehdä esimerkiksi siten, että manuaalisesti suoritettu testi tehtäisiin automaatiolle testattavaksi ja vertailtaisiin siinä syntyviä eroja, tai mikäli aikataulullisesti onnistuisi, olisi vertailun kannalta suositeltavaa ottaa huomattavasti enemmän ja etenkin useita erilaisia testitapauksia testattavaksi, eli siis myös muita kuin pelkkiä hälytystestejä. Mikäli edellä mainitusti toimittaisiin, saataisiin vertailtua huomattavasti laajemmin näitä kahta eri menetelmää. Sen seurauksena saataisiin myös luotettavampia tuloksia näiden kahden menetelmän eroavaisuuksista. Manuaalisen ja automaatiotestaukseen käytetyn ajan laajemmalla vertailulla voitaisiin arvioida testiautomaation kustannustehokkuutta. Tämä onnistuisi, kun selvittäisi manuaalisen ja automaatiotestin kirjoittamiseen ja ylläpitoon kuluneen ajan, muut investoinnit, sekä sen, kuinka monta kertaa testi on ajettu regressio- tai verifikaatiotestinä.

Tätä insinöörityötä voidaan myös mahdollisesti hyödyntää esimerkiksi uusien työntekijöiden kouluttamisen tukena. Työssä on esiteltynä jonkin verran työkaluja, joita Yritys X:n ohjelmistotestauksessa käytetään, molemmat yrityksessä käytössä olevat testime-

netelmät, ohjelmistotestauksessa mahdollisesti odotettavissa olevat virhetilanteet sekä millä tavalla virheitä pystytäisiin välttämään. Työn ansiosta uusi työntekijä saa samalla myös käsityksen siitä, miten tärkeässä osassa kehitettävien ja valmistettavien laitteiden korkealaatuisuus yrityksessä on, mitä laadulla oikeastaan tarkoitetaan ja miten riskinhallinnasta hyödytään. Lisäksi uusi työntekijä ymmärtäisi, miten tärkeää myös testauksista havaittujen tulosten kirjaaminen huolellisesti ja selkeästi on tuotteen laadun kannalta.

Lähteet

1 Kasurinen Jussi Pekka. (2013). Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo.

2 Katara, Vuori, Jääskeläinen. 2013. Ohjelmistojen testaus. Tampereen teknillinen yliopisto. Verkkodokumentti. <http://www.cs.tut.fi/~testaus/s2013/luennot/TIE-21200_2013.pdf>. 23.08.2013. Luettu 17.01.2016.

3 Korjus Nina. 2015. Ohjelmistotestauksen perusteet. Noppa. Verkkodokumentti. 2015. <https://www.google.fi/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwi_6dGSnrXKAhVKEiwKHYASBxkQFggcMAA&url=https%3A%2F%2Fnoppa.lut.fi%2Fnoppa%2Fopintojakso%2Fct60a4150%2Fmateriaali%2Fohjelmistotestauksen_perusteet_-manuaali.pdf&usg=AFQjCNHr-oLsPDs8viNuJ_Wo1KjN986eRQ>. 16.02.2015. Luettu 19.01.2016.

4 What is Manual Testing? Software Testing Class. Verkkodokumentti. 2016. <http://www.softwaretestingclass.com/what-is-manual-testing/>. Luettu 15.01.2016.

5 Robot Framework. Verkkodokumentti. <http://robotframework.org/>. Luettu 18.01.2016.

6 Lehtonen Lasse. 2015. Johdatus Potilasturvallisuuteen. Duodecim oppiportti. Verkkokurssi. <http://www.oppiportti.fi/op/dvk00034>. Luettu 18.2.2016.

7 Ståhlberg Tom. 2015. Terveydenhuollon laitteiden lakisääteiset määräykset kansainvälisillä markkinoilla. Tekes. Verkkodokumentti. <https://www.tekes.fi/globalassets/julkaisut/terveydenhuollon_laitteiden_lakisaateiset_maaraykset_opas.pdf>. 14.01.2015. Luettu 21.01.2016.

8 SFS. (2010). SFS-käsikirja 133 CE-merkintä perustiedot. Helsinki: Suomen Standardisoimisliitto SFS ry.

9 Standardi tutuksi. Suomen Standardisoimisliitto SFS ry. Verkkodokumentti. <http://www.sfs.fi/julkaisut_ja_palvelut/standardi_tutuksi>. Luettu 19.01.2016.

10 SFS, EN, ISO? Suomen Standardisoimisliitto SFS ry. Verkkodokumentti. <http://www.sfs.fi/julkaisut_ja_palvelut/standardi_tutuksi/sfs_en_iso>. Luettu 19.01.2016.

11 Hiltunen, Linko, Hemminki, Hägg, Järvenpää, Saarinen, Simonen & Kärhä. (2011). Laadukkaan mittaamisen perusteet. Espoo: Mittatekniikan keskus.

12 Laatua terveydenhuollon palveluihin. Suomen Standardisoimisliitto SFS ry. Verkkodokumentti. 2013.

<http://www.sfs.fi/ajankohtaista/uutiset/laatua_terveydenhuollon_palveluihin.1836.news>. Luettu 10.04.2016.

13 ISO 9001:2015. Suomen Standardisoimisliitto SFS ry. Verkkodokumentti. 2015. <http://www.sfs.fi/julkaisut_ja_palvelut/tuotteet_valokeilassa/iso_9000_laadunhallinta/iso_9001_2015>. Luettu 18.01.2016.

14 Terveydenhuollon laitteiden laatustandardin suomennos julkaistaan kesäkuussa. Suomen Standardisoimisliitto SFS ry. Verkkodokumentti. 2016. <http://www.sfs.fi/ajankohtaista/uutiset/terveydenhuollon_laitteiden_laatustandardin_suomennos_julkaistaan_kesakuussa.3638.news>. Luettu 13.04.2016.

15 ISO 13485 – Medical devices. International Organization for Standardization. Verkkodokumentti. <http://www.iso.org/iso/iso13485>. Luettu 13.04.2016

16 Laitila Erkki. (2011). Näin paikannan tyypilliset ohjelmistoviat: Päättelytekniikka ohjelmisto-ongelmien selvittämiseen! Jyväskylä: Uusi IT.

17 Grove Consultants software testing. Version 2_5. ISTQB: Advanced Level Certificate Core Module.

18 Katara, Vuori, Jääskeläinen. 2015. Ohjelmistojen testaus. Tampereen teknillinen yliopisto. Verkkodokumentti. <http://www.cs.tut.fi/~testaus/s2015/luennot/TIE-21204_2015.pdf>. 17.08.2015. Luettu 07.02.2016.

19 Toppinen-Tanner, Ahola. (2012). Kaikkea Stressistä. Helsinki: Työterveyslaitos cop.

20 Saarijärvi Päivi. (2015). Väsymys työelämässä: riskit ja hallinta. Helsinki: Neirol.

21 What is Risk based testing? ISTQB Exam Certification. Verkkodokumentti. 2016. <http://istqbexamcertification.com/what-is-risk-based-testing/>. Luettu 23.04.2016