Ilpo Pöyhönen

Lääkintälaitteiden ohjelmistot

Suunnittelun kehityskohteita vesiputous- ja XP-mallin näkökulmasta

VTT TIEDOTTEITA – RESEARCH NOTES 2320

Lääkintälaitteiden ohjelmistot
Suunnittelun kehityskohteita vesiputous- ja XP-mallin näkökulmasta

Ilpo Pöyhönen
VTT Tuotteet ja tuotanto

ISBN 951–38–6766–8 (nid.)
ISSN 1235–0605 (nid.)

ISBN 951–38–6767–6 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1455–0865 (URL: http://www.vtt.fi/inf/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 3, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 3, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 3, P.O.Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax + 358 20 722 4374

VTT, Tekniikankatu 1, PL 1300, 33101 TAMPERE
puh. vaihde 020 722 111, faksi 020 722 3365

VTT, Tekniikankatu 1, PB 1300, 33101 TAMMERFORS
tel. växel 020 722 111, fax 020 722 3365

VTT Technical Research Centre of Finland, Tekniikankatu 1, P.O. Box 1300, FI-33101 TAMPERE, Finland
phone internat. +358 20 722 111, fax +358 20 722 3365

Toimitus Anni Kääriäinen

Otamedia Oy, Espoo 2006

Pöyhönen, Ilpo. Lääkintälaitteiden ohjelmistot. Suunnittelun kehityskohteita vesiputous- ja XP-mallin näkökulmasta [Software of medical devices. Targets fot development of the design from the point of view of the waterfall and XP model]. Espoo 2005. VTT Tiedotteita – Research Notes 2320. 61 s. + liitt. 2 s.

Avainsanat medical devices, software engineering, safety related software, cost-effectiveness, design, requirements, specifications, traceability, risk management, risk-informed life cycle models

Tiivistelmä

Ohjelmistojen merkitys potilaan hoidossa kasvaa jatkuvasti. Ohjelmistot ohjaavat lääkintälaitteen toimintoja tai laskevat ja diagnosoivat mitattua potilasinformaatiota monimutkaisten algoritmien avulla. Tästä on seurannut se, että myös päätökset potilaan hoidosta tai hoitamatta jättämisestä perustuvat lisääntyvässä määrin ohjelmistojen laskemiin tuloksiin (laboratoriotulokset, hoitosuunnitelmat, diagnosointi ja tehohoito).

Ohjelmistovirheet ovat luonteeltaan systemaattisia, mikä voi aiheuttaa useiden potilaiden altistamisen haitalliselle tapahtumalle ennen kuin ohjelmistovirhe havaitaan. Ohjelmistojen turvallisuuden ja vaatimustenmukaisuuden arviointi on sen tähden avainasemassa.

Ohjelmistotuotantoprosessi on lääkintälaitteiden ohjelmistokehityksen kulmakivi. Prosessin toiminta ja tuotokset ratkaisevat sen, täyttääkö ohjelmisto valmiissa tuotteessa sille asetetut viranomais- ja suorituskykyvaatimukset ja voidaanko suunnittelu tehdä kustannustehokkaasti ja toistettavasti.

Kehittämiskohteiden valinta ja yhdenmukaistettujen standardien soveltuvuuden arviointi omaan toimintaan edellyttää yritykseltä kriittistä ja perinpohjaista tarkastelua. Tässä julkaisussa esitetään keinoja, joiden avulla yrityksissä voidaan kehittää yrityskohtaista ohjelmistotuotantoprosessia.

Tarkastelun kohteena ovat standardien ja riskienhallinnan tarjoamaan tukeen perustuvat menettelytavat, ja kehittämiskohteet kuvataan perinteisen vesiputousmallin näkökulmasta. Lisäksi tarkastellaan lyhyesti myös XP-mallin soveltuvuutta lääkintälaitteiden ohjelmistojen suunnitteluun.

Esitettyjen keinojen tarkoituksena on kehittää yrityksen suunnitteluprosessia siten, että suunnittelussa tunnistetaan tarvittavat lakisääteiset vaatimukset ja kukin suunnitteluvaihe tuottaa myös suunnitteludokumentaation, jota tuotteen hyväksyntäprosessi edellyttää.

Pöyhönen, Ilpo. Lääkintälaitteiden ohjelmistot. Suunnittelun kehityskohteita vesiputous- ja XP-mallin näkökulmasta [Software of medical devices. Targets fot development of the design from the point of view of the waterfall and XP model]. Espoo 2005. VTT Tiedotteita – Research Notes 2320. 61 p. + app. 2 p.

Keywords medical devices, software engineering, safety related software, cost-effectiveness, design, requirements, specifications, traceability, risk management, risk-informed life cycle models

Abstract

The significance of the software in the patient's care increases continuously. The software controls the functions of the medical device or they calculate and diagnose measured patient information using complex algorithms. Due to this the decisions on the patient's care or leaving without care are increasingly based on the results calculated by the software (laboratory researchs, treatment plans, diagnosing and intensive care).

The software errors have a systematic character which can cause the exposure of several patients to a harmful event before the software errors are detected. Therefore the evaluation of the safety and compliance of requirements of software is in the key position.

The software engineering process is the cornerstone of the medical device software development. The operation of the process and results determine the fact whether the software meets the requlatory and performance requirements in the final product and can a design be made cost-effectively and repeatably.

Choice of targets for development and evaluation of the suitability of harmonized standards to own operation requires a critical and thorough examination of the company. In this report means are presented the means which the companies can use to develop a company-specific software engineering process.

The subject of the examination are procedures which are based on the support offered by the standards and the risk management and the target for development are described from the point of view of the traditional waterfall model. Also, the suitability of the XP model for the design of the software of medical devices is briefly examined.

The purpose of presented methods is to develop the design process of the company so that in the design the necessary requlatory requirements are identified and each design phase produces the design documentation which is required during approval process of the product.

Alkusanat

Julkaisu kuuluu osana Trusted Software Technology -hankkeeseen (TRUST), jossa tutkitaan uusia menettelytapoja ratkoa turvallisuuskriittisiin ohjelmistoihin liittyviä ongelmia. Menettelytapojen avulla pyritään edistämään riskitietoisen ohjelmistotuotantoprosessin kehittämistä.

Kiitän Planmecan henkilökuntaa, erityisesti Kustaa Nyholmia, Jaana Sievästä ja Petri Peräsaarta hyödyllisistä ja kriittisistä kommenteista.

Tekijä

Sisällysluettelo

Symbolit ja terminologia

COTS-komponentti	Commercial Off The Self components. Valmiit kaupalliset ohjelmistokomponentit tai ohjelmat, jotka integroidaan omaan sovellukseen.
Default	Oletusarvo tai oletustila. Alkutila, joka järjestelmälle asetetaan käynnistysvaiheessa tai tilasta toiseen siirryttäessä. Alkutila edellyttää, että parametreihin ja syötteisiin asetetaan toiminnan kannalta sopivat ja turvalliset arvot. Lääkintälaitteissa esimerkiksi hälytykset voivat olla oletusarvoisesti joko päällä tai pois. Hälytysrajat voidaan oletusarvoisesti asettaa esimerkiksi 10–90 %:iin mittausalueesta.
DMR	Device Master Record (http://www.ghtf.org/sg1/inventorysg1/pd_sg1_n011r17.pdf)
DSDM	Dynamic Systems Development Method (http://www.dsdm.org)
GHTF	Global Harmonized Task Force
GUI	Graphics User Interface
IEC	International Electrotechnical Commission
IEEE	The Institute of Electrical and Electronics Engineers
In-house	Yleensä termiä käytetään ohjelmistotestauksen yhteydessä. In-house-testeillä varmistetaan ohjelmiston toimivuus. Yrityksen laadunvarmistusosasto tai suunnittelijat tekevät in-house-testit ennen ohjelmiston julkistusta (http://en.wikipedia.org/wiki/Software_testing).
ISO	International Organization for Standardization
IVDD	In Vitro Diagnostic Medical Devices Directive 98/79/EC on in vitro diagnostic medical devices
EVO	Ohjelmistokehitysmalli, joka muodostuu peräkkäisistä vesiputousmalleista. Jokaisessa syklissä kehitettävään ohjelmistoon lisätään uusia ominaisuuksia. (Haikala & Märijärvi 1998.)
MDD	Medical devices Directive 93/42/EEC concerning medical devices

NB	Notified Body
Therac-25	Therac-25-tapaus on valitettava esimerkki, jossa sädehoitolaitteen ohjelmistopäivitystä ei testattu riittävästi, päivityksessä ei huomioitu ihmisen käyttäytymistä, käyttöliittymä mahdollisti virheen syntymisen eikä järjestelmän käyttöönottovaiheessa järjestetty riittävää käyttökoulutusta. Ohjelmistovirheen seurauksena kuusi ihmistä sai yliannoksen säteilyä (eri lähteissä eri tietoa, yliannoksen seurauksena kolme potilasta kuoli). Tapausta on käsitelty laajasti turvallisuuskriittisten ohjelmistojen koulutuksessa. Lähde: Nancy Leveson, Safeware: System Safety and Computers, ISBN: 0201119722.
V-malli	Ohjelmistokehitysmalli, jossa suunnittelu sidotaan tiukasti testaukseen. Jossain määrin onkin puhuttu myös testauksen V-mallista. (http://en.wikipedia.org/wiki/V_model, http://www.informatik.uni-bremen.de/uniform/gdpa/vmodel/vm.htm, http://www2.sas.com/proceedings/sugi30/141-30.pdf.)
XP	Extreme Programming
White-box	Sanan ”glass-box” synonyymi. Järjestelmä tai komponentti, jonka sisältö tai toteutus tunnetaan. (IEEE 610.12:1990.)

1. Johdanto

Ohjelmistojen rooli terveydenhuollon laitteissa kasvaa jatkuvasti. Ohjelmistot ovat aktiivisesti mukana ohjaamassa laitteen toimintoja tai laskemassa ja diagnosoimassa mitattua potilasinformaatiota monimutkaisten algoritmien avulla. Näin ollen päätökset potilaan hoidosta tai hoitamatta jättämisestä perustuvat lisääntyvässä määrin myös ohjelmistojen laskemiin tuloksiin (laboratoriotulokset, tehohoidon valvontalaitteet, kuvantamislaitteet).

Tästä syystä lääkintälaitteiden ohjelmistojen turvallisuuden arviointiin kiinnitetään yhä enemmän huomiota ja ohjelmistoille asetetaan tiukkoja suorituskyky- ja viranomaisvaatimuksia.

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

Ohjelmiston arviointi tapahtuu direktiivien (MDD, IVDD) ja standardien ISO 13485, ISO 14971 ja IEC 60601-1-4 vaatimusten pohjalta. Lisäksi IEC-työryhmässä on kehitteillä uusi standardi IEC 62304, joka kattaa vaatimukset lääkintälaitteiden ohjelmistojen suunnittelusta (IEC 60601-1-4:n soveltamisala on ohjelmistoa sisältäville sähkökäyttöisille lääkintälaitteille, kun taas IEC 62304 soveltuu edellä mainittujen lisäksi myös yksittäiselle ohjelmistolle).

Käytännössä ohjelmiston turvallisuutta ja vaatimustenmukaisuutta ei voida osoittaa valmiista ohjelmistosta, joten ohjelmistojen turvallisuuden arviointi ulotetaankin valmiista ohjelmistosta sitä tuottavan prosessin arviointiin (suunnittelu, vaiheistus, validointi sekä riskienhallinta).

Julkaisussa käsitellään suunnittelun aikaisia yksityiskohtia, joiden kehittämisellä voidaan nopeuttaa suunnittelua, parantaa ohjelmiston turvallisuutta ja vaatimustenmukaisuuden osoittamista sekä systematisoida itse ohjelmistotuotantoprosessia. Esimerkkeinä käytetään perinteistä vesiputousmallia ja myös ketterään ohjelmistokehitykseen lukeutuvaa XP-mallia.

Julkaisun tarkoituksena on kuvata lääkintälaitteen ohjelmistosuunnittelua ja sitä prosessia, jolla ohjelmistosuunnittelu saadaan toteutettua kustannustehokkaasti ja viranomaisvaatimukset täyttäen turvallisesti ja laadukkaasti.

2. Yleisimmät ongelmat

Ohjelmistojen rooli lääkintälaitteissa sekä terveydenhuollon sovelluksissa ja toimenpiteissä on lisääntynyt merkittävästi ja tuonut mukanaan joukon erilaisia ongelmia. Esimerkiksi käyttäjä–laite–ohjelmisto-rajapinta muodostuu helposti hyvin monimutkaiseksi, koska toiminnan lopputuloksen sanelevat oletusarvot, useat erilaiset esiasetukset ja alkutilanteet. Tästä syystä suunnittelu edellyttää tuekseen usean eri alan asiantuntijoista koostuvaa laajaa sidosryhmää, mitä valmistajat eivät kaikilta osin aina ole tyydyttävästi kyenneet tiedostamaan.

Vähänkin laajemman ohjelmiston kaikkien ominaisuuksien, toimintojen ja muuttujien tutkiminen kaikissa eri tilanteissa ja olosuhteissa vaatii paljon aikaa ja resursseja. Tämä johtaa usein siihen valitettavaan tilanteeseen, että ohjelmistoa ei testata tai analysoida riittävän kattavasti.

Lievimmillään ongelmat aiheuttavat valmistajalle pieniä viiveitä markkinoille saattamisessa tai käyttäjille lieviä käyttö- tai toimintahäiriöitä. Vakavimmillaan ohjelmistojen aiheuttamat toimintahäiriöt vaarantavat potilasturvallisuutta merkittävästi ja johtavat esimerkiksi hoitovirheisiin tai jopa potilaan kuolemaan.

Lakisääteisistä vaatimuksista johtuen on valmistajan kyettävä osoittamaan ohjelmiston suorituskyky ja turvallisuus asetettuihin standardeihin ja direktiiveihin nähden. Tämä on usein ensimmäinen ongelma, joka konkreettisesti liittyy lääkintälaitteiden ohjelmistoihin. Ongelma aiheuttaa viiveitä ohjelmiston markkinoille saattamisessa, kun ohjelmiston valmistaja ei kykene osoittamaan viranomaisille ohjelmiston täyttävän sille asetettuja vaatimuksia.

Viranomaisvaatimuksiin liittyvät ongelmat aiheuttavat aikataulullisia viiveitä ja lisädokumentoinnin tarvetta. Lisädokumentointi voi myös turhauttaa suunnittelijoita, kun jo tehtyä työtä joudutaan dokumentoimaan uudestaan. Käytännössä tällöin on kysymys ohjelmiston suunnitteludokumentaation puutteista, jotka aiheutuvat suunnitteluprosessin eri vaiheiden tulostietomäärityksistä. Lopputuloksena voivat olla huomattavat lisäkustannukset hyväksyntätarkastuksissa tai viiveet markkinoille saattamisessa.

Merkittävämpi ongelma on itse ohjelmiston toimintaan liittyvät suorituskyky tai turvallisuusongelmat. Ohjelmistojen kannalta on huomioitava, että ohjelmistovirheet ovat systemaattisia, mistä voi seurata useiden potilaiden piiloaltistaminen vaaralle tai riskille ennen kuin ohjelmistovirhe havaitaan. Kysymyksessä ovat tällöin suunnittelun lähtötietojen puutteellisuus, puutteelliset testaukset, validoinnit tai riskienhallinnan kattavuus.

Edelleen on käytössä myös suunnitteluprosesseja, joissa ohjelmistoja kehitetään ilman muodollista dokumentoitua ohjelmistotuotannon elinkaarta. Ohjelmiston kehitys tapahtuu ilman huolellista suunnittelua ja riskienhallintaa periaatteella ”määrittele, kokeile, käytä”. Menettelyllä voidaan saada erittäin innovatiivisia ja toimivia tuotteita, mutta menettelystä seuraa väistämättä vakavia ongelmia markkinoille saattamisen yhteydessä. Suunnittelumenetelmästä seuraa väistämättä seuraavia haittoja:

- hitaus ja kalleus, koska systemaattinen tekotapa puuttuu
- valmistuminen vaikeasti suunniteltavissa
- yksittäiset ohjelmistomuutokset vaikeita, koska aiempia suunnitteluratkaisuja ja perusteita ei ole dokumentoitu
- muutostenhallinta vaikeaa, koska suunnittelu ei sisällä ylläpitoa tai erillistä dokumentoitua muutostenhallintaprosessia
- epävarmuus – kaikkia vaihtoehtoja ei pystytä testaamaan, ongelmia voi jäädä piileviksi
- ei kyetä osoittamaan ohjelmiston vaatimustenmukaisuutta (puutteita potilasturvallisuudessa, riskianalyysissä, suunnitteludokumentaatiossa)
- viranomaishyväksynnät viivästyvät
- koodin vakaus ja laatu kyseenalaista.

Edellä mainitussa suunnittelumenetelmässä suurimmat puutteet ovat varmaan prosessin kustannustehokkuudessa, toistettavuudessa eri tuotteiden kesken sekä suunnitteludokumentaation tuottamisessa. Asiakasvaatimusten muuttaminen teknisiksi toteutettaviksi vaatimuksiksi saattaa myös aiheuttaa ongelmia.

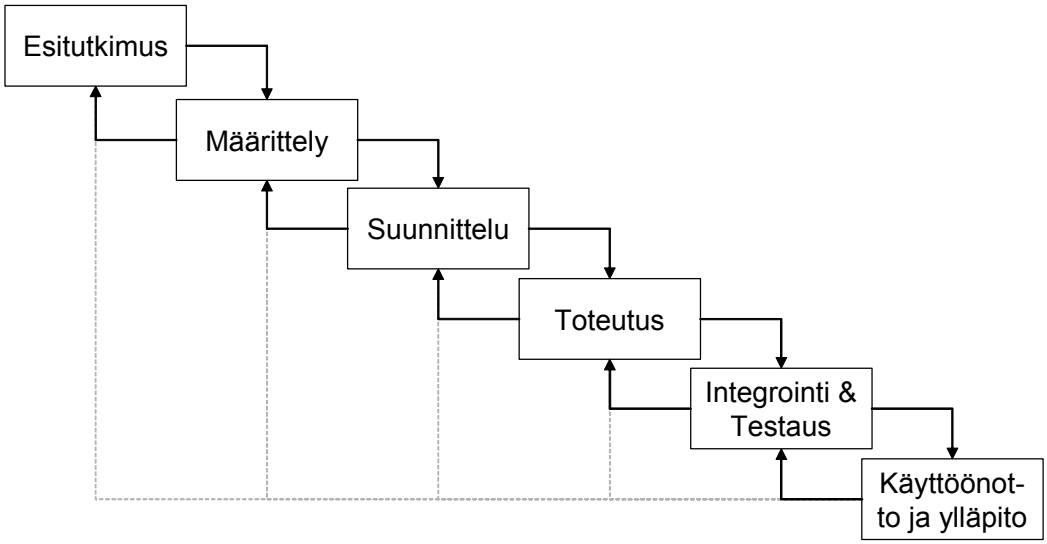
Perinteinen vesiputousmalli tai ketterät ohjelmistotuotantomenetelmät eivät välttämättä sellaisenaan tarjoa ihanteellista ratkaisumallia lääketieteellisten laitteiden ohjelmistosuunnitteluun, koska menetelmät eivät sisällä lääkintälaitteille asetettuja viranomaisvaatimuksia tai kattavaa riskienhallintaa.

3. Ohjelmistosuunnittelu

Ohjelmistosuunnittelu koostuu useista erillisistä peräkkäisistä toiminnoista, joissa edellisen toiminnon tuotosta käytetään aina seuraavan toiminnon lähtötietona. Menettelytavasta onkin ruvettu käyttämään nimeä ohjelmistotuotannon vaihejakomalli.

Ohjelmistosuunnittelun vaihejakomalli on ohjeistettu prosessi, joka määrittelee kullekin elinkaaren vaiheelle ominaiset tehtävät ja dokumentointivaatimukset. Prosessi alkaa ohjelmistolta vaadittavien ominaisuuksien tunnistamisesta ja dokumentoinnista ja päättyy ohjelmiston kelpuutukseen kaikkien vaadittujen ominaisuuksien suhteen. Kelpuutus kattaa myös ohjelmiston julkaisun (release) ja tuotannollistamissuunnitelman. Vaihejakomalleja on useita eri malleja, kuten perinteinen vesiputousmalli, spiraalimalli, EVO tai viimeisimpinä tulleet ketterät ohjelmistotuotantomenetelmät.

Vaihejakomalli määrittelee tarkastuspisteet, joissa katselmoidaan eli varmistetaan, että ohjelmistotuotantoprosessi voi edetä seuraavaan vaiheeseen (Haikala & Märijärvi 1998). Vaiheiden määrä ja vaihekohtaiset tehtävät riippuvat kulloinkin suunniteltavasta ohjelmistosta, sovellusalueesta sekä asiakas- ja viranomaisvaatimuksista.



Kuva 1. Perinteinen vesiputousmalli.

Perinteinen vesiputousmalli sisältää kuvan 1 mukaiset vaiheet, jotka sisältävät seuraavanlaisia toimintoja (Haikala & Märijärvi 1998):

– Esitutkimus, jossa kartoitetaan ja tutkitaan asiakkaan tarpeet ja haetaan vastausta kysymykseen ”miksi ohjelmisto tai järjestelmä tulisi tehdä?” Asiakasvaatimuksia

selvitetään haastatteluiden ja aivoriihen avulla. Vaihe tuottaa asiakasvaatimusdokumentin.

- Määrittely, jossa esitutkimusvaiheen tieto jalostetaan analyysien ja pohdintojen jälkeen järjestelmävaatimuksiksi.
- Suunnittelu, jossa järjestelmäsuunnittelu voidaan jakaa useisiin eri tasoihin. Tässä vaiheessa tulevat jo mukaan mahdolliset alihankinnat tai COTS-komponenttien ostot ja otetaan kantaa myös suunnittelutyökaluihin. Mikäli ostotoiminta tapahtuu erillisen osto-osaston kautta, tulee suunnittelutiimin määritellä vaatimukset osto-osastolle.
- Toteutus, jossa määriteltyjen vaatimusten ja suunnitteludokumentaation mukaisesti laaditaan ensimmäinen ”toimiva ohjelmakoodi”, joka voi tarkoittaa ensimmäistä virheetöntä käännöstä tai ensimmäistä toimivaa kokeiluversiota. Ohjelmiston mukana seuraavan dokumentaation tuottaminen tulisi aloittaa viimeistään tässä vaiheessa.
- Integrointi ja testaus, jossa ohjelmistosta haetaan virheitä ennalta laaditun suunnitelman mukaisesti. Testaus kattaa kaikki ohjelmamoduulit, ohjelmamoduulien keskinäisen kommunikoinnin sekä järjestelmätestauksen.
- Käyttöönotto ja ylläpito, jossa ohjelmisto asennetaan, otetaan käyttöön ja annetaan käyttökoulutus. Tässä vaiheessa ohjelmistoon voidaan asentaa myös uusia ominaisuuksia ja korjataan käytön aikana havaittuja virheitä.

Vaiheiden välillä joudutaan väistämättä tekemään muutamia iteratiivisia kierroksia, kun edellisen vaiheen tuotosta joudutaan tarkentamaan tai seuraavassa vaiheessa todetaan ko. ratkaisun olevan mahdoton toteuttaa.

Standardissa ISO/IEC 12207 kuvataan eräs malli ohjelmistosuunnittelun vaihejakomallille. Ohjelmistotuotantoa sekä vaatimuksia eri vaiheille käsitellään laajasti myös IEEE:n standardisarjassa (http://standards.ieee.org/catalog/olis/se.html).

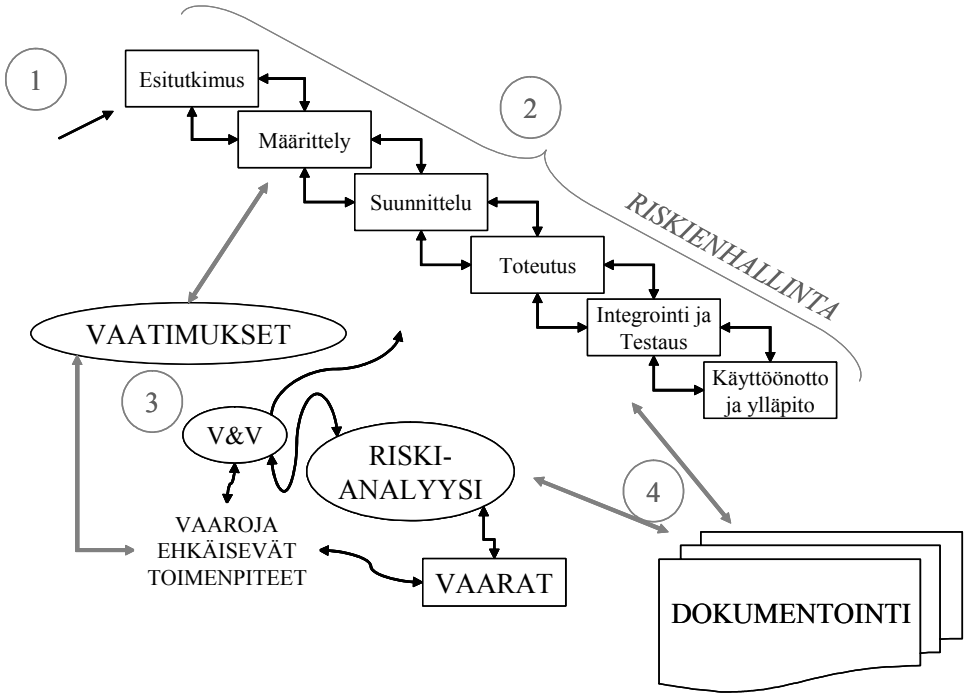
3.1 Lääkintälaitteiden ohjelmistosuunnittelu

Käytännössä ohjelmiston turvallisuutta ja vaatimustenmukaisuutta ei voida osoittaa valmiista ohjelmistosta. Tämän takia ohjelmistojen turvallisuuden arviointi ulotetaankin valmiista ohjelmistosta sitä tuottavan prosessin arviointiin (suunnittelu, vaiheistus, validointi sekä riskienhallinta). Tämä on huomioitava silloin, kun ohjelmistoille asetetaan vaatimuksia, joiden täyttyminen osoitetaan suunnitteludokumentaation avulla.

Lääkintälaitteiden ohjelmistosuunnittelulle voidaan soveltaa esimerkiksi ISO/IEC12207-elinkaarimallia, johon lisätään yrityksen omat tarpeet ja lakisääteiset vaatimukset (MDD tai IVDD ja ISO 13485, ISO14971 sekä IEC 60601-1-4:n vaatimukset).

Lääkintälaitteiden toiminta perustuu siihen, että käytetään hyväksi useita eri teknologioita, joita sitten yhdistetään toiminnallisesti eri rajapintojen kautta toisiinsa. Yhteistä eri tekniikoille on kuitenkin (anturit) se, että lopullinen potilasliitynnän antureiden, käyttöliittymien, tietokantojen ja lähiverkkojen yhdistäminen toiminnalliseksi kokonaisuudeksi tapahtuu ohjelmistojen avulla.

Perinteinen ohjelmistotuotannon vesiputousmalli ei välttämättä sovellu yritykselle sellaisenaan, minkä takia tarvitaan yrityskohtaista räätälöityä mallia, jossa ideaalimallia viritetään yrityksen tarpeisiin sopivaksi. Räätälöinti kohdistuu pääasiassa lakisääteisten vaatimusten lisäämisestä suunnittelun lähtötiedoiksi, riskienhallinnan toimenpiteistä sekä jäljitettävän suunnitteludokumentaation kehittämistarpeista. Lisäksi yrityskohtaiset tarpeet voivat aiheuttaa muutoksia ideaalimalliin.



Kuva 2. Lääkintälaitteen ohjelmistosuunnittelun elinkaari.

Lääkintälaitteen ohjelmistotuotannon elinkaari sisältää neljä merkittävää eroa perinteiseen vesiputousmalliin verrattuna (kuva 2):

1. Valmistajan tulee kyetä osoittamaan ennen markkinoille saattamista, että lääkintälaitteen ohjelmisto täyttää sille asetetut lakisääteiset vaatimukset (MDD). Vaatimukset liittyvät turvallisuuteen, suorituskykyyn ja vaikuttavuuteen, ja ne tulee lisätä suoraan suunnittelun lähtötietovaatimuksiksi.
2. Ohjelmistojen suunnittelu on toteutettava siten, että koko suunnittelun elinkaari katetaan tarkoituksenmukaisilla riskienhallinnan toimenpiteillä. Riskianalyysi

alkaa suunnittelun alkaessa ja loppuu, kun suunniteltu ohjelmisto vapautetaan tuotantoon. Todellisuudessa riskienhallinnan on katettava myös tuotanto- sekä käyttövaihe (ISO 14971). Riskienhallinnan integroiminen osaksi ohjelmistosuunnittelun elinkaarta edellyttää hyvin ohjeistettua suunnitteluprosessia, jossa kaikki vaiheet on kuvattu hyvin ja jokaiselle vaiheelle on määritelty tarvittavassa laajuudessa omat menetelmäohjeet.

3. Riskianalyysin on osoitettava vaatimusmäärittelystä ne vaatimukset, joilla ehkäistään haitallisten tapahtumien toteutumista. Käytännössä tämä edellyttää muutamaa kierrosta vaatimusmäärittelyn ja riskianalyysin välillä. Edellä mainitut vaatimukset ovat turvallisuuden kannalta kriittisiä, joten verifiointi- ja validointisuunnitelmassa tulee huomioida näiden vaatimusten verifiointi ja validointi.

   Riskianalyysin tulosten perusteella voidaan ei-kriittisten vaatimusten dokumentointia keventää esimerkiksi verifioinnin ja validoinnin osalta. Menetelmä tuo kustannussäästöä niin resurssien kuin dokumentointiin käytettävän ajan säästymisenä.

4. Ohjelmiston vaatimustenmukaisuus arvioidaan ohjelmiston suunnitteludokumenteista. Arviointi kohdistuu suunnitteludokumentteihin ja ohjelmistoa tuottavaan suunnitteluprosessiin. Suunnitteludokumentaatio on avainasemassa tuotteen hyväksyntäprosessissa, joten elinkaarimallin eri vaiheille on määriteltävä erittäin tarkkaan myös tulosten dokumentointi. Suunnitteludokumentaatio on käytännössä MDD:n määrittelemä DMR (ks. liite A), joka sisältää kaiken tuotteen suunnitteludokumentaation, valmistusdokumentaation, testiraportit, asennus-, huolto- ja käyttöohjeet sekä riskianalyysiraportit. Ohjelmistojen osalta DMR:ään lisätään ohjelmistotuotannon vaihedokumentit, kuten esitutkimusraportti, järjestelmävaatimukset, arkkitehtuurispesifikaatio, ohjelmiston vaatimusspesifikaatio, testisuunnitelmat ja testiraportit, hyväksyntätestiraportit, tuotannollistamissuunnitelmat, merkinnät sekä pakkaukset.

   Lääkintälaitteille asetettavista vaatimuksista (IEC 60601-1-4) johtuen edellytetään lisäksi seuraavat dokumentit:

- riskienhallintasuunnitelma (voi olla osa projektisuunnitelmaa, huomioitava tällöin riskienhallintasuunnitelmalle asetettavat vaatimukset) sekä riskianalyysi
- verifiointisuunnitelma ja verifiointiraportti
- validointisuunnitelma ja validointiraportti.

Lääkintälaitteiden ohjelmistoille on kehitteillä uusi standardiluonnos IEC 62304, joka määrittelee elinkaariprosessin toimintoineen ja tehtävineen. Luonnos tuottaa puitteet

elinkaarimallille toimintoineen ja tehtävineen. Elinkaarimallin avulla lääkintälaitteen ohjelmistoja voidaan suunnitella ja ylläpitää turvallisesti. Standardi määrittelee myös vaatimukset kullekin elinkaaren vaiheelle.

Standardi noudattelee varsinaisen elinkaarimallin ja tukiprosessien periaatetta ja perustuu olettamukselle, että suunnittelu ja ylläpito toteutetaan ISO 13485 -laatujärjestelmän mukaisesti.

Standardi ei edellytä tiettyä elinkaarimallia, vaikka informatiivisena esimerkkinä on käytetty V-mallia.

Standardi soveltuu lääkintälaitteiden ohjelmistojen suunnitteluun ja ylläpitoon, ja sitä voidaan soveltaa, kun ohjelmisto itse muodostaa lääkintälaitteen, tai se on sulautettu ohjelmisto, tai ohjelmisto on olennainen osa lopullista lääkintälaitetta.

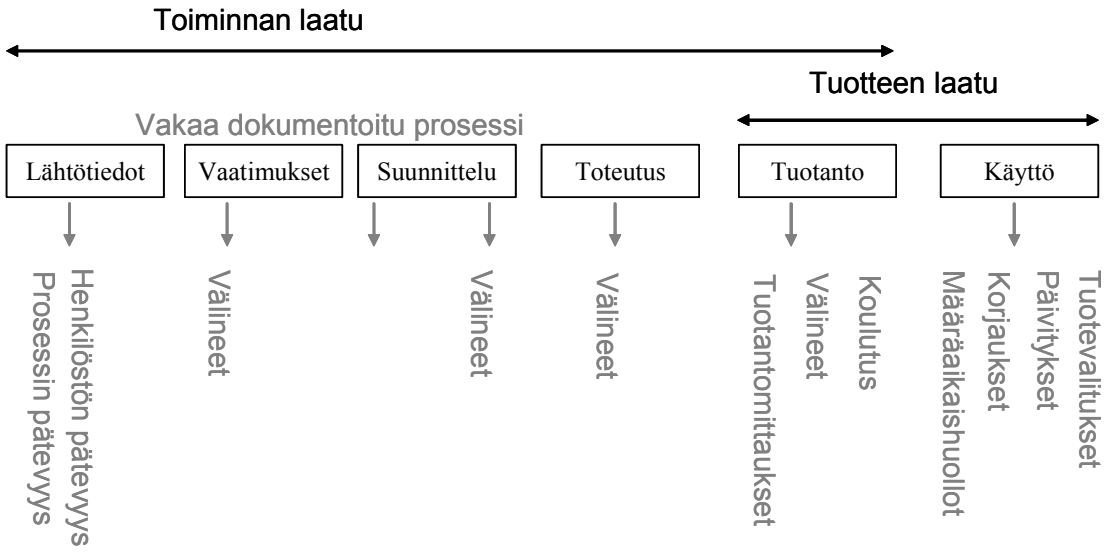
4. Kehitysnäkökohtia

Tässä luvussa käsitellään ohjelmistotuotannon vaihejakomallin eri osa-alueiden kehityskohteita, joilla suunnittelun tai ohjelmiston laatua voidaan kehittää. Kehityskohteita käsitellään lääkintälaitteiden ohjelmistoille asetettujen vaatimusten näkökulmasta, ja esimerkkinä ovat perinteisen vesiputousmallin vaiheet.

4.1 Laatumittarit

Suunnitteluprosessin kehittäminen edellyttää vanhan prosessin perinpohjaista tuntemusta ja halua etsiä muutostarpeita. Lisäksi tarvitaan mittarit, joilla laatua voidaan seurata ja arvioida. Laadun mittaaminen edellyttää, että prosessilla on tietty vakausaste. Alati muuttuvaa prosessia tai kertaluonteista toimintaa on vaikeaa mitata tai arvioida luotettavasti.

Useimpiin laatuongelmiin ei löydy yksiselitteistä pikaratkaisua. Ensisijaisesti yrityksen on kehitettävä omaan toimintaansa soveltuvat laatumittarit, joita seurataan säännöllisesti. Tulosten talletus on ensisijaisen tärkeää, koska vasta hieman pidemmällä aikavälillä ja isommasta datajoukosta pystytään arvioimaan, ovatko käytössä olevat laatumittarit oikeita tai tarkoituksenmukaisia. Tulosten perusteella voidaan sitten tehdä arviot korjaavista toimenpiteistä.



Kuva 3. Tuotteen elinkaareen sisältyy erilaisia mitattavia kohteita.

Laatumittareita voidaan periaatteessa kohdistaa kaikkiin tuotteen elinkaareen osiin varhaisesta suunnittelusta tuotteen käytöstä poistoon asti (kuva 3). Kohteina olisivat tällöin prosessien kypsyys, toistettavuus, dokumentointi, suunnitteluhenkilöstön pätevyys, me-

netelmät, työkalut, tuotantomittausten kattavuus, koulutus, muutostenhallinta, tuotemuutokset ja päivitykset.

Tuotteen laadun mittaamiseen käytettävät laatumittarit eivät välttämättä ole samoja kuin suunnitteluprosessin laadun mittaamiseen käytettävät mittarit. Lopputuloksena kummankin mittarin osalta korjaava toimenpide voi kohdistua itse suunnitteluprosessiin, valmistukseen tai koulutukseen.

Laatumittareiden on oltava riittävän yleisellä tasolla, jotta prosessin mittaus voidaan aloittaa. Suunnitteluprosessin ja tuotteen laadun mittareina voidaan käyttää esimerkiksi seuraavia tekijöitä:

- aikaa, jossa tuote saavuttaa positiivisen kassavirran
- asiakastyytyväisyyskyselyitä (mihin ollaan erittäin tyytyväisiä, mihin ollaan vähemmän tyytyväisiä, suorituskyky, käytettävyys, luotettavuus)
- asiakasvalituksia (kuinka paljon, missä ajassa, mihin liittyy, mihin vuodenaikaan, kuinka merkittäviä, onko turvallisuusriskejä)
- käyttäjäarviota (tietylle kohderyhmälle tehty haastattelu tai kysely, joka kohdistuu tiettyihin toimintoihin tai tuotteen kohtiin)
- sisäisten auditointien poikkeamaseurantaa
- projektien läpimenoaikoja ja budjettiseurantaa
- koodin läpikäyntejä (virheraportointi)
- in-house-testeissä havaittuja poikkeamia
- regressiotestausta (uusi muutos voi tuoda ongelmia vanhaan koodiin, kuinka usein näin käy)
- vaatimusmäärittelyn muutoksia (kuinka usein muutetaan määrittelyjä, missä ajassa tuotteeseen lisätään vaatimuksia).

Laatumittareina voidaan käyttää myös yksityiskohtaisia testauksen pass- ja fail-kriteerejä tai ohjelmistosta löytyvien virheiden määrää.[1] Nämä mittarit soveltuvat hyvin silloin, kun kehitettävä ohjelmisto on koko ajan sama eikä siinä tehdä radikaaleja muutoksia. Tällaiset mittarit eivät välttämättä sovellu eri ohjelmistoperheiden väliseen laadun mittaukseen.

[1] Ketterien ohjelmistotuotantomenetelmien avainkäytäntöjä ovat pienet julkistukset, jolloin uusi julkistus voidaan tehdä vaikka yhden uuden vaatimuksen lisäyksen jälkeen. Tämän perusteella virhemäärien laskenta ei välttämättä ole sovelias laatumittari XP:ssä.

Laatua voidaan sitten pyrkiä kehittämään seuraavilla toimenpiteillä:

- Katselmuksissa keskitytään ongelmakohtiin (katselmukset voivat olla joko suunnittelukatselmuksia tai koodikatselmuksia, suunnittelukatselmuksien tueksi voidaan laatia valmiita tarkastuslistoja, koodikatselmuksia tuetaan yrityskohtaisilla tyylioppailla).
- Laaditaan kattavat koodausohjeet (määritellään ohjelmointikieli, merkistöt, aikaformaatit, tietokannat, virheentarkastusrutiinit, virheistä toipuminen jne.).
- Käytetään koodin tarkastuslistoja ja parityöskentelyä.
- Kehitetään vaatimusmäärittelyä (riskianalyysi on aidosti mukana määrittelyvaiheessa; vaatimukset voidaan jakaa kriittisiin tai ei-kriittisiin vaatimuksiin).
- Opiskellaan riskianalyysitekniikoita, integroidaan riskianalyysi kiinteämmäksi osaksi itse suunnitteluprosessia.
- Kerätään aktiivisesti asiakaspalautetta.
- Laaditaan suunnittelulle muutostenhallintaprosessi (muutokset analysoidaan, selkeä päätöksentekoprosessi muutoksille, muutosten dokumentointi, jäljitettävyys vanhoihin suunnitteludokumentteihin, riskianalyysien ulottaminen muutoksiin tarvittavassa laajuudessa).

Laatumittarit keskitetään helposti prosessien epäkohtien mittaamiseen, mutta pienellä viiveellä prosessien kehitys parantaa myös tuotteen laatua ja saattaa näin ollen olla parempikin kohde kuin pelkät tuotteeseen liittyvät laatumittarit.

4.1.1 Menetelmäohjeet ja suunnittelun aloitus

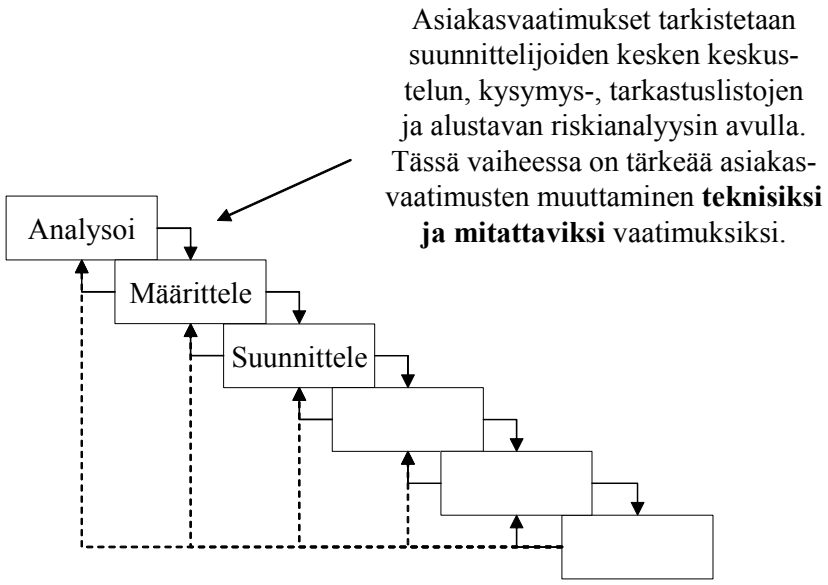
Laatujärjestelmien eräs tarkoitus on, että yrityksen toiminnot puretaan erillisiksi prosesseiksi. Prosessien toimintaa ohjataan sitten erilaisilla valvotuilla menetelmäohjeilla ja opasdokumenteilla. Ulkoiset arvioijat arvioivat sitten prosessien toimintaa näitä menetelmäohjeita ja laatujärjestelmästandardien asettamia vaatimuksia vastaan.

Eräs käytännössä havaittu ongelma on syntynytkin siinä, että prosessien edellyttämät menetelmäohjeet on laadittu sellaisten henkilöiden toimesta, jotka eivät käytännössä tunne ko. prosessia, tai menetelmäohjeet on laadittu ulkoisia arvioijia varten eivätkä ne täten vastaa kyseisen prosessin normaalia toimintaa.

Esimerkiksi tuotteen tai ohjelmiston suunnittelu on prosessi (vaihejakomalli), joka edellyttää useampaa menetelmäohjetta toimiakseen. Prosessin kannalta optimaalisin ohjeistus olisi kuvata ja ohjeistaa suunnitteluprosessi juuri sellaisena, kuin suunnittelijat sitä

päivittäin tekevät. Mikäli eri suunnittelijoiden kesken on erilainen näkemys suunnitteluprosessin sisällöstä ja tavoitteista tai prosessi ei tuota riittävän laadukasta suunnitteludokumentaatiota, on se merkki siitä, että käytännön toiminta ja menetelmäohjeet eivät ole yhteneväiset. Tällöin yrityksen on järjestettävä lisäkoulutusta tai tehtävä muutoksia suunnitteluprosessiin, minkä jälkeen päivitetään laadittuja menetelmäohjeistuksia. Ongelmat suunnitteluprosessissa voidaan paljastaa säännöllisin aikavälein tehtävillä sisäisillä auditoinneilla, joissa saatetaan havaita seuraavankaltaisia ongelmia:

- Suunnitteluprosessi on tehty liian jäykäksi, suunnittelu ei todellisuudessa toimi niin, ja suunnittelutiimi rimpuilee ohjattua ja systemaattista suunnitteluprosessia vastaan.
- Dokumentointi ei ole osa normaalia toimintaa, eikä prosessin eri vaiheissa ole määritelty vaatimuksia prosessin lähtö- ja tulostiedoiksi.
- Riskianalyysi ei ole osa suunnittelun elinkaarta, riskianalyysi tehdään, kun se on tehtävä tai tehdään sitten joskus kun ei ole muuta; tästä seuraa vaikeuksia auditoinneissa tai jäljitettävyyden ja vaatimustenmukaisuuden osoittamisessa.



Kuva 4. Suunnittelun alkuvaihe ratkaisee pitkälti suunnittelun onnistumisen.

Kuvasta 4 voidaan päätellä, että suunnittelun alkuvaihe on merkittävän tärkeä ohjelmiston suunnittelun kannalta. Tärkeää on pystyä analysoimaan asiakasvaatimukset riittävän ammattitaitoisella joukolla ja muuttaa ne mitattaviksi teknisiksi järjestelmävaatimuksiksi, jolloin vaatimusmäärittely helpottuu ja suunnittelutiimin on helpompi tunnistaa ohjelmiston turvallisen toiminnan kannalta kriittiset vaatimukset. Panostus suunnittelun alkuvaiheeseen voi tuoda seuraavia etuja:

- lyhyempi aika esitutkimuksesta tuotteeksi
- resurssien optimaalinen käyttö – halvempi (alusta alkaen suunnittelijoilla on selvä työjako, suunnittelua voidaan kohdistaa kriittisiin kohtiin)
- tuotteen valmistuminen paremmin suunniteltavissa
- olennaisetkin muutokset vielä mahdollisia potentiaalisten ongelmakohtien paljastuessa – suunnittelun optimointi
- hyväksyttävä lopputulos varmempi (validointi osoitettavissa yrityksen omalle laadunvarmistustiimille ja kolmansille osapuolille, tuote saadaan sovitussa ajassa markkinoille, viranomaishyväksynnät onnistuvat helpommin).

Suunnittelun alkuvaiheessa laaditaan myös ohjelmistolta edellytettävät riskienhallinta-verifiointi- ja validointisuunnitelmat, joiden mukaan tuotekohtainen riskienhallinta, verifiointi ja validointi toteutetaan. Tästä syystä suunnittelun alkuvaiheen kehittämisellä voidaan parantaa merkittävästi suunnittelun laatua ja kustannustehokkuutta.

Suunnitteluprosessi on erittäin monitahoinen prosessi, joka edellyttää tuekseen useampia tukiprosesseja, ja sillä tulee olla kiinteä ohjeistettu vuorovaikutus yrityksen muihin prosesseihin. Toiminnan tehokkuuden ja toistettavuuden kannalta on tärkeää kuvata suunnitteluprosessin eri vaiheet ja kultakin vaiheelta edellytettävät lähtötiedot sekä tulostiedot. Kullekin vaiheelle laaditaan määrämuotoiset tarkastuslistat, joiden mukaan suunnittelukatselmuksissa voidaan tarkastaa vaihetuotokset.

4.1.2 Ohjelmiston erityispiirteet vaikuttavat suunnitteluprosessiin

Elektronisiin laitteisiin asennetaan erilaisia ohjelmistoja, kuten reaaliaikakäyttöjärjestelmiä sovelluksineen, ohjelmoitavia mikrokontrollereita tai jopa Windows-käyttöjärjestelmä sovellusohjelmineen. Windows-ohjelmien tyypillisiä tunnuspiirteitä ovat esimerkiksi ohjelmiston laajuus (koko, levinneisyys) ja erilaisten rajapintojen ja prosessien runsaus, kun taas sulautetun ohjelmiston tunnuspiirteitä ovat lähinnä laitelähеisyys (anturi tai toimielin sisältää ohjelmistoa) tai nopeat vasteaikavaatimukset.

Eri ohjelmistojen tunnuspiirteistä (sulautetut, laajat, monimutkaiset tai reaaliaikaohjelmistot jne.) johtuen niiden kehityksessä vallitsevat erilaiset lainalaisuudet. Esimerkiksi laajojen tai monimutkaisten ohjelmistojen kehitys ja laadunhallinta voi olla hyvinkin erilaista.

Laajojen ohjelmistojen kehitystyössä voivat aikataulut ja resurssipula olla erittäin merkittävä tekijä. Laajan ohjelmiston tunnuspiirteitä voivat olla esimerkiksi suorittavan tiedoston koko, lähdekoodin rivimäärä, hajautus (ohjelmisto suorittaa tehtäviä eri palve-

limilla ja kommunikoi keskenään Internetin tai lähiverkon kautta) tai se, että vaatimuksia on erittäin paljon. Tällöin ohjelmistokehityksen vaikuttavia seikkoja ovat

- projektin ja resurssien hallinta (aikataulut)
- eri osapuolten kommunikointi (laaja ohjelmisto, paljon kehittäjiä, paljon muita projektihenkilöitä)
- mahdollisuus valmiskomponenttien käyttöön.

Monimutkaisten ohjelmistojen suunnittelu voi taas kaatua siihen, että ohjelmistoa ei saada toimimaan tai se ei täytä sille asetettuja suorituskykyvaatimuksia. Monimutkaisessa ohjelmistossa vaatimusten tai suorituskyvyn määrittäminen voi olla oleellisesti vaikeampaa kuin laajoissa ohjelmistoissa. Monimutkaisen ohjelmiston tunnuspiirteitä voivat olla esimerkiksi monimutkaiset algoritmit, datan käsittely, laskentaan perustuva diagnosointi tai päätöksenteko ja loogiset päättelypuut, tai vaatimuksia voi olla vähemmän, mutta ne ovat kriittisempiä (kaikki vaikuttaa kaikkeen).

Monimutkaisen ohjelmiston kehitykseen vaikuttavat seuraavat seikat:

- asiantuntijoiden käyttö korostuu
- koodaus monimutkaisempaa
- vaatimusmäärittely vaikeampaa
- validointi vaikeampaa ja kriittisempää
- suorituskyvyn osoittaminen (pelkkä vasteajan mittaus ei riitä, pitää osoittaa diagnosoinnin tehokkuus, tarkkuus jne.)
- sovellusaluevaatimukset monimutkaistuvat ja lisäävät monimutkaisuutta.

Mikäli ohjelmistoja suunnitellaan XP-menetelmää soveltaen, voidaan osia menetelmän eduista menettää ohjelmiston koon kasvaessa (http://www.xprogramming.com/what_is_xp.htm).

Ketteryyttä voidaan yrittää ylläpitää jakamalla laajojen järjestelmien suunnittelu pienempiin osiin. Ratkaisu edellyttää arkkitehtuurimäärittelyä, jonka mukaan järjestelmä voidaan paloitella eri kehitystiimien suunniteltavaksi. Arkkitehtuurimäärittely auttaa myös eri kehitystiimejä rajapintatarkasteluiden tekemisessä, jolloin myöhemmin tehtävä yhdistyminen onnistuu varmemmin. Hyvällä arkkitehtuurilla todennäköisesti vältetään ohjelmiston ”spagettikoodimainen” rakenne, jota kukaan ei kokonaisuutena kykene testaamaan.

Monimutkaisuus valitsee järjestelmän eri osien välillä, toisaalta se voi olla myös käyttäjän ja järjestelmän välillä. Järjestelmä tuottaa paljon erilaista informaatiota, jonka perusteella käyttäjä tekee mielessään oikean päätöksen ja ohjaa järjestelmän tiettyyn tilaan.

Ulospäin näkyvää monimutkaisuutta voidaan hallita osittain hyvällä käytettävyydellä. Hyvä käytettävyys voi ilmetä esimerkiksi käyttöliittymän räätälöintinä, joka huomioi erilaisia kulttuureja tai hoitokäytäntöjä tai peräti eri markkina-alueista johtuvia lakeja. Käytettävyyttä voidaan lisätä myös siten, että järjestelmä lajittelee tuotettavaa informaatiota ja välittää sen käyttäjälle valmiiksi lajitellussa ja ymmärrettävässä muodossa.

Yksittäisiin terveydenhuollon laitteisiin suunnitellut ohjelmistot ovat tyypiltään enemmän monimutkaisia ohjelmistoja. Tämä tulee huomioida suunnitteluprosessin kehittämisessä. Mikäli käytettävyydellä halutaan hallita monimutkaisuutta, tulee suunnittelussa huomioida käytettävyydelle asetettujen standardien vaatimuksia (IEC 60601-1-6, ISO 9241).

4.2 Vaihekohtaisia kehityskohteita

Tässä kohdassa käsitellään ohjelmistokehityksen vaihekohtaisia kehityskohteita lääkintälaitteiden ohjelmistosuunnittelulle asetettujen vaatimusten näkökulmasta.

Tavoitteena on kehittää ohjelmistosuunnittelua siten, että suunnittelu täyttäisi paremmin viranomaisvaatimukset ja että ohjelmistosuunnittelun kustannustehokkuus ja laatu paranisivat.

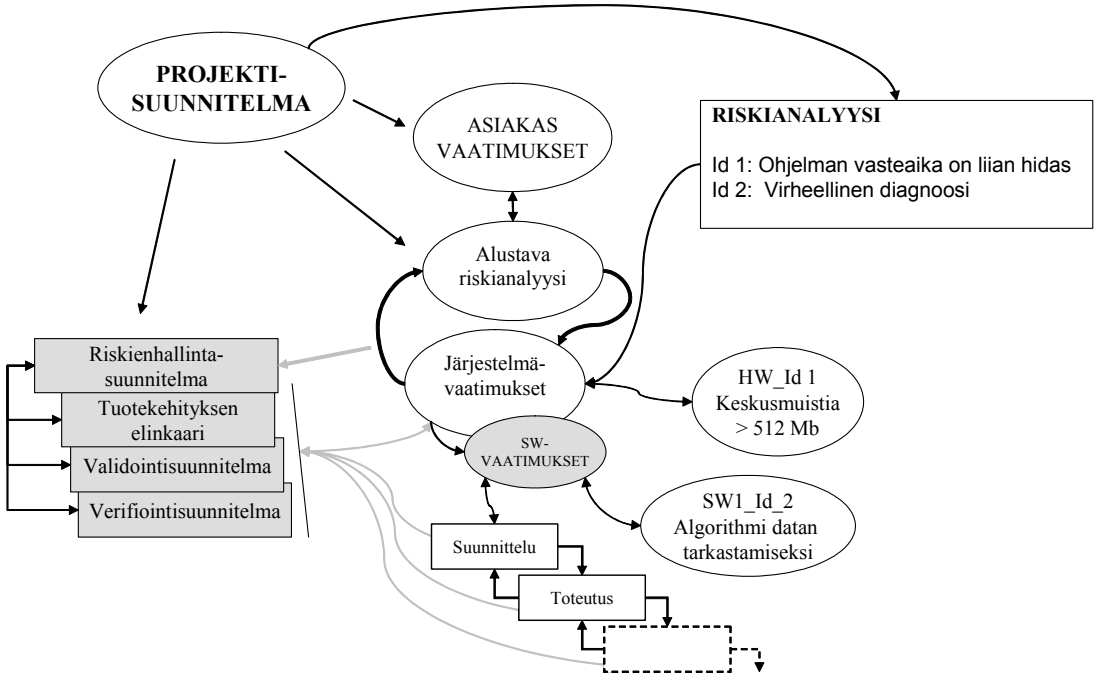
4.2.1 Esitutkimus

Esitutkimusvaiheessa hanketta pohditaan ja tarkastellaan lähinnä taloudelliselta kannalta, jolloin puntaroidaan hankkeen järkevyyttä ja taloudellisia mahdollisuuksia. Usein hanke saa alkunsa asiakkaiden tai yrityksen markkinointitiimin esittämistä toiveista.

Mikäli idea saa yritysjohdon hyväksynnän ja siitä käynnistyy uusi hanke, käynnistetään samalla uuden tuotteen suunnitteluun tähtäävät toimet. Projektisuunnitelmassa määritellään aikataulu, kustannukset, resurssit ja muut yrityskohtaiset vaatimukset hankkeen tuotteistamiseksi.

Samalla on käynnistettävä, joko osana projektisuunnitelmaa tai erillisinä tehtävinä, tuotekohtaisten riskienhallinta-, verifiointi- ja validointisuunnitelman laadinta. Suunnitelmissa tulee huomioida MDD:n ja IEC 60601-1-4:n vaatimukset. Suunnitelmat edellyttävät myös alustavaa riskianalyysiä, jotta voidaan määritellä riskienhallinta-

toimenpiteiden laajuus ja kattavuus suunnittelussa. Alustava riskianalyysi voi olla esimerkiksi aivoriihityyppinen analyysi, jolla kartoitetaan suunniteltavaan tuotteeseen liittyviä riskejä lähinnä sovelluksen ja sen aiottujen toimintojen kautta. Tuloksia käytetään riskienhallinta-, verifiointi- ja validointisuunnitelmien laatimiseksi. Suunnitelmia voidaan tarvittaessa tarkentaa myöhemmin, kun suunnittelu on saanut hieman tarkempia spesifikaatioita (kuva 5).



Kuva 5. Suunnittelun alkuvaiheessa määritellään suunnitelmat.

Projektisuunnitelmissa tulee huomioida lääkintälaitteiden ohjelmistoille asetetut suoritus- ja turvallisuusvaatimukset, jotka edellyttävät projektin resursseilta riittävää kliinistä asiantuntemusta sekä riskienhallinnan ammattilaisia ja sovellusalueen osaajia. Ilman näitä resursseja on syytä epäillä, että tehdyt riskianalyysit eivät ole riittävän kattavia tai järjestelmä- ja ohjelmistovaatimusten laadinta ei täytä ohjelmistolle asetettuja turvallisuus- ja suorituskykyvaatimuksia.

Koska ohjelmiston vaatimustenmukaisuuden arviointi perustuu suunnitteludokumentaation arviointiin, tulee yrityksen suunnitteluprosessin sisältää vaatimukset vaihekohtaiselle dokumentaatiolle ja tukea näiden dokumenttien tuottamista valmiilla dokumenttipohjilla sekä riittävän yksityiskohtaisilla toimintaohjeilla.

Toimialakohtaisista vaatimuksista johtuen seuraaville dokumenteille voisi laatia valmiit pohjat:

- riskienhallintasuunnitelma, riskianalyysi ja riskienhallinnan yhteenvetoraportti (voi olla myös osa projektisuunnitelma, huomioitava silloin standardien ja direktiivien asettamat vaatimukset)
- verifiointisuunnitelma ja raportti
- validointisuunitelma ja raportti
- järjestelmävaatimukset sekä ohjelmiston vaatimusmäärittely, jotka mahdollistavat jäljitettävyyden itse vaatimusmäärittelyn, riskianalyysin, toteutuksen, testauksen ja validoinnin välillä (erityisesti kriittiset vaatimukset).

Tuotteen markkinoille saattamisen yhteydessä tehtävissä hyväksyntäarvioinneissa tarkastetaan näitä seikkoja.

4.2.2 Määrittely

Määrittelyvaihe johtaa asiakasvaatimuksista järjestelmävaatimukset, jotka tarkemman analysoinnin jälkeen johdetaan joko laitteisto- tai ohjelmistovaatimuksiksi.

Tässä vaiheessa alustava riskianalyysi tarkentuu toimintojen ja ominaisuuksien tarkemmaksi riskianalyysiksi. Laaditut suunnitelmat päivittyvät ja tarkentuvat.

Vaiheelle merkittävin toimialakohtainen vaatimus on tunnistaa järjestelmästä ne ominaisuudet ja toiminnot, jotka voivat aiheuttaa riskin tai vaaran toteutumisen järjestelmän aiotussa käyttötarkoituksessa. Riskianalyysin tuloksista johdetaan vaatimukset, joilla eliminoidaan analysoitua riskiä tai vaaraa toteutumasta.

Analysoinnin tulee kattaa myös laitteen virheellinen käyttö. Virheellinen käyttö voi johtua inhimillisistä virheistä, puutteellisesta koulutuksesta tai virheellisistä suunnitteluratkaisuista (esimerkiksi therac-25-tapaus).

Käytännössä määrittelyvaiheessa tapahtuu useampia kierroksia järjestelmä-, ohjelmisto- ja laitteistovaatimusten välillä, ja luultavaa on, että suunnitteluvaiheestakin palataan vielä tarkentamaan vaatimuksia. Tämä vaihe on suunnittelun onnistumisen kannalta erittäin tärkeää, koska suunnittelukustannukset kasvavat oleellisesti, jos tästä edetään virheellisten vaatimusten kanssa yksityiskohtaiseen suunnitteluun ja toteutukseen.

Määrittely tulee toteuttaa siten, että määrittelyn, riskienhallinnan, toteutuksen ja testauksen välisten toimenpiteiden välillä saadaan rakennettua kattava jäljitettävyys. Jäljitettävyyden tulee toimia suunnitteluprosessissa eteen- tai taaksepäin.

Seuraavaksi käsitellään yksityiskohtaisesti muutamia vaatimusmäärittelyyn ja hallintaan liittyviä seikkoja.

4.2.2.1 Vaatimusmäärittely

Vaatimusmäärittely on eräs tärkeimmistä ohjelmistojen suunnittelun osa-alueista (Pöyhönen & Hukki 2004), johon kulminoituvat myös useimmat ohjelmistojen toimintahäiriöt ja luotettavuusongelmat. Tästä syystä vaatimusmäärittelyprosessin kehittämiseen on syytä kiinnittää huomiota.

Vaatimusmäärittelyn tekee vaikeaksi ehkä eniten ohjelmistojen luonne, koska ohjelmistoilta puuttuu selkeät mittarit, milloin ohjelmisto on hyvä, riittävän suorituskykyinen, ja milloin se on huono. Ohjelmistojen käyttö hankaloittaa myös vaatimusmäärittelyä, koska ohjelmiston tietyt toiminnot riippuvat useista eri tekijöistä, alkutilanteista ja oletusarvioista.

Vaatimusmäärittelyn onnistumiseen vaikuttaa myös oleellisesti se, että toiminnan kannalta oikeat ihmiset ovat määrittelemässä ohjelmistovaatimuksia. Taulukossa 1 on muutama esimerkki siitä, minkälaisia ammattilaisia, tietoa ja asiantuntemusta on oltava mukana määrittelyvaiheessa.

Taulukko 1. Vaatimusmäärittely edellyttää usean eri alan asiantuntijoita.

Asiakasvaatimus	Järjestelmävaatimus	SRS	Laitteistovaatimus	Tieto + asiantuntijat
Ohjelmisto soveltuu tehomonitorointiin sekä tutkimukseen	Ohjelmisto sisältää tulosten analysointiominaisuuden	Analyysialgoritmi (algoritmi tuottaa useita erillisiä vaatimuksia)	Vasteaikavaatimus < 0,01 us L2-välimuistia 256 Mb	Kliininen tutkimus Järjestelmäasiantuntija Algoritmiasiantuntija Kliinikko Sovellusalueosaaja Ohjelmistosuunnittelija Riskianalyysiasiantuntija
Nopeuttaa tutkimusta, on helppokäyttöinen	Edellyttää tutkimuksen esiasetuksia	Default-arvot ohjelman käynnistyttyä Ylä- ja alarajahälytykset	Keskusmuistia 512 Mb Levytilaa 5 Gt	Kliinikko Käyttäjä Sovellusalueosaaja Ohjelmistosuunnittelija

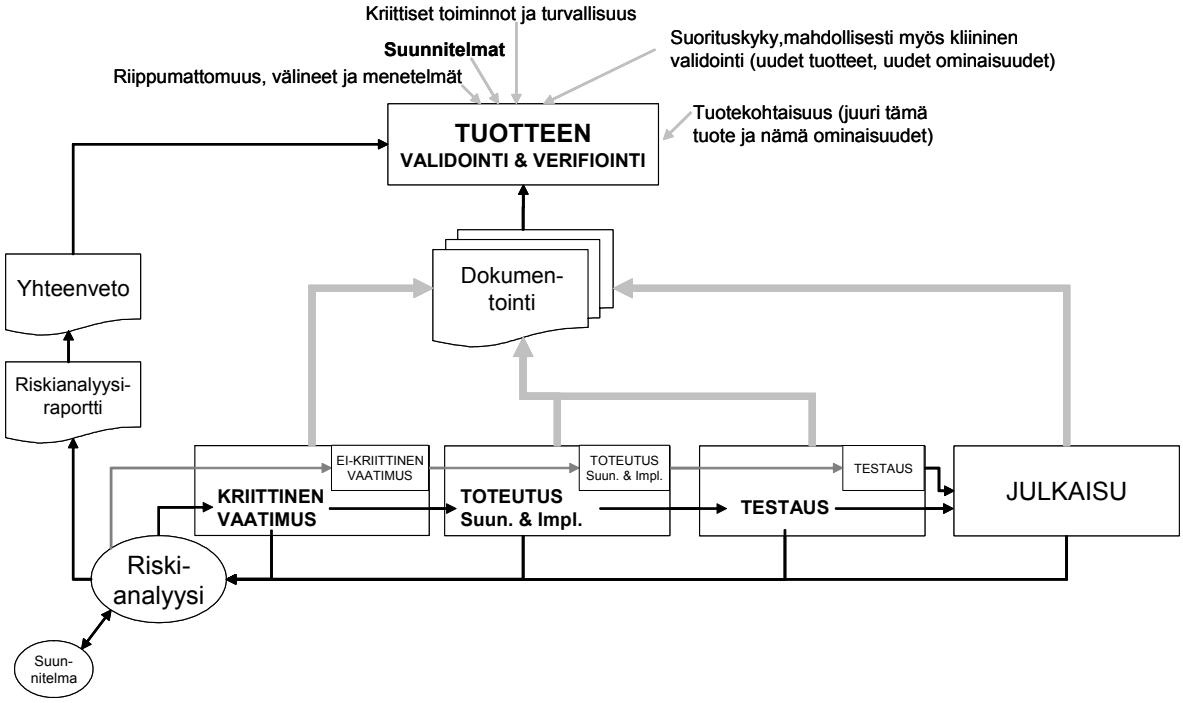
Tutkimus tuottaa lisäarvoa EU:ssa, USA:ssa ja Aasiassa	1. Tutkimusalgoritmi edellyttää väestöpohjan tunnistamista	Potilaan kansallisuuden tunnistus (GUI) Algoritmi väestöpohjaisten erojen tunnistukseen	Levytilaa 5 Gb	Kliininen tutkimus sisältäen vaikuttavuuden, suorituskyvyn sekä tarkkuuden Kliinikko Järjestelmäasiantuntija sekä ohjelmistosuunnittelija lokalisointiin Riskianalyysiammattilainen
	2. Ohjelmiston tulee täyttää lakisääteiset vaatimukset USA:ssa, EU:ssa sekä Aasiassa	Ohjelmisto suunniteltava kansallisten standardien mukaisesti Täytettävä tarvittavat suoritus-kyky- vaatimukset	Laitteiston täytettävä tarvittavat sähköturvallisuus- sekä suoritus-kykyvaatimukset	Viranomaisvaatimusten tuntija Eri markkina-alueiden asiantuntija

Vaatimusmäärittelyyn liittyy paljon ongelmia, ja usein ongelmat liittyvät määrittelyvaiheen tiedonkulkuun tai riittämättömään määrään eri alan asiantuntijoita määrittelyvaiheessa. Lopputuloksena kaikesta ovat kasvaneet suunnittelukustannukset sekä turvallisuusriskit ohjelmiston käytössä.

Periaatteessa ohjelmistovaatimusten tarkastus on helppoa. Kirjoitetaan vaatimus ja testataan se kaikilla vaihtoehdoilla, kaikissa eri tilanteissa, kaikilla mahdollisilla potilailla ja kaikilla eri koulutustaustan ja kulttuurin omaavilla käyttäjillä.

Edellä mainitun kaltainen testaus on mahdotonta siihen kuluvan ajan ja kustannusten takia. Lisäksi inhimilliset tekijät (osaaminen, kiire, dokumentoinnin puutteet ja resurssit) vaikuttavat siihen, että kaikkia ominaisuuksia ei vain muisteta testata.

Vaatimustenhallintaa helpottaisi hyvin ohjeistettu ja systemaattinen suunnitteluprosessi, jossa vaatimuksia pohdittaisiin modulaarisuuden ja arkkitehtuurin kannalta (ks. kohta 4.2.3.1 Arkkitehtuuri).



Kuva 6. Riskianalyysin avulla tunnistetaan kriittiset vaatimukset.

Riskienhallinnan avulla on mahdollista osoittaa ohjelmistosta turvallisuuden kannalta kriittiset toiminnot ja niitä ohjaavat ohjelmistovaatimukset (kuva 6). Näitä toimintoja voidaan sitten analysoida, testata ja jäljittää suunnittelun eri vaiheissa hyvinkin tarkasti. Turvallisuuden ja suorituskyvyn kannalta merkityksettömimmät toiminnot ja vaatimukset voidaan jättää hieman vähemmälle huomiolle. Menettely edellyttää tuekseen myös huolellista arkkitehtuurisuunnittelua, jossa turvallisuuskriittiset toiminnot sijoitetaan yhteen tai kahteen moduuliin.

Standardissa IEEE 830 kuvataan vaatimuksia ohjelmiston vaatimusspesifikaation laatimiseksi. Standardissa on myös alustava luonnos vaatimusmäärittelyn pohjaksi.

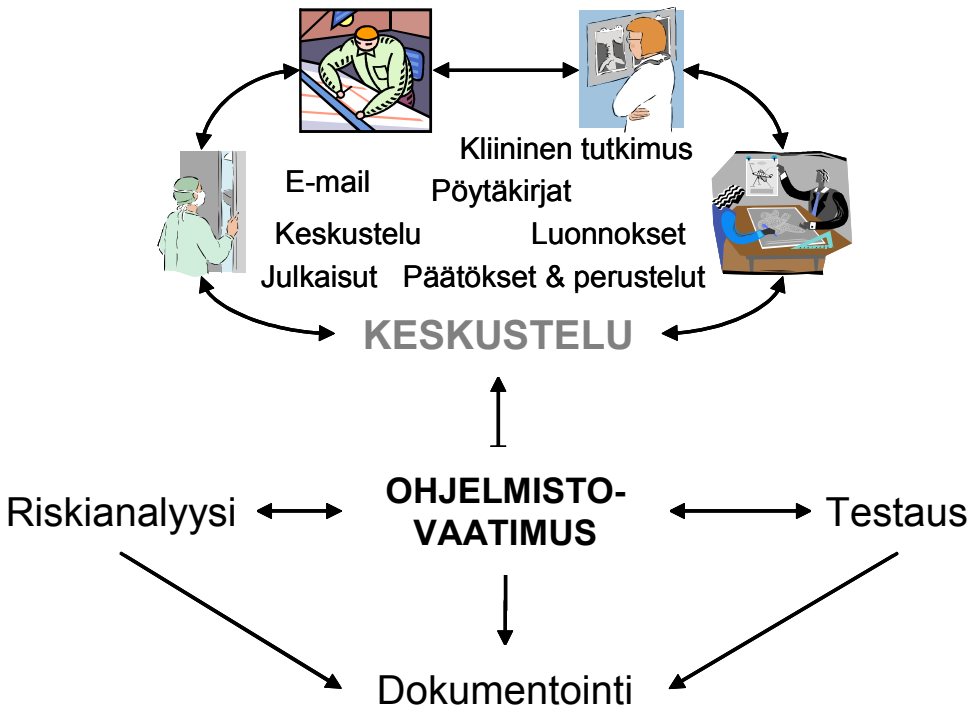
4.2.2.2 Vaatimustenhallinta

Vaatimusten muuttuminen kesken suunnittelun on enemmän tosiasia kuin poikkeus. Muuttuvilla vaatimuksilla pyritään usein parantamaan tuotetta, tuomaan tuotteeseen uusia ominaisuuksia tai lisäämään tuotteen turvallisuutta.

Tehtiin vaatimusmäärittelyn muutos mistä syystä tahansa, se vaikuttaa useaan eri kohtaan suunnitteluprosessissa. Yhden kriittisen vaatimuksen muuttamisesta saattaa pahimmillaan seurata koko vaatimusmäärittelyvaiheen uusiminen: testisuunnitelmia joudutaan uusimaan, koodia joudutaan muuttamaan, koodin läpikäynti joudutaan uusimaan ja tehtyjä riskianalyysejä joudutaan uusimaan tai päivittämään.

Järjestelmävaatimukset täydentyvät ja tarkentuvat ohjelmiston vaatimusmäärittelyvaiheessa. Vaatimusten tarkentuminen saattaa tuoda jopa uusia vaatimuksiakin. Mikäli vaatimusmäärittelyä joudutaan muuttamaan toteutus- tai testausvaiheissa, sitä suuremmat kustannusvaikutukset muutoksella on. Sama pätee myös tuotteen turvallisuuteen tai vaatimustenmukaisuuden osoittamiseen. Mitä myöhäisemmässä vaiheessa muutos tapahtuu, sitä todennäköisempää on, että kaikkia turvallisuusominaisuuksia ei muisteta tai ehditä analysoida tai vaatimustenmukaisuutta varmentavat dokumentit jäävät päivittämättä.

Vaatimusmäärittelyn muutokset tulisi ohjeistaa siten, että tietyssä vaiheessa suunnittelu jäädytetään eikä uusia muutoksia enää hyväksytä. Jäädytyksen jälkeen vaatimukset toteutetaan, päivitetään riskianalyysi, testataan, dokumentoidaan ja validoidaan.



Kuva 7. Vaatimustenhallinnan kannalta on tärkeää dokumentoida myös vaatimukseen liittyvä keskustelu ja päätöksenteko.

Vaatimusmäärittelyyn liittyy aina riskienhallinta, testaus ja dokumentointi, joka mahdollistaa ko. vaatimuksen arvioinnin ja muutoksen jälkikäteen. Tärkeää olisi myös taltioida vaatimukseen liittyvä keskustelu ja pohdinta siitä, kuinka vaatimus on syntynyt, miten se on päätetty testata ja onko ko. vaatimus turvallisuuskriittinen vaatimus (kuva 7).

Vaatimukseen liittyvä pohdiskelu auttaa myöhemmin tehtävissä muutoksissa arvioimaan, onko ko. vaatimus nykyteknologialla ja ratkaisuilla enää tarpeen vai tulisiko vaatimus korvata kokonaan toisella vaatimuksella, koska teknologiamuutokset, tekniikka-

muutokset tai muutokset aiotussa käyttötarkoituksessa ovat saattaneet poistaa tarpeen ko. vaatimukselle. Keskustelun taltioimista edesauttavat esimerkiksi räätälöitävät vaatimustenhallinnan työkalut, joiden avulla voidaan tallentaa sähköposteja, kokouspöytäkirjoja tai eri formaatissa olevia tiedostoja.

4.2.2.3 Koodin muokkaus (refactoring)

Ohjelmistojen suunnitteluun on tullut käsite ”koodin muokkaus” (XP:n avainkäytännöt), joka lyhyesti tarkoittaa sitä, että ohjelmiston ulospäin näkyvään toiminnallisuuteen ei puututa mutta toiminnallisuuden mahdollistavia teknisiä ratkaisuja voidaan muuttaa. Toiminnallisuus voi tässä yhteydessä olla joko ohjelmiston ja käyttäjän välinen rajapinta tai ohjelmiston eri alijärjestelmien välillä oleva toiminnallisuus.

Mikäli koodin muokkaus ei aiheuta uusia vaatimuksia, voidaan vaatimukset testata vanhojen testitapausten pohjalta. Testitapaukset on laadittu vaatimusten pohjalta, joten esimerkiksi white-box-testauksen pitäisi paljastaa, onko refaktorointi muuttanut myös ohjelman ulkoista toimintaa. Näissä tapauksissa on tapauskohtaisesti arvioitava, palautetaanko koodi ennalleen vai kirjoitetaanko uusi testitapaus.

Mikäli koodin muokkauksen yhteydessä ei jouduta kirjoittamaan uusia testitapauksia ja ohjelma läpäisee jo kirjoitetut testitapaukset, ei suunnitteludokumenttien päivitystä välttämättä edellytetä. Käytännön kannalta olisi kuitenkin hyvä, että koodimuutos kuvataan ainakin moduulitasolla. Vielä parempi olisi, jos kuvattaisiin myös vaatimukset, joihin koodimuutoksia on tehty. Versionhallintatyökalut ovat hyvä työväline jäljittämään myös refaktorointia, joten näiden työkalujen käyttö on vahvasti perusteltua koodin ylläpidossa.

Koodin muokkaus on sinällään käsite, joka pitää kirjoittaa auki suunnittelua tukevissa menetelmäohjeissa. Koodin muokkauksen yhteydessä pitää määritellä selvästi, mikä toiminto on koodin muokkausta ja milloin suunnittelun dokumentointi täytyy päivittää, mikä on bugikorjaus ja mikä on normaali muutospyyntö eli uusi vaatimus.

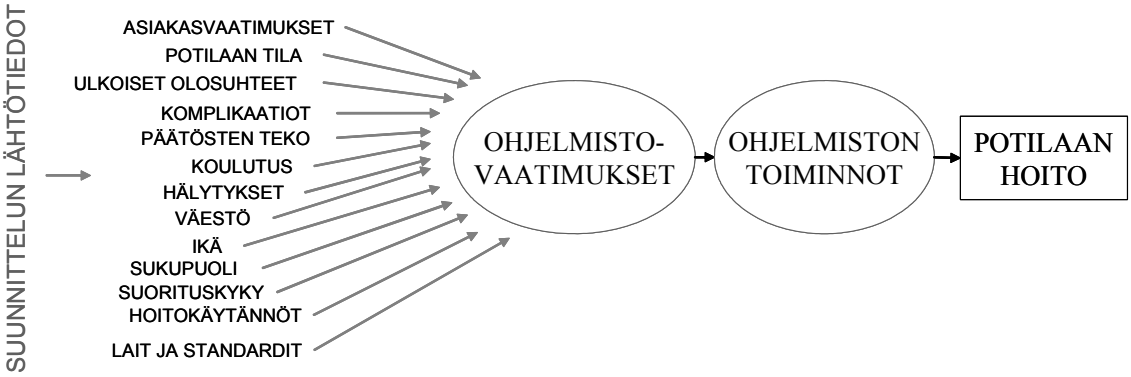
4.2.2.4 Lakisääteiset vaatimukset ovat osa määrittelyä

Terveydenhuollon tuotteen (lääkintälaite tai ohjelmisto) tulee täyttää tietyt lakisääteiset vaatimukset. Vaatimukset kohdistuvat pääasiassa turvallisuuteen ja suorituskykyyn sekä vaikuttavuuteen, ja ne voidaan lyhyesti määritellä seuraavasti:

- Tuotteen tulee olla riittävän turvallinen ja suorituskykyinen ja sen tulee soveltua aiottuun käyttötarkoitukseensa.

– Ohjelmistojen kehityksessä tulee noudattaa elinkaarimallia.
– Elinkaaren eri vaiheisiin on sovellettava kullekin vaiheelle soveltuvia riskienhallintakeinoja.
– Vaatimusmäärittelyyn tulee lisätä vaatimukset, joilla eliminoidaan riskianalyysissä havaittujen vaarojen syntymistä (käytännössä tämä tarkoittaa muutamaa iteratiivista kierrosta vaatimusmäärittelyn ja riskianalyysin välillä).
– Suunnitteludokumentaation perusteella arvioidaan tuotteen vaatimustenmukaisuus.
– Verifioinnille ja validoinnille on omat erityisvaatimukset.

Mikäli valmistaja ei kykene osoittamaan em. vaatimusten täyttymistä, voi markkinoille saattaminen aiheuttaa lisäkustannuksia. Kustannuksia voi syntyä viranomaishyväksyntöjen yhteydessä tapahtuvina uusintatarkastuksina tai taloudellisina menetyksinä peruuntuneiden kauppojen muodossa. Usein asiakkaat asettavat kaupan ehdoksi, että tuote on hyväksytty ennen toimitusta. Edellä mainitut ongelmat aiheuttavat väistämättä myös aikataulullisia viiveitä.



Kuva 8. Suunnittelun lähtötietoihin tulee lisätä myös lakisääteiset vaatimukset.

Vaatimusten täyttyminen osoitetaan pääasiallisesti tuotteen suunnittelu–dokumentaatiosta. Tämä pätee erityisesti ohjelmistojen kohdalla. Käytännössä on kuitenkin havaittu, että ensiksi suunnitellaan tuote ja tämän jälkeen suunnitellaan lakisääteisten vaatimusten täyttyminen ja täytetään tarvittavat dokumentit. Suunnitteluprosessi tulisikin rakentaa siten, että lakisääteiset vaatimukset lisätään suunnittelun lähtötiedoiksi ja tarkastetaan ensimmäisessä katselmuksessa (kuva 8).

Tällöin tuotteen markkinoille saattaminen ja siihen liittyvä aikataulutus on paremmin ennustettavissa. Suunnittelun lähtötietojen kerääminen edellyttää ohjeistettua prosessia ja tuekseen erilaisia tarkastuslistoja, jotka sitten eri vaiheiden katselmuksissa voidaan tarkastaa (ks. kuva 8). Tällä menettelyllä saadaan myös rakennettua ohjelmiston vaatimustenmukaisuuden osoitusta myöhemmin tarvittaviin hyväksyntöihin.

Viranomaisvaatimukset voitaisiin aivan hyvin rinnastaa suoraan myös järjestelmävaatimuksiksi, jolloin niiden lisääminen suunnittelun lähtötietovaatimuksiksi olisi entistä perustellumpaa.

4.2.3 Suunnittelu

Suunnitteluvaiheessa järjestelmä-, laitteisto- ja ohjelmistovaatimuksia ruvetaan toteuttamaan. Tässä kohtaa täytyy muistaa, ettei kaikkia järjestelmävaatimuksia välttämättä voida johtaa ohjelmisto- tai laitteistovaatimuksiksi.

Suunnitteluvaiheessa vaatimuksia arvioidaan ja ne pyritään jakamaan toimintojen perusteella loogisiin osakokonaisuuksiin. Tässä vaiheessa riskianalyysin tulosten perusteella kriittisiksi arvioidut vaatimukset pyritään jakamaan arkkitehtuurisesti yhteen tai kahteen moduuliin ja vähemmän kriittiset vaatimukset pyritään sijoittamaan muihin moduuleihin (ks. kohta 4.2.3.1 Arkkitehtuuri).

Suunnitteluvaihe mielletään usein tuotteen tai ohjelmiston varsinaiseksi suunnitteluksi. Lääkintälaitteille asetetut vaatimukset edellyttävät myös osoitusta standardien vaatimustenmukaisuudesta, joka tarkoittaa tiettyjen hyväksyntätestien ja arviointien suorittamista kolmannella osapuolella (tarkastuslaitokset). Suunnitteluvaiheessa on tärkeää huomioida myös nämä testit ja aikataulutus, koska hyväksyntätestien tulos voi olla kielteinen (standardin vastaiset suunnitteluratkaisut -> suunnittelua joudutaan uusimaan).

Hyväksyntätestit voidaan osittain määritellä ja aikatauluttaa myös projektisuunnitelmassa, jolloin vaihekohtaiset suunnittelut ja hyväksyntätestit saadaan limitettyä sopivasti keskenään.

Suunnittelussa (projektisuunnitelman tulisi määritellä myös suunnitteluympäristö) joudutaan määrittelemään suunnittelun aikana käytettävät työkalut ja menetelmät. Suunnitteluvaiheessa on tärkeää huomioida, että välineet ja menetelmät ovat sopivia ja käyttökelpoisia yrityksen suunnitteluprosessin käyttöön (ks. kohta 4.3.2 Validointi).

Suunnitteluvaiheessa on tärkeää myös suunnitteluratkaisujen jäljittäminen vaatimuksiin, toteutukseen ja testaukseen.

4.2.3.1 Arkkitehtuuri

Ohjelmiston arkkitehtuurille ei löydy yksiselitteisesti hyväksyttyä määritelmää (http://www.sei.cmu.edu/architecture/definitions.html). Arkkitehtuurilla tarkoitetaan

sellaista kuvausta, jossa järjestelmä jaetaan erillisiin ohjelmamoduuleihin ja -unitteihin sekä tarkkoihin moduuli- ja unittikohtaisiin toimintoihin.

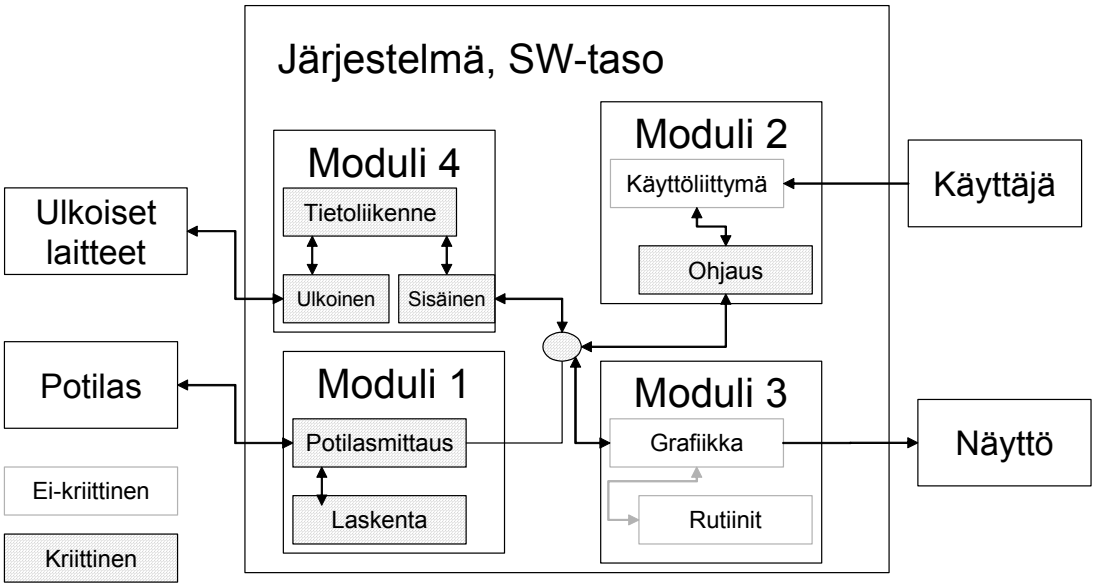
Arkkitehtuurin avulla määritellään, minkälaisia osakokonaisuuksia järjestelmässä on (hahmotetaan järjestelmää ja puretaan se selkeiksi osakokonaisuuksiksi), mitä mikäkin moduuli tekee (toiminnallisuustarkastelu ja turvallisuuteen vaikuttavien moduulien tunnistaminen) ja miten eri moduulit kommunikoivat keskenään (rajapintamäärittely).

Arkkitehtuurin avulla voidaan laatia tarkemmat ja konkreettisemmat moduuli- ja unittikohtaiset verifiointi- ja validointisuunitelmat ja saadaan laadittua selkeät tehtäväjaot eri suunnittelijoiden välille.

Arkkitehtuurimäärittely on aina tuotekohtainen, ja siinä tulee ohjelmistoarkkitehtuurin lisäksi määritellä myös mekaniikka-, sähkö- ja elektroniikka-arkkitehtuuri. Monimutkaisten laitteiden, käyttöliittymien ja ohjelmistoa sisältävin järjestelmien osalta saatetaan myös toiminnallisuus joutua määrittelemään arkkitehtuurin avulla.

Lääkintälaitteiden ohjelmistoille sovellettavan standardin (IEC 60601-1-4) vaatimukset keskittyvät olennaisesti turvallisuuteen ja suorituskykyyn (välillisesti standardin kautta ja suora vaatimus MDD:stä), joten tässä tapauksessa arkkitehtuurispesifikaatiosta pitää kyetä osoittamaan turvallisuuteen ja suorituskykyyn vaikuttavat moduulit, unitit, toiminnot sekä rajapinnat.

Arkkitehtuurille ei ole olemassa yhtä ainoaa oikeaa ratkaisua, eli valmistaja voi itse määritellä, minkälainen ohjelmistoarkkitehtuuri on ja millä suunnitteluratkaisuilla ja dokumenteilla arkkitehtuuri täyttää sille asetetut vaatimukset. Kuvassa 9 on esimerkki ohjelmiston turvallisuusarkkitehtuurista.



Kuva 9. Esimerkki ohjelmiston turvallisuusarkkitehtuurista.

Arkkitehtuurista on saatava jäljitettävyys vaatimusspesifikaatioon ja riskienhallinnan toimenpiteisiin. Vaatimus tarkoittaa käytännössä, että järjestelmälle tehdään riskianalyysi ja havaittuja riskejä poistetaan tai pienennetään tietyillä ohjelmistovaatimuksilla.

Arkkitehtuurin, vaatimusmäärittelyn ja riskienhallinnan välistä yhteyttä voidaan hahmottaa seuraavalla tavalla:

- Arvioidaan alustavasti toimintojen kautta, mitä riskejä laitteessa voi olla (alustava riskianalyysi).
- Missä riskit sijaitsevat (arkkitehtuuri)?
- Voidaanko kriittiset toiminnot sijoittaa yhteen moduuliin (turvallisuusarkkitehtuurimäärittely)?
- Miten havaittuja riskejä voidaan kontrolloida tai eliminoida (riskianalyysi, vaatimusspesifikaatio)?
- Purevatko käytetyt keinot ja ovatko ne riittäviä (riskianalyysi, vaikuttavuuden arviointi, arkkitehtuurimäärittely)?

Arkkitehtuurimäärittelyyn saattaa käytännössä tulla useampia näkymiä, kuten toiminnallisuus-, turvallisuus-, laitteisto- tai ohjelmistoarkkitehtuuri, ja sen määrittämiseen liittyy useita eri sidosryhmiä (Koskimies & Mikkonen 2005), joilla on erilaisia vaatimuksia ja näkemyksiä arkkitehtuurista. Usein vaatimukset ovat vielä ristiriidassa keskenään, joten lopullinen arkkitehtuuri on aina jonkinasteinen kompromissi eri sidosryhmien vaatimuksista.

Yrityksen tulisi pohtia mahdollisuutta kehittää ohjelmistotuotteilleen ns. yleinen turvallisuusarkkitehtuurimalli, joka määrittelee yleiset arkkitehtuuriratkaisut yrityksen suunnittelemille lääkintälaitteiden ohjelmistoille. Malli kattaisi toteutuksen eli rakenteen, suunnittelussa ja määrittelyssä tarvittavat työkalut sekä valmiit dokumenttipohjat arkkitehtuurispesifikaation dokumentoinnille sekä riskianalyysimallin, kriittisten osien tunnistamiseksi ja dokumentoimiseksi.

Lääkintälaitteiden ohjelmistosuunnittelussa voidaan turvallisuusarkkitehtuurin määrittelemistä pitää riittävänä, jos samalla huomioidaan standardin IEC 60601-1-4 vaatimukset.

Turvallisuusarkkitehtuurimallia voidaan verrata suunnittelukulttuuriin, jossa asioita suunnitellaan tietyllä ennalta hyväksi havaitulla menetelmällä. Hyväksi havaittua menetelmää voidaan soveltaa useissa eri projekteissa, mikä toisaalta alentaa suunnittelukustannuksia ja toisaalta antaa varmuuden tuotteen turvallisuudesta ja vaatimustenmukaisuudesta. Samalla malli toisi turvallisuuskriittisten ohjelmistojen arkkitehtuurin määrittelemiseen toistettavan ja ennalta tunnetun suunnittelutavan.

Mikäli turvallisuusarkkitehtuuria joudutaan päivittämään esimerkiksi ohjelmistoteknologioiden muutosten myötä, voidaan olemassa oleva arkkitehtuuriratkaisu suunnitella ja päivittää erillisenä muusta ohjelmistosuunnittelusta ja integroida olemassa oleviin ohjelmistoihin huolellisen testauksen ja pilotoinnin jälkeen.

Olipa arkkitehtuuri kuinka hyvä tahansa, niin ilman hyvää ja kattavaa dokumentointia arkkitehtuuria ei ole olemassa (Koskimies & Mikkonen 2005). Dokumentointiin perustuvat myös ohjelmistojen ylläpito ja vaatimustenmukaisuuden osoitus.

4.2.3.2 Algoritmien suunnittelu ja toteutus

Ohjelmistoa sisältävän laitteen toiminta perustuu käyttöjärjestelmän ja sovellusohjelmien suorittamiin toimintoihin. Toimintoja voi ohjata joko käyttäjä tai ohjelmat omien rajapintojensa kautta. Useita näistä toiminnoista voidaan käytännössä kutsua algoritmeiksi. Yksinkertaisimmillaan algoritmi on näppäimistön lukurutiini, jolla käyttäjän antama syöte välitetään ohjelmiston analysoitavaksi.

Ohjelmiston suorituskyvyn ja turvallisuuden kannalta monimutkaisemmat algoritmit muodostavat ohjelmiston selkärangan. Lääkintälaitteissa tehdään algoritmien laskemien tulosten perusteella hoitosuunnitelmia tai diagnooseja potilaan tilasta.

Kriittisten ja merkittävien algoritmien suunnittelu edellyttää arkkitehtuurimäärittelyä, jotta algoritmin toiminnan kannalta kriittiset ohjelmistovaatimukset voidaan sijoittaa yhteen moduuliin.

Algoritmin suunnittelu ja ohjelmointi edellyttää formaalia elinkaarimallin mukaista ohjelmistosuunnittelua, jossa algoritmin suunnittelu, riskienhallinnan toimenpiteet, koodaus ja testaus sekä kelpuutus saadaan osoitettua ulkopuoliselle arvioijalle.

Algoritmisuunnittelussa tulee dokumentoida vähintään seuraavat seikat:

- kirjallisuusviitteet
- algoritmin toiminnan rajoitukset (miten rajoitukset on havaittu: omat tutkimukset, kliininen tutkimus, kirjallisuusviitteet, riskianalyysit)
- algoritmin toiminnan ennakkoehdot (potilaana aikuinen, lapsi, potilaan tila, asetusarvot, oletusarvot, ympäristö)
- algoritmin rajapinnat (mistä ohjelmamoduuleista algoritmia voidaan ohjata tai siihen voidaan syöttää dataa, muuttaa parametreja ja mihin ohjelmamoduuleihin algoritmi välittää tietoa)
- suorituskyky (kuinka tehokas, kuinka tarkka, millä alueella, millä väestöpohjalla jne.)
- algoritmin oikeellisuuden varmistaminen (potilasdatan riittävyys, oikeellisuus, testit, validointi jne.).

Algoritmisuunnittelun tueksi voisi laatia tarkastuslistan, jolla em. asioiden täyttyminen voidaan varmistaa suunnittelukatselmusten yhteydessä.

4.2.4 Toteutus ja integrointi & testaus

Toteutusvaiheen ja integrointi & testaus -vaiheen välinen ero on perinteisessä vesiputousmallissa hieman hämärä. Molemmissa vaiheissa voidaan ohjelmoida ja kummankin vaiheen tuotosta voidaan testata. Tästä syystä vaiheiden merkitystä lääkintälaitteiden ohjelmistoille käsitellään ikään kuin yhtenä vaiheena.

Tärkeää tässä vaiheessa on varmistaa kaikkien vaatimusten ja turvallisuusvaatimusten[2] siirto tuotteeseen tai ohjelmistoon. Lisäksi täytyy arvioida toteutuksen onnistuminen ja vaikuttavuus testaamalla.

[2] Turvallisuusvaatimukset: vaatimukset, joilla riskianalyysissä havaittujen riskien toteutumista pyritään ehkäisemään.

Testaustulos tulee dokumentoida riskianalyysiin (riskienhallinnan yhteenveto) ja verifiointi- ja validointiraportteihin.

Suunnitteluratkaisujen jäljitettävyys tulee tarvittaessa viedä myös koodiin saakka.

4.2.5 Käyttöönotto ja ylläpito

Käyttöönotto kattaa ohjelmiston julkaisun, tuotannon sekä asennuksen. Julkaisussa tai ennen sitä pitää määritellä ”master median” säilytys ja siirto tuotantoon sekä määritellä ohjelmiston tunnistus (julkaisupäivä, sarjanumerointi jne.) sekä merkinnät. Hyväksyntätestauksissa kirjoitettavat lausunnot kirjoitetaan tunnistustietojen perusteella.

Ylläpidossa palataan normaaliin käyttöön ja käytön aikana mahdollisesti tapahtuviin päivityksiin joko bugikorjausten tai asiakkaiden pyynnöstä tehtäviin ohjelmistopäivityksiin.

Vaiheen merkittävänä vaatimuksena voidaan pitää sitä, että valmistajan riskienhallintaprosessin tulee kattaa myös ylläpitovaihe (kuva 10).

Kun käyttövaiheessa ilmenee ongelmia, tulee yrityksen käynnistää tarvittavat toimenpiteet ongelman ratkaisemiseksi. Mikäli ongelma on aiheuttanut vaaratilanteen, tulee ongelman merkittävyys arvioida ja tarvittaessa käsitellä.

4.3 Muut kehityskohteet

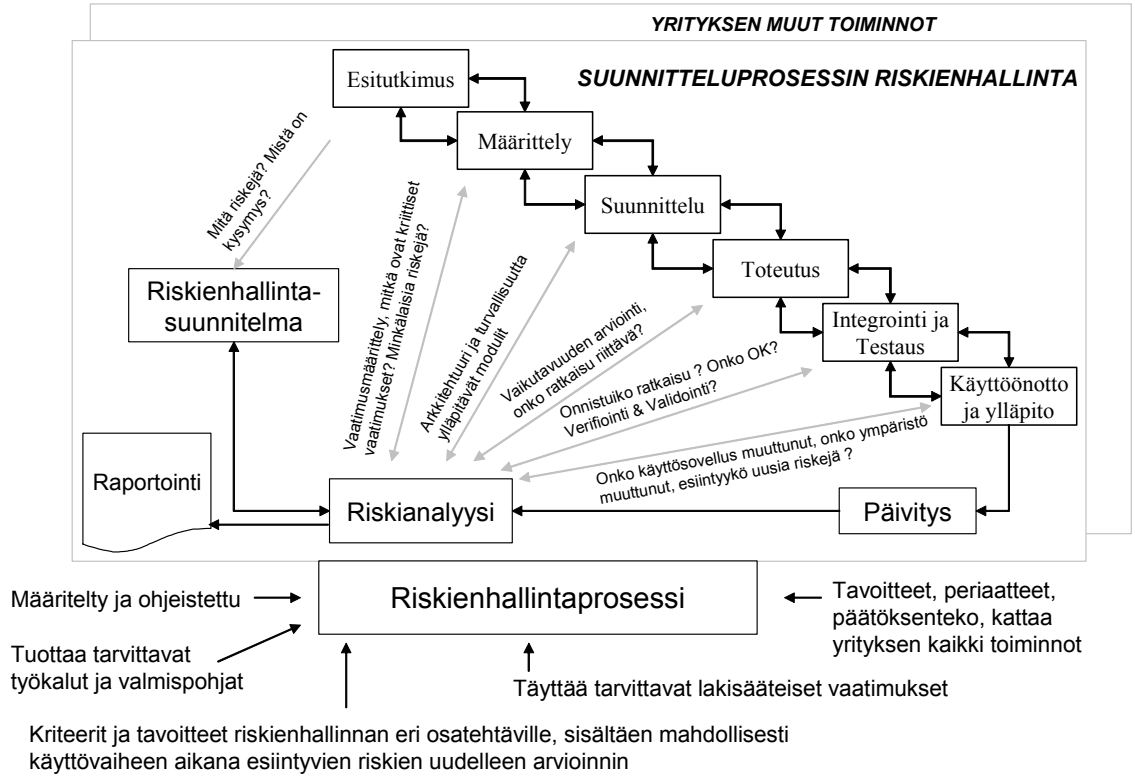
4.3.1 Riskienhallinta mukana suunnittelussa

Riskienhallinnalla tarkoitetaan johtamiskäytäntöjen, menettelytapojen ja käytännön toimien systemaattista soveltamista riskien analysoimiseksi, merkityksen arvioimiseksi ja kontrolloimiseksi. Riskienhallintaa käytetään tunnistamaan ja hallitsemaan valmistajan toimintaan liittyviä riskejä.

Lääkintälaitteiden suunnittelulta edellytetään kattavaa riskienhallintaa ja osoitusta tuotteen turvallisuudesta (MDD, ISO 14971, IEC 60601-1-4). Ilman tuotteen riskianalyysiä tätä vaatimusta on mahdoton toteuttaa.

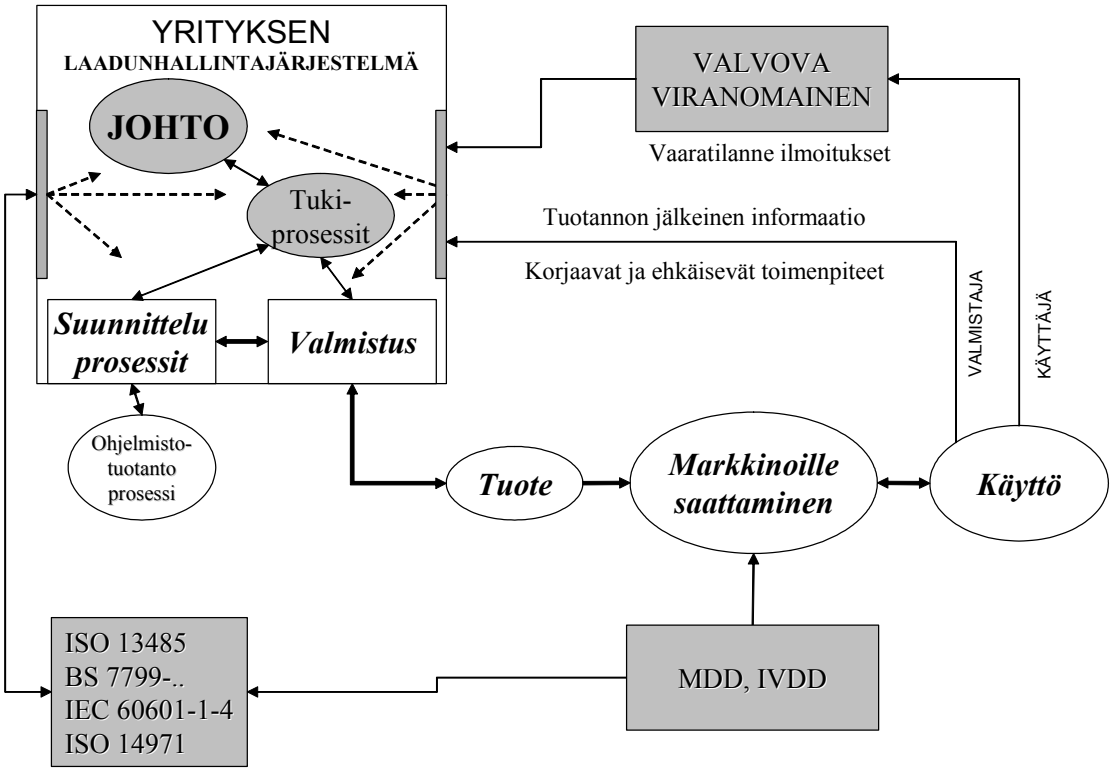
Riskienhallintaprosessin tulee asettaa vaatimukset yrityksen tavalle toteuttaa koko toiminnan kattavaa riskienhallintaa, mukaan lukien erityisesti tuotteen suunnittelun aikainen riskienhallinta. Suunnittelun aikaisen riskienhallinnan tulee kattaa koko suunnitte-

lun elinkaari. Kunkin elinkaaren vaiheen jälkeen on arvioitava, miten vaiheen aikana tehdyt ratkaisut ja toimenpiteet vaikuttavat laitteen turvallisuuteen (kuva 10).



Kuva 10. Yrityskohtainen riskienhallintaprosessi.

Riskienhallintaprosessi on kokonaisvaltainen prosessi, joka määrittelee yrityskohtaisen riskienhallinnan. Prosessi kattaa riskienhallintapolitiikan, periaatteet, työkalut, menetelmät ja kullekin yrityksen osa-alueelle soveltuvat toimenpiteet riskien analysoimiseksi. Riskienhallintaprosessi tulee toteuttaa ISO 14971:n vaatimusten mukaisesti; standardi määrittelee myös vaatimukset käyttövaiheen aikaiselle riskienhallinnalle (kuva 11).



Kuva 11. Riskienhallinnan on katettava myös käyttövaihe (ISO 14971).

Lisätietoa riskienhallinnasta on seuraavissa lähteissä:

- Lääkelaitoksen julkaisu ”Terveydenhuollon laadunhallinta, Lääkintälaitejärjestelmien turvallisuus”, http://www.nam.fi/uploads/julkaisut/Julkaisu_01_2004_04-07-06.pdf
- http://pkrh.vtt.fi/
- http://riskianalyysit.vtt.fi/
- Pöyhönen et al. 2002: Vaatimukset ohjelmistoa sisältäville lääkintälaitteille. Hallinta ja menetelmät vaatimustenmukaisuuden osoittamiseksi. VTT Tiedotteita 2150. Espoo: VTT. http://www.vtt.fi/inf/pdf/tiedotteet/2002/T2150.pdf
- GHTF:n www sivut, http://www.ghtf.org
- Google-hakusanat: ”medical device software risk management”.

Riskienhallintaan ei ole olemassa yhtä oikeaa ratkaisua. Yrityskohtaiset tarpeet ja tavoitteet sekä toimialakohtaiset vaatimukset sanelevat myös ehtoja riskienhallintaprosessille.

4.3.2 Validointi

Validoinnilla varmistetaan tuotteen asiakastarpeiden ja määriteltyjen ominaisuuksien toteutuminen. Terveydenhuollon tuotteen osalta validointi voidaan jakaa esim. suunnittelun, tuotantoprosessien ja kliinisten tulosten validointiin. Ohjelmistolla voi olla vaikutusta näihin kaikkiin osatekijöihin. Validoinnin avulla osoitetaan, että ohjelmiston vaatimukset ovat oikein ja täydellisesti toteutettu ja ne ovat jäljitettävissä tuotteen (järjestelmän, laitteen) vaatimuksiin (turvallisuus ja vaikuttavuus tai turvallisuus vastaan kliiniset hyödyt). Toimenpiteillä saadaan myös osoitettua, että ohjelmisto suorittaa aiotun toimintansa oikein, on riittävän suorituskykyinen sekä turvallinen (Pöyhönen et al. 2002) eikä direktiivin (MDD) olennaisten vaatimusten täyttyminen ole uhattuna ohjelmiston kautta.

Validointi kohdistetaan ensisijaisesti itse tuotteeseen, mutta sen tulee kattaa myös tuotteen suunnittelussa käytettävät välineet ja menetelmät sekä tuotannossa tarvittavat välineet ja laadunvarmistustoimenpiteet sekä muut tuotteeseen liittyvät erityisprosessit.

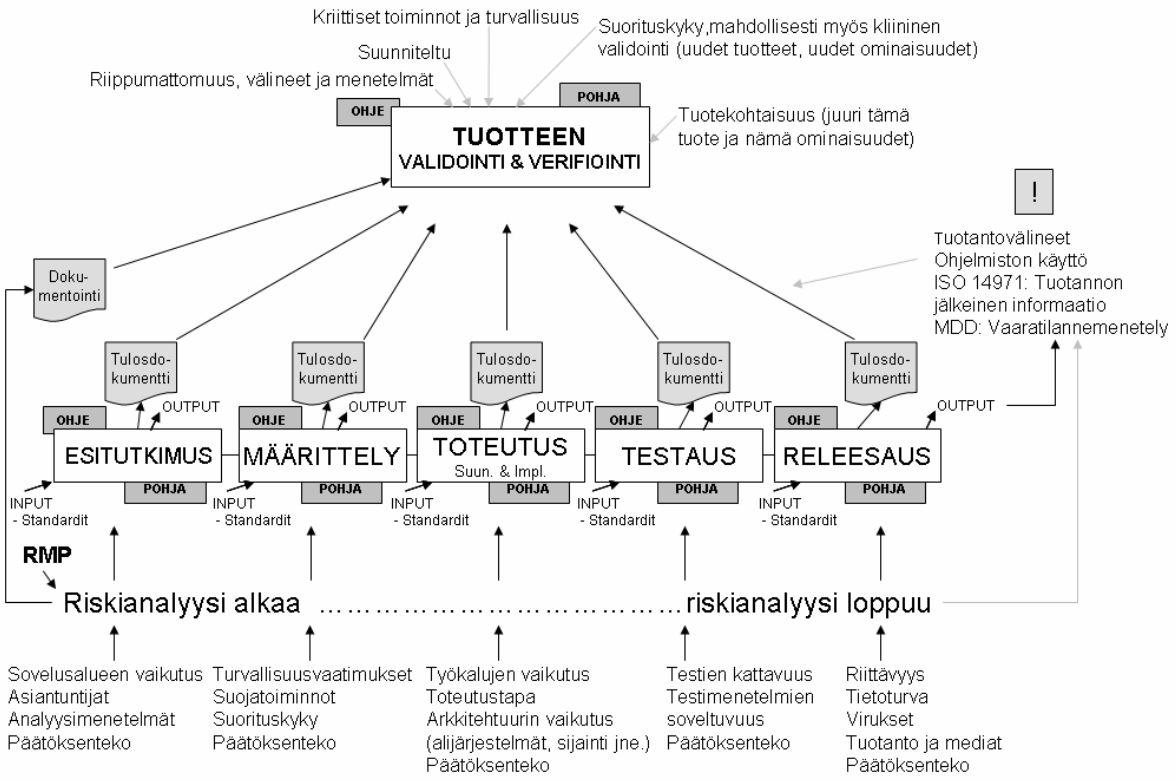
Ohjelmistosuunnittelun eri vaiheille on suunniteltu paljon erilaisia ohjelmistotyökaluja, kuten vaatimustenhallintaan, muutostenhallintaan, lähdekoodin tarkastukseen ja ylläpitoon, dokumentointiin ja versionhallintaan. Työkalut tuovat ohjelmistokehitykseen kustannussäästöä, jäljitettävyyttä ja systemaattisuutta. Näin ollen niiden käytön lisäämistä suunnittelun tukena voidaan pitää erittäin suositeltavana.

Työkalujen tehokas käyttö edellyttää yrityskohtaista räätälöintiä, konfigurointia sekä koulutusta, ja työkalujen täytyy soveltua olemassa olevaan suunnitteluprosessiin. Todennäköisesti suunnitteluprosessia joudutaan ohjeistamaan jonkin verran uusien työkalujen myötä. Tämän jälkeen käyttöönotettu työkalu tulee kelpuuttaa käyttöön joko kertaluonteisesti tai jatkuvana validointina, ja validoinnin tulee osoittaa seuraavat seikat:

- Työkalu soveltuu aiottuun käyttötarkoitukseen.
- Työkalun edellyttämät prosessikuvaukset ja menetelmäohjeet on laadittu ja käyttöönotettu.
- Riittävä koulutus on järjestetty.
- Työkalu ei heikennä suunnitellun tuotteen ominaisuuksia.
- Tapauskohtaisesti muut esille tulevat seikat otetaan huomioon.

Koska lääkintälaitteiden turvallisuus on eräs tärkeistä vaatimuksista, validoinnin avulla varmistetaan, että riskianalyysissä havaitut vaarat on muutettu vaatimusmäärittelyyn ominaisuuksiksi, joilla havaittuun vaaraan liittyvää riskiä vähimmäistetään.

Validointi ei ole kertaluonteinen toiminto, vaan se kattaa kaikki suunnittelun vaiheet aina esitutkimuksesta ohjelmiston tuottamiseen (kuva 12).



Kuva 12. Validointi kattaa kaikki suunnittelusta valmistukseen.

Validoinnille laaditaan aina projektin alkuvaiheessa suunnitelma, jota voidaan tarkentaa projektin etenemisen mukaan. Suunnitelmassa otetaan huomioon aiottu käyttötarkoitus, olosuhteet sekä turvallisuusvaatimukset, ja sen tulee käsitellä seuraavia asioita:

- tavoite, mitä ollaan tekemässä ja mihin pyritään (kohteet, kattavuus ja soveltuvuus, rajaukset)
- viitteiden, referenssien ja termien määrittely
- resurssien määrittely, kuka suunnittelee, kuka verifioi, kuka validoi (vastuut, pätevyys ja aikataulutus, riippumattomuus)
- se, mitä tehdään (määritellään kullekin ohjelmistotuotannon elinkaaren vaiheelle tarkoituksenmukaiset ja riittävän kattavat toimenpiteet)
- välineet, menetelmät, kuka tekee, missä vaiheessa tekee, mitä pitää dokumentoida
- hyväksyntäkriteereiden määrittely ainakin alustavasti (voivat täydentyä ja tarkentua suunnittelun edetessä)

- tuotteelle ominaisten suorituskyky-, turvallisuus- ja viranomaisvaatimusten täyttyminen
- kliiniset kokeet, tulokset sekä raportointi
- lakisääteisten vaatimuksien huomioiminen (IEC 60601-1-4: turvallisvaatimukset, riippumattomuus, ammatilliset suhteet, suunnittelutiimin jäsen ei validoi omaa suunnitteluaan).

Validoinnista on lisää tietoa seuraavissa lähteissä:

- Pöyhönen et al. 2002: Vaatimukset ohjelmistoa sisältäville lääkintälaitteille. Hallinta ja menetelmät vaatimustenmukaisuuden osoittamiseksi. VTT Tiedotteita 2150. Espoo: VTT. http://www.vtt.fi/inf/pdf/tiedotteet/2002/T2150.pdf
- IEEE 1012: IEEE Standard for Software Verification and Validation
- IEEE 1059: Verification and Validation Plans.

Validointiprosessi nähdään usein tukiprosessina ohjelmistotuotantoprosessissa (ISO/IEC 12207), mutta sen laatua parantavaa vaikutusta ei tule väheksyä. Validointi on toiminto, jolla voidaan parantaa suunnitteluprosessia ja varmistaa, että ohjelmisto täyttää sille asetetut vaatimukset. Se myös auttaa luomaan ja kehittämään yrityksen toimintoja, tehostamalla korjaavia toimenpiteitä, vähentämällä tuotteen takaisinkutsuja (recall) ja pienentämällä kustannuksia pitkällä aikavälillä.

Tehokkaiden validointimenetelmien avulla myös ohjelmistomuutokset tehdään luotettavammin ja hallittavammin (on huomattava, että hyvin useat ohjelmistoprojektit ovat vanhan olemassa olevan ohjelmiston modifiointia) ja parannetaan tuotekehitystä tukevaa dokumentaatiota ja raportointia sekä varmistetaan laatutavoitteiden toteutumista.

5. Extreme Programming -malli

Extreme Programming tai lyhyemmin XP on Kent Beckin kehittämä kevyt ja kurinalainen ohjelmistokehitysmalli, joka perustuu yksinkertaisuuden, kommunikoinnin, palautteen ja vakaumuksen periaatteisiin (http://www.xprogramming.com/what_is_xp.htm). XP on suunniteltu erityisesti pienten tiimien käyttöön, jotka kehittävät ohjelmistoja alati muuttuvassa ympäristössä, jossa myös ohjelmistovaatimukset muuttuvat jatkuvasti.

XP perustuu hyvin pitkälle henkilöiden tai roolien keskinäiseen kommunikointiin ja 12 erilaiseen avainkäytäntöön. Menetelmässä on useita olennaisia muutoksia ja uusia mahdollisuuksia perinteiseen vesiputousmalliin nähden. Pitää kuitenkin huomioida mahdolliset menetelmän rajoitukset, silloin kun ohjelmistolle asetetaan tiukkoja lakisääteisiä vaatimuksia tai ulkopuolinen taho joutuu arvioimaan ohjelmiston turvallisuutta. Tämä aiheuttaa sen, että yrityksellä tulee olla rohkeutta soveltaa teoreettista mallia ja lisätä siihen omia yrityskohtaisia ja toimialakohtaisia menetelmiä.

XP:hen siirtyminen on yritykselle iso ponnistus, ja siihen saattaa vaikuttaa yrityksessä tapahtuva organisaatiomuutos, kilpajuoksu eri yritysten välillä tai pyrkimys lyhentää entisestään suunnittelun aloituksen ja markkinoille saattamisen välistä aikaa.

Malli tuo mukanaan paljon uusia termejä ja käytäntöjä, jotka edellyttävät sisäistä koulutusta ennen menetelmän käyttöönottoa.

5.1 XP:n avainkäytännöt lyhyesti

XP:n kaksitoista avainkäytäntöä (http://www.xprogramming.com/what_is_xp.htm) esitellään tässä lyhyesti ja tarkastellaan ko. käytäntöjen soveltuvuutta ja merkitystä lääkintälaitteen ohjelmistojen suunnitteluun:

- Suunnittelupeli (The Planning Process, The Planning Game)

  XP-suunnittelupelissä XP-asiakas määrittelee halutut ominaisuudet ja niiden kaupallisen merkityksen. Tämän jälkeen hän ohjelmoijien tuottaman kustannusarvion perusteella valitsee toteutettavat ominaisuudet ja arvioi, mitkä ominaisuudet siirretään myöhemmäksi. XP:n suunnittelupelin teho on siinä, että sen on helppo ohjata projekti onnistuneeseen lopputulokseen.

  Suunnittelupeli vastaa aika pitkälle esimerkiksi vesiputousmallin esitutkimusta ja määrittelyvaihetta mutta poikkeaa siinä, että nyt määritellään vain sen verran vaatimuksia, että kyseinen iteraatio saadaan toteutettua toimivaksi ohjelmaversioksi.

Suunnittelupeliin tulee integroida kaikki toimialakohtaiset ja lakisääteiset vaatimukset silloin, kun kehitetään yrityskohtaista mallia.

- Pienet julkistukset (Small Releases)

  XP-tiimi julkaisee ohjelmiston hyvin nopeasti ja päivittää sitä säännöllisesti hyvin lyhyissä jaksoissa.

  Ohjelmistojulkistuksia voidaan tehdä vaikka päivittäin. Julkistukset tulee ymmärtää yrityksen sisäisinä julkistuksina eikä kaupallisina julkistuksina, koska jokainen julkistettu versio, jonka yritys toimittaa kaupalliseen käyttöön, on hyväksytettävä.

  Lyhyin väliajoin tapahtuva julkistus nopeuttaa testaamista ja vakauttaa koodia, koska ohjelmistomuutokset julkistusten välillä on helpompi testata ja arvioida.

- Kielikuva (Metaphor)

  Yhteinen kielikuva pyrkii tarjoamaan suunnittelutiimille yhteisen tavan keskustella ja poistaa monimutkaisten asioiden tuomia ongelmia, parantaa kommunikaatiota ja tehostaa suunnitteluratkaisujen syntymistä. Löyhästi mukaillen kielikuva voisi täten olla osittain arkkitehtuurin määrittelemistä eri sidosryhmien kesken.

  Asian voisi kiteyttää siten, että yrityksellä on yhteinen terminologia, jota käytetään suunnittelussa ja tuotekohtaisissa suunnitteludokumenteissa. Yhteinen terminologia yhdistää eri sidosryhmiä ja parantaa niiden kommunikointia.

- Yksinkertainen suunnittelu (Simple Design)

  XP:llä toteutetun ohjelman tulisi olla yksinkertaisin ohjelma, joka täyttää ko. iteraatiokierroksen vaatimukset. Ohjelmistoon ei jää kovinkaan paljoa ns. tulevaisuudessa toteutettavaa. Sen sijaan painopisteenä on täyttää ohjelmiston liiketaloudelliset tavoitteet. On kuitenkin varmistettava, että ohjelmisto suunnitellaan hyvin ja suunnittelussa noudatetaan koodin muokkauksen periaatteita.

- Testaus (Testing)

  XP-tiimi keskittyy ohjelmiston validointiin koko ajan. Ohjelmoijat kehittävät ohjelmistoa kirjoittamalla ensiksi testit, sitten ohjelmiston, joka täyttää testien edellyttämät vaatimukset. Asiakkaat tuottavat hyväksyntätestit, joilla varmistetaan ja todetaan heidän haluamiensa ominaisuuksien toteutuminen.

  XP:n perusajatus on, että testausta tehostetaan automatisoidulla testauksella. Menettelytapa parantaa tuottavuutta, ohjelmiston laatua ja vakautta. Testaukselle on laadit-

tava kuitenkin kriteerit siitä, mitkä testit voidaan automatisoida ja mitkä testeistä joudutaan edelleen testaamaan perinteisin menetelmin.

Lääkintälaitteiden ohjelmistoista automatisoinnin piiriin voidaan liittää selvät on-off-tyyppiset tapaukset, jotkin vasteaikamittaukset ja kuormitustestit sekä moduulien sisäisiin ja ulkoisiin rajapintoihin liittyvät testit.

Lääkintälaitteiden ohjelmistoja ei kuitenkaan voida kokonaan testata automaattisesti, koska ohjelmiston (suorituskyky, turvallisuus) monimutkaiset algoritmit muodostavat ohjelmiston toiminnan aiotussa käyttötarkoituksessa ja ympäristössä.

Käytännössä tämä tarkoittaa sitä, että ohjelmiston avulla tehtävät diagnoosit tai hoitopäätökset vahvistetaan aina hoitohenkilökunnan päätöksellä. Päätöksenteon tukena käytetään ohjelmiston laskemien tuloksien lisäksi tapauksesta riippuen ainakin potilaan ulkoisen tilan arviota, haastattelutuloksia, potilaskertomustuloksia sekä arviota hoidon vaikutuksesta.

Tämänkaltaista ketjua ei käytännössä voida mitenkään toteuttaa automatisoidulla testauksella. Tästä syystä täytyy yrityskohtaisesti pohtia, minkälaiset testitapaukset voidaan toteuttaa automatisoidusti ja mitkä tapaukset jäävät edelleen perinteisen testauksen pariin.

– Koodin muokkaus (Refactoring)

  XP-tiimi parantaa järjestelmän suunnittelua koko kehityskaaren ajan. Tämä toteutetaan pitämällä ohjelmisto puhtaana, ilman kaksoiskappaleita (sorsa.cpp, sorsa1.cpp, sorsax1.cpp), hyvin kommunikoivana (koodi puhdasta, selkeää, suunnittelijat tietävät toistensa tekemiset, kommentoitua), yksinkertaisena ja loppuunsaatettuna.

  Koodin muokkaus tarkoittaa tässä yhteydessä sitä, että ohjelmiston toiminnallisuuteen ei puututa mutta toiminnallisuuden mahdollistavia teknisiä ratkaisuja voidaan muuttaa.

  Koodin muokkaus on erittäin hyödyllinen käytäntö, joka tulee määritellä ohjelmistokehityksen toimintaohjeisiin. Ohjeissa pitää määritellä tarkat kriteerit uudelle vaatimukselle, muutokselle, bugikorjaukselle ja koodin muokkaukselle (ks. kohta 4.2.2.3).

– Pariohjelmointi (Pair Programming)

  XP:llä ohjelmoivat työskentelevät pareissa yhdellä koneella. Tällä parannetaan koodin laatua ja tuotetaan parempia ohjelmistoja. Tällä voidaan saavuttaa alemmat kustannukset kuin, että yksi ohjelmoija työskentelisi yksin.

Pariohjelmointi on erittäin hyvä käytäntö, joka varmaan parantaa koodin laatua ja ohjelmiston vakautta. Menetelmää ei välttämättä ole kuitenkaan aina mielekästä soveltaa käytäntöön. Joskus saattaa olla esimerkiksi niin pieniä ja selkeitä ohjelmoitavia osakokonaisuuksia, että niiden ohjelmointiin ei ole järkevää kiinnittää kuin yksi ohjelmoija.

Pariohjelmoinnin periaatteisiin voi tutustua tarkemmin www-osoitteessa http://www.pairprogramming.com/.

- Lähdekoodin yhteisomistus (Collective Ownership)

  Projektissa kehitetty koodi kuuluu kaikille ohjelmoijille. Tämä sallii tiimin edetä vauhdikkaasti, koska jos jotain tarvitsee muuttaa, se voidaan tehdä ilman viiveitä. Versionhallintaohjelmistojen käyttö tukee myös tätä käytäntöä.

  Käytäntö sisältää ristiriidan luotettavuuden ja jäljitettävyysvaatimusten kanssa. Mikäli jotain pitää muuttaa, sen vaikutus pitää analysoida. Silloin, kun muutos vaikuttaa muiden ohjelmoijien koodiriveihin, ne pitäisi myös analysoida.

- Jatkuva integrointi (Continuous Integration)

  XP-tiimi integroi ja rakentaa ohjelmistoa useita kertoja päivässä. Tämä pitää kaikki ohjelmoijat samalla tasolla ja sallii hyvin nopean edistymisen. Yllättävää kyllä, useammin tapahtuva integrointi eliminoi integrointiongelmia, kun taas integrointiongelmat kiusaavat enemmän harvemmin integroivia tiimejä.

- 40 tunnin työviikko (40-hour Week)

  Väsyneet ohjelmoijat tekevät virheitä. XP-tiimi ei tee kohtuuttomasti ylitöitä, pitää itsensä pirteänä, terveenä ja tehokkaana.

  Käytännöllä on vaikutusta lopputulokseen mutta ei välttämättä kovinkaan paljon prosessikuvauksiin. Yrityskulttuurin ja henkilöstön hyvinvoinnista huolehtimisen pitäisi periaatteessa hoitaa sama asia.

- Asiakas mukana suunnittelussa (On-Site Customer)

  Asiakas on aktiivisesti mukana XP-projektissa, jolloin hän voi määrittää vaatimuksia, asettaa prioriteetteja ja vastata ohjelmoijien esittämiin kysymyksiin. Tällä on vaikutusta kommunikoinnin parantumiseen ja dokumentointitulosteiden vähentymiseen – usein eräs kalleimmista ohjelmistoprojektin vaiheista.

Asiakas tulee mieltää tässä kohdassa rooliksi, jota edustavat yrityksen sopiviksi arvioimat tahot. Sopivat henkilöt voisivat olla esimerkiksi markkinoinnista tai viranomaisvaatimuksista vastaavat. Tärkeää olisi myös saada suunnittelutiimiin mukaan oikea loppukäyttäjä.

Asiakkaan mukanaolo suunnittelussa tuottaa varmaan paremman lopputuloksen. Sen varjolla ei kuitenkaan voida vähentää lääkintälaitteiden ohjelmistojen dokumentointitarvetta.

- Standardit ja tyylioppaat (Coding Standard)

  Termi ”standardit ja tyylioppaat” (Coding Standard) tarkoittaa XP:ssä lähdekoodin yhdenmukaisuutta, jossa tiimin käytettävissä oleva lähdekoodi on kirjoitettu samankaltaisesti (lähdekoodi on varustettu muutoshistorialla, kommenteilla; muuttujamäärittelyt ovat samankaltaiset jne.). Tämä edellyttää, että yritys soveltaa viimeisimpiä sovellettavia standardeja ja laatii niistä yrityskohtaiset tyylioppaat. Säännöillä voidaan varmistaa, että koodia on helppo lukea, tarkistaa ja se on yhteensopiva. Tämä helpottaa tiimiä, joka työskentelee tehokkaasti pareissa ja jakaa koodia myös muille tiimin jäsenille

  Laajasti tämä tarkoittaa sitä, että on hyvä, että noudatetaan standardeja ja niiden pohjalta laaditaan omia ohjelmoinnin tyylioppaita.

Käytäntöjen kuvaukset on kirjoitettu erittäin suppeasti, ja metodologiaan ja käytäntöön voi tutustua tämän kohdan alussa mainitulla www-sivulla. Tärkeää tässä yhteydessä on tunnistaa ko. käytännön käyttökelpoisuus, mahdollisuudet tai ristiriidat lääkintälaitteiden ohjelmistojen suunnittelussa.

XP-metodologiasta on paljon kirjallisuutta ja www-sivuja. Seuraavassa listataan muutamia lähteitä, joiden avulla voi tutustua tarkemmin avainkäytäntöihin ja XP-teoriaan.

Kirjallisuuslähteet:

- Kent Beck: Extreme Programming Explained – Embrace Change
- Jennifer Stapleton: DSDM Dynamic Systems Development Method: The Method; ISBN 0-201-17889-3
- David Astels, Granville Miller & Miroslav Novak: A Practical Guide to eXtreme Programming; ISBN 0-13-067482-6
- sivusto http://www.xprogramming.com, erittäin kattava valikoima XP:hen liittyvää kirjallisuutta.

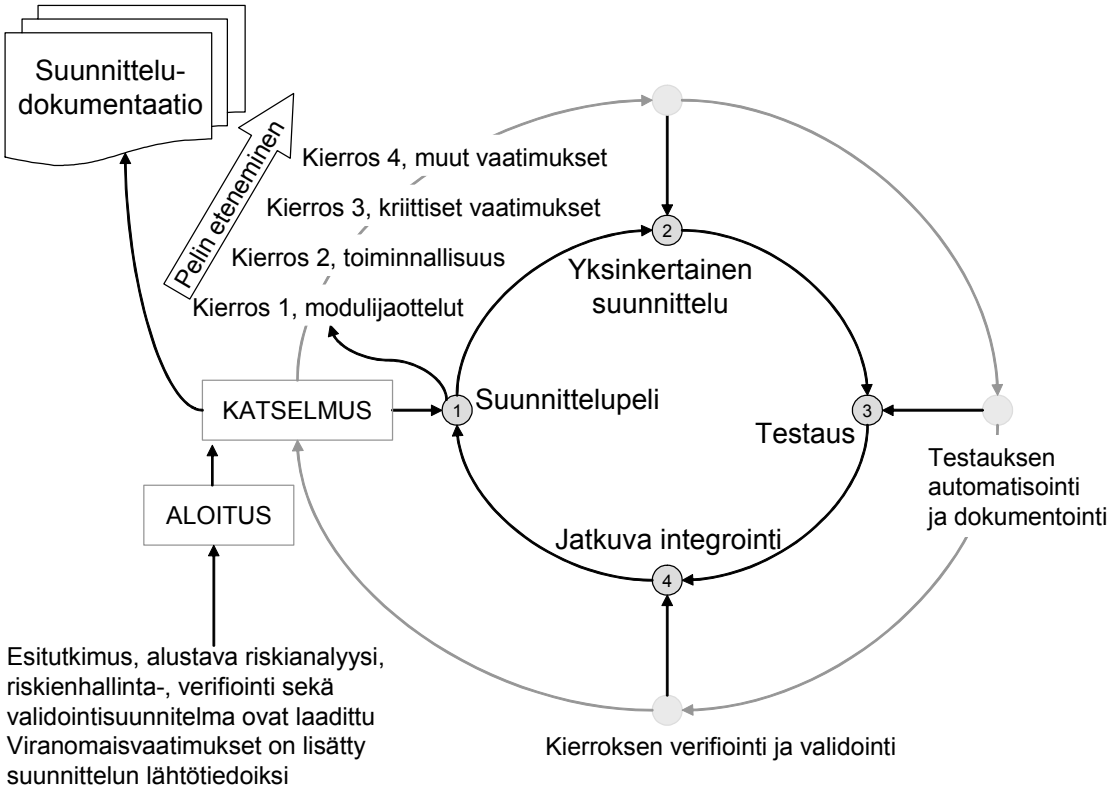
Www-linkit

– XProgramming.Com an agile Software development resource	http://www.xprogramming.com http://www.xprogramming.com/what_is_xp.htm
– Extreme Programming: A gentle introduction	http://www.extremeprogramming.org/
– http://www.extrememodeling.org/	Welcome to eXtremeModeling.org!
– Agile Alliance	http://www.agilealliance.org/
– Books by Agile Alliance Members	http://www.agilealliance.org:8080/allbooks
– Welcome to Portal of Agile Software Development methodologies at VTT Electronics	http://agile.vtt.fi
– DSDM (Dynamic Systems Development Method) Consortium	http://www.dsdm.org

5.2 XP-luonnos lääkintälaitteiden ohjelmistosuunnitteluun

Seuraavassa kuvataan esimerkki lääkintälaitteiden ohjelmistojen suunnittelemiseksi XP-mallia soveltaen.

Ennen virallista tuotekehitysprojektin aloittamista on ohjelmistoa saatettu prototypioida jo useiden kuukausien ajan. Prototypioinnin tulosten perusteella voidaan tehdä päätös hankkeen lopettamisesta tai käynnistämisestä. Mikäli kehityshanke päätetään käynnistää, laaditaan hankkeelle projektisuunnitelma, riskienhallintasuunnitelma sekä verifiointi- ja validointisuunnitelma, ja hankkeen vaatimat resurssit määritellään myös tässä vaiheessa.



Kuva 13. Luonnos lääkintälaitteen ohjelmiston suunnittelemiseksi XP-mallia soveltaen.

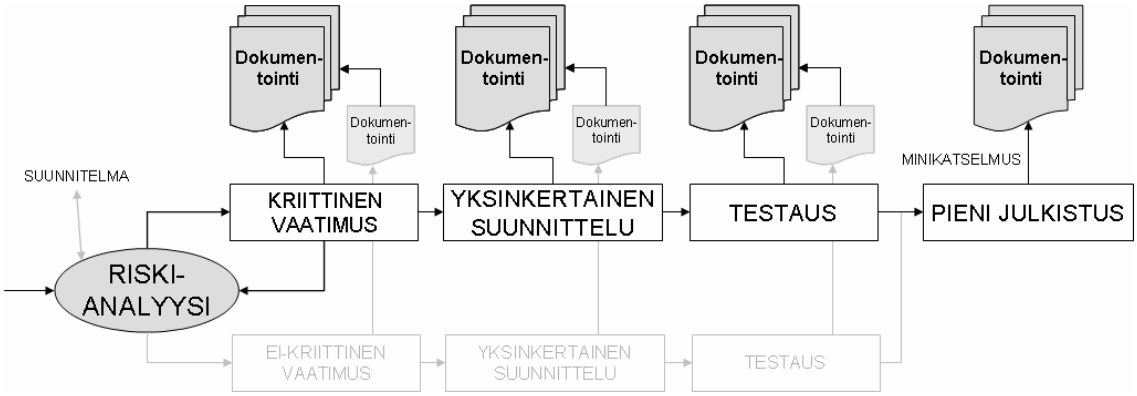
Alustavat riskienhallintaan liittyvät kysymykset tulee dokumentoida, ja niiden avulla laaditaan tuotekohtainen riskienhallintasuunnitelma.

Yrityskohtainen XP alkaa tässä vaiheessa (kuva 13). Tästä eteenpäin tapahtuvaan suunnitteluun, joka tähtää markkinoille saatettavan tuotteen suunnittelemiseksi, pitää soveltaa kaikkia lääkintälaitteelle ja ohjelmistolle sovellettavia vaatimuksia.

Optimaalisin tapa aloittaa ovat arkkitehtuurin perustuvat XP-iteraatiot, joissa ensimmäisillä kierroksilla toteutetaan moduulijaottelu ja toisella kierroksella hieman toiminnallisuutta. Näillä iteraatioilla riskienhallinnan toimenpiteet sekä dokumentaatio voivat olla selvästi vähäisempiä. Käytännössä ensimmäisellä tai toisella kierroksella ei voida toteuttaa kriittisiä vaatimuksia, turvallisuutta ylläpitäviä toimintoja tai suorituskykyyn liittyviä algoritmeja, koska suunnittelu on vielä liian aikaisessa vaiheessa.

Esimerkiksi kolmannella iteraatiokerralla toteutetaan suorituskyvyn kannalta kriittiset vaatimukset sekä turvallisuutta ylläpitävät ja varmistavat ominaisuudet. Jotta näin voidaan menetellä, on suunnittelussa tunnistettava kriittiset ja ei-kriittiset vaatimukset (kuva 14). Jaottelu edellyttää riskienhallintaprosessia ja analyysiä. Tällä kierroksella riskit analysoidaan erittäin kattavasti sekä arvioidaan riskienhallintatoimenpiteiden tehok-

kuus. Verifiointi- ja validointiraportit ja muu tarvittava suunnitteludokumentaatio täydennetään tuloksia vastaavaksi kierroksen aikana.

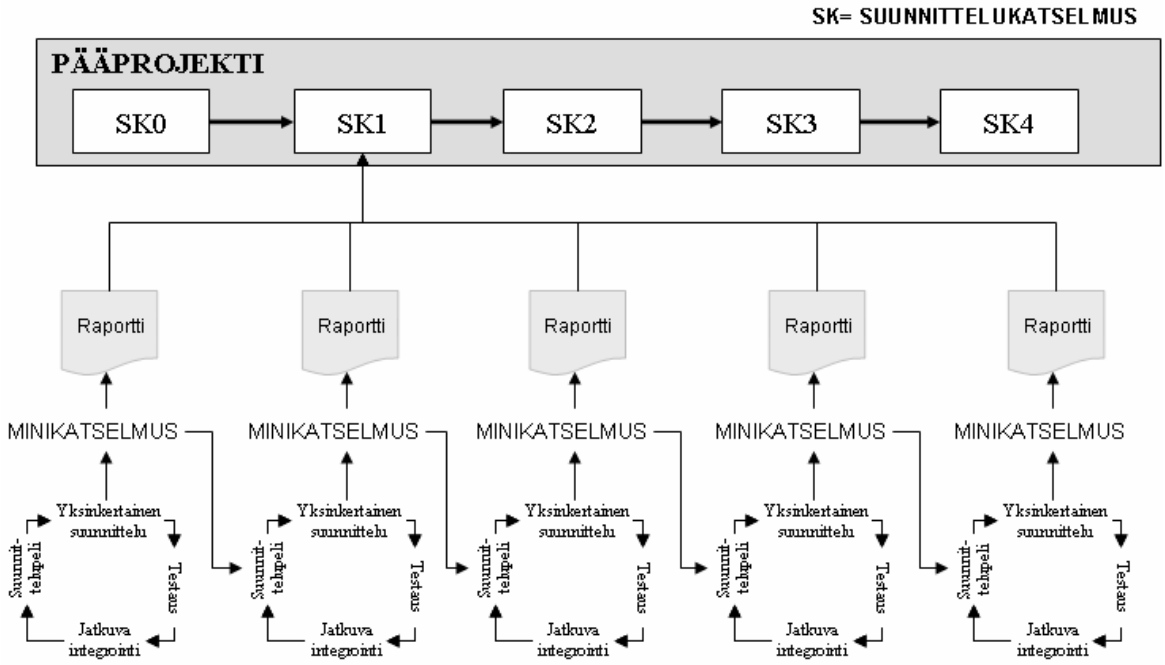


Kuva 14. Kriittiset vaatimukset toteutetaan saman kierroksen aikana.

Kullekin suunnittelukierrokselle valittavia ominaisuuksia valittaessa korostuu suunnittelupelin ja arkkitehtuurin merkitys (vrt. kielikuva vaikeiden asioiden hahmottamisessa ja selkiyttämisessä tai käyttäjätarinat), jonka avulla järjestelmä voidaan jaotella selkeisiin toteutettaviin osakokonaisuuksiin.

XP-katselmuksia voitaisiin kutsua ns. minikatselmuksiksi. Minikatselmuksissa todetaan vaiheen tuloksien täyttävän suunnitelmissa asetetut vaatimukset, minkä jälkeen voidaan siirtyä seuraavan vaatimusjoukon toteuttamiseen.

Minikatselmukset mahdollistavat ohjelmistokehityksen riippumattoman etenemisen laitteen muiden ominaisuuksien suunnittelusta. Pääprojektin suunnittelukatselmukset on aikataulutettu mekaniikka-, elektroniikka- ja sähkösuunnittelun mukaan, jolloin suunnittelukatselmuksia voidaan pitää ainoastaan muutamien kuukausien välein, kun taas XP-mallissa yhden iteraation välinen aika saattaa olla minimissään jopa 1–2 viikkoa.



Kuva 15. Minikatselmukset mahdollistavat riippumattoman etenemisen.

Minikatselmuksissa on yleensä paikalla ko. vaiheen suunnittelijat ja projektipäällikkö sekä tarpeellinen määrä muita laadunvarmistuksesta vastaavia henkilöitä. Tilaisuuden tehostamiseksi vaiheen tehtävistä voidaan laatia valmiit kysymyslistat, jotka kokouksen puheenjohtaja käy läpi ja kirjaa saadun vastineen kysymyslistaan.

Minikatselmuksista laaditaan pöytäkirja, jonka liitteeksi voidaan liittää tarkastuslistat kussakin katselmuksessa läpikäydystä asiasta. Tällöin pääprojektin katselmusten yhteydessä minikatselmukset voidaan käydä melko nopeasti läpi (kuva 15).

6. Yhteenveto

Julkaisussa on kuvattu lääkintälaitteiden ohjelmistokehitykseen liittyviä ongelmia ja esitetty joitain keinoja parantaa ohjelmistosuunnittelua ja tätä kautta kohentaa ohjelmistotuotteen laatua ja turvallisuutta.

Esimerkkeinä on käytetty pääasiallisesti perinteistä vesiputousmallia, mutta muutamia esimerkkejä on käsitelty myös ketterän ohjelmistokehityksen näkökulmasta.

Ohjelmistokehityksen parantamiseksi ei ole olemassa yhtä ainoaa patenttiratkaisua, jolla kaikki ongelmat poistuvat. Yrityskohtaisten kehityskohteitten valintaan vaikuttavat yrityskulttuuri, toimiala, käytetyt suunnittelumenetelmät ja työkalut sekä monet muut seikat.

Yhteenvetona voisi kuvata seuraavat toimenpiteet, jotka hankkeen aikana erityisesti nousivat esille.

6.1 Terminologian ja kommunikoinnin merkitys suunnittelussa

Yrityksen suunnitteluosasto muodostuu nykyisin useista eri sidosryhmistä, toimialoista ja asiantuntijoista, ja eri osastot voivat sijaita jopa eri paikkakunnilla tai mantereilla. Tästä syystä suunnittelun aikainen tiedonvälitys vaikuttaa suoraan suunnittelun lopputulokseen.

Suunnittelun aikainen tiedonkulku auttaa vähentämään virheellisiä suunnitteluratkaisuja ja parantaa osaltaan myös suunnittelun kustannustehokkuutta. Erittäin tärkeää suunnittelun tiedonkulussa olisi saada haltuun ns. hiljainen tieto, joka on syntynyt yrityksen eri asiantuntijoille vuosien kokemuksesta, erehdyksistä, virheistä sekä toistoista. Täytyy huomata vielä, että eri asiantuntijoilla on hallussaan erilaista hiljaista tietoa, joka kuitenkin tukee toinen toistaan.

Tiedonkulun parantaminen edellyttää yritykseltä yhteisen suunnitteluterminologian määrittelemistä ja kaikki sidosryhmät kattavaa koulutusta. Hiljaisen tiedon haltuun ottaminen tapahtuu esimerkiksi yhteistyötä ja avoimuutta lisäämällä.

XP:n avainkäytännöt mahdollistavat ainakin teoriassa paremman suunnittelun aikaisen kommunikoinnin ja tiedonkulun. Käytännöt täytyy kuitenkin avata ja määritellä yrityksen suunnittelua tukevissa toimintaohjeissa.

Vaatimustenhallinnan työkalut myös osaltaan (oikein käytettynä) parantavat tiedonkulun kehittymistä ja virheistä oppimista.

6.2 XP- ja lääkintälaitteiden ohjelmistosuunnittelu

Ohjelmistotuotantoprosesseja on kehitetty useita erilaisia, ja jokaisessa niissä on tietty oma ideaalinen tapa, jolla ohjelmistoja suunnitellaan. Ohjelmistotuotantoprosessia kehitettäessä tulee yrityksen huomioida se, että ei ole olemassa valmista ohjelmistotuotantomallia, joka soveltuu sellaisenaan yrityksen omaan käyttöön, koska ideaalimalli

- ei välttämättä huomioi yrityksen omia tarpeita
- ei välttämättä istu yrityksen laadunohjausjärjestelmään sellaisenaan
- ei välttämättä huomioi riittävästi toimialakohtaisia vaatimuksia (esimerkiksi terveydenhuollon tuotteita ja ohjelmistoa: MDD, ISO 14971, IEC 60601-1-4, ISO 13485 lääkintälaitteet).

XP-malli on kuitenkin erittäin varteenotettava vaihtoehto lääkintälaitteiden ohjelmistokehitykseen. Mallissa on useita avainkäytäntöjä, jotka mahdollistavat paremman kustannustehokkaan ja nopean suunnittelun sekä paremman tiedonkulun suunnittelun aikana.

Puhdas metodologia ei huomioi lääkintälaitteiden ohjelmistosuunnittelulta edellytettäviä lakisääteisiä vaatimuksia, joissa korostuvat riskienhallinta, suunnitteludokumentaation perusteella tapahtuva vaatimustenmukaisuuden arviointi, jäljitettävyys sekä validointi.

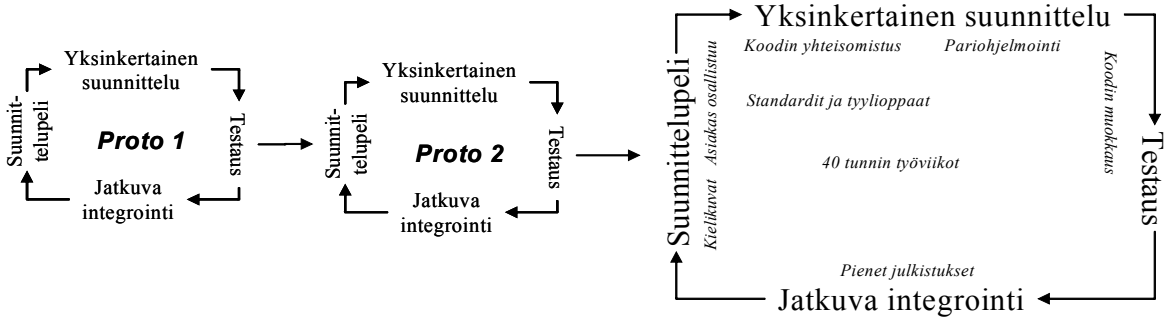
Yritykseltä vaaditaankin XP-mallia käyttöönotettaessa rohkeutta soveltaa ja ottaa käyttöön ainoastaan metodologian hyvät puolet ja jättää sellaiset ominaisuudet pois, jotka eivät jostain syystä sovellu yrityksen omaan käyttöön.

Tämän jälkeen prosessia on pilotoitava riittävässä laajuudessa ja tehtävä pilotoinnissa havaittujen ongelmien korjaukset prosessimalliin. Tämän jälkeen laaditaan lopulliset prosessin edellyttämät menetelmäohjeet ja dokumenttipohjat ja otetaan prosessi käyttöön.

Edellytyksenä XP-mallin tehokkaalle käyttöönotolle lääkintälaitteiden ohjelmistosuunnittelussa ovat seuraavat seikat:

- Vaatimusmäärittelyprosessi, jossa vaatimukset voidaan jakaa riskianalyysin tulosten perustella kriittisiin ja ei-kriittisiin vaatimuksiin.
- Projektihallinta tapahtuu erillisenä osa-alueena ja tarvittavat riskienhallinta-, verifiointi- ja validointisuunnitelmat on laadittu ennen suunnittelupelin aloitusta (mitä paremmin pöytä on katettu ennen ruokailua, sitä varmemmin päästään syömään ilman välikohtauksia).

– Ennen virallisen suunnittelupelin aloitusta tehdyt protoiluversiot tukevat suunnittelua ja arkkitehtuurimäärittelyä. Tästä syystä protoilun katsotaan tarkentavan ja varmentavan alustavia suunnitelmia ja tätä kautta parantavan suunnittelun lopputulosta.
– Mikäli protoilun aikana tehtyjä tuloksia käytetään osana suunniteltavaa ohjelmistoa, tulee kaikkien protoiluvaiheen aikaisten tulosten täyttää lääkintälaitteen ohjelmistoille asetettavat vaatimukset (kuva 16).



Kuva 16. Protoiluvaiheiden tuloksia saa käyttää osana valmista tuotetta vain silloin, kun prototuotokset täyttävät kaikki tarvittavat vaatimukset (riskienhallinta, jäljitettävyys, dokumentointi).

XP-mallin käytännöt ja periaatteet mahdollistavat ainakin kustannus- ja aikataulusäästöjä. Jatkuva integrointi mahdollistaa osaltaan pienempien osakokonaisuuksien toteuttamisen, joka selkiyttää suunnittelua huomattavasti.

Aika näyttää, kuinka XP-malli siirtyy vakiintuneena menetelmänä tuotantokäyttöön, mutta mahdollisuuksia menetelmällä kuitenkin on.

6.3 Yhteistyö

Lääkintälaitteiden ohjelmistojen turvallisuuden määrittäminen ja arviointi poikkeaa oleellisesti esimerkiksi lääkintälaitteen sähköturvallisuusvaatimuksista. Siinä missä sähköturvallisuudelle pystytään määrittelemään esimerkiksi tarkat vuotovirta- ja jännitekestovaatimukset sekä vaatimusten täyttymisen varmistavat hyväksyntätestit, niin ohjelmistoilta puuttuu selkeät hyväksyntäkriteerit ja -prosessit, joilla ohjelmistojen todetaan täyttävän asetetut vaatimukset.

Selkeiden hyväksyntäkriteerien ja arviointimenetelmien puute on saattanut luoda eri tarkastuslaitosten kesken erilaisia hyväksymis- ja arviointikäytäntöjä ohjelmistojen vaatimustenmukaisuuden arvioinnissa. Pidemmällä aikavälillä tämä ei ole valmistajien ku-

ten ei myöskään tarkastuslaitosten etu ja aiheuttaa varmaan ylimääräistä työtä, kun ohjelmistoa pyritään kehittämään useamman eri markkina-alueen vaatimukset täyttäväksi.

Ongelma on varmaankin kaksitahoinen: Ohjelmistoja tarkastavat tahot eivät välttämättä ymmärrä käytännön problematiikkaa eivätkä tunne riittävästi erilaisia ohjelmistoteknologioita mutta asettavat kuitenkin vaatimuksia ja tulkitsevat niitä omalla vakiintuneella tavallaan. Valmistajat eivät välttämättä taas ymmärrä aina vaatimusten taustoja, eivätkä käytössä olevat suunnitteluprosessit sinänsä tue näiden vaatimusten täyttymistä. Ongelma putkahtaa aina esiin, kun uutta tuotetta saatetaan markkinoille.

Tarvitaan laajempaa yhteistyötä, jossa standardointielimissä olisi edustajia niin viranomaispuolelta kuin myös ohjelmistoja valmistavista ja kehittävistä yrityksistä. Näin laaditut standardit soveltuisivat uusien ohjelmistoteknologioiden arviointiin. Lisäksi laadituista standardeista tulisi laatia valmistajien ja viranomaisten kanssa yhteistyössä ns. soveltamisohjeet siitä, kuinka valmistajat soveltavat vaatimuksia suunnittelussa ja kuinka viranomaiset tarkastavat ohjelmistoja näiden standardien vaatimusten pohjalta.

Toisin sanoen valmistajat ja valvovat viranomaiset loisivat yhteisiä pelisääntöjä ohjelmiston vaatimustenmukaisuuden osoittamiseksi ja tarkastamiseksi.

Menettelytavalla saataisiin varmasti lisättyä vuoropuhelua eri osapuolten välillä, mikä näkyisi myös molemminpuolisena toiminnan kehittymisenä.

6.4 Ohjelmistotuotannon työkaluohjelmistot

Ohjelmistotuotantoprosessin eri vaiheille on kehitetty useita erilaisia työkaluohjelmistoja, joiden tarkoituksena on hallita paremmin joko ko. vaiheen aikaisia toimintoja tai koko ohjelmistosuunnitteluprojektia, mukaan lukien useampien eri vaiheiden tehtävät. Työkalut avustavat esimerkiksi projektinhallinnassa, vaatimusten- ja muutostenhallinnassa, lähdekoodin tarkastuksessa sekä testauksessa.

Työkalut tuovat uusia mahdollisuuksia ohjelmistokehitysprosessiin ja voivat parantaa dokumentointia, lisätä jäljitettävyyttä sekä tuoda yhdenmukaisen toimintatavan eri suunnittelijoiden välille, jolloin suunnittelun lopputulos ja aikataulu ovat paremmin ennustettavissa.

Työkalujen käyttöönotto edellyttää yritykseltä henkilöstön koulutusta ja tietynlaista kulttuurimuutosta, jossa henkilöstö siirtyy vanhoista totutuista menetelmistä käyttämään uutta yhteistä suunnittelutyökalua. Tämä saattaa aiheuttaa alkuvaiheessa muutosvastarintaa, mikäli työkalun käyttöönottoa ei ole koulutettu riittävästi.

Toinen merkittävä tekijä työkalujen käyttöönotossa on niiden räätälöitävyys, joka mahdollistaa yrityskohtaisten tarpeiden integroimisen suoraan työkaluihin. Tämä edellyttää perinpohjaista tutustumista tuotteeseen sekä samalla oman suunnitteluprosessin täydellistä tuntemusta tarvittavine ongelmakohtineen ja parannusehdotuksineen. Tällainen mahdollisuus tuo varmasti pidemmällä aikavälillä kustannussäästöä ja parantaa suunnittelun laatua.

Työkaluohjelmistojen käytössä tulisi pyrkiä tavoitteeseen, jossa suunnitteluratkaisuihin ja riskienhallintaan liittyvä päätöksenteko jää suunnittelutiimin harteille ja työkalut tukevat vaatimustenhallintaa, toteuttamista ja dokumentointia.

Tämä on haasteellinen tehtävä ja oman hankaluutensa tuo lisäksi se, että yksi työkalu ei hallitse koko suunnittelun elinkaarimallia. Työkalujen hankinnassa tulisikin aina tutkia työkalujen yhteensopivuus tai käyttää saman tuoteperheen työkaluja.

Kukaan ei voi täydellisesti kiistää ohjelmistotuotannon tueksi tarkoitettujen työkalujen etuja. Täytyy kuitenkin muistaa, että mikään työkaluohjelmisto ei itsekseen rupea tuottamaan valmista koodia. Ohjelmistoja täytyy kokeilla, vertailla ja niiden käyttöön pitää kouluttaa riittävä määrä henkilöstöä. Lisäksi työkaluohjelmistot edellyttävät väistämättä jonkinasteisia prosessimuutoksia ja tätä kautta uusia ohjeistuksia ja käytäntöjä suunnitteluprosessiin.

Työkalujen käyttöönotto on kuitenkin perusteltua, koska ”automatisoimalla dokumentointia” jää suunnittelijoille enemmän aikaa keskittyä itse suunnitteluun. Automatisoitu dokumentaatio on myös tasalaatuisempaa kuin eri suunnittelijoiden eri aikoina tuottama dokumentaatio.

7. Loppusanat

Tässä julkaisussa on kuvattu parannusehdotuksia lääkintälaitteiden ohjelmistosuunnittelulle.

Esitetyt parannuskeinot eivät varmaankaan sovellu kaikkiin suunnittelun aikaisiin ongelmiin, mutta toivottavasti julkaisu osaltaan onnistuu avaamaan lääkintälaitteiden ohjelmistosuunnitteluun liittyviä ongelmia.

Laatujärjestelmien eräs tärkeistä periaatteista on jatkuva laadun parantaminen. Periaate soveltuu ohjelmistotuotantoprosessin kehittämiseen erittäin hyvin. Vaikka prosessi toimii tänään, saattaa se jo huomenna kirskahdella huolestuttavasti.

Suorituskykyinen ja luotettava ohjelmisto saavutetaan tavoittelemalla jatkuvasti yhteistyötä ja avoimuutta.

Lähdeluettelo

Haikala, I. & Märijärvi, J. 1998. Ohjelmistotuotanto. Helsinki: Suomen ATK-kustannus Oy.

IEC 60601-1-4. EN 60601-1-4:1996. Medical electrical equipment – Part 1-4: General requirements for safety – Collateral standard: Programmable electrical medical systems (IEC 60601-1-4:1996) + Amendment A1:1999 to EN 60601-1-4:1996.

IEC 60601-1-6. IEC 60601-1-6 Ed. 1.0 b:2004. Medical electrical equipment – Part 1-6: General requirements for safety – Collateral standard: Usability.

IEEE-830. IEEE 830:1998. IEEE Recommended Practice for Software Requirements Specifications

ISO 9241. ISO 9241-11:1998. Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability.

ISO 13485. EN ISO 13485:2000. Quality systems – Medical devices – Particular requirements for the application of EN ISO 9001:1994 (revision of EN 46001:1996).

ISO 14971. EN ISO 14971:2000. Application of risk management to medical devices (ISO 14971:2000).

ISO/IEC 12207. ISO/IEC 12207:1995. Information technology – Software life cycle processes + Amendment 1:2002 + Amendment 2:2004.

IVDD. Direktiivi 98/79/EY. In vitro -diagnostiikkaan tarkoitetut lääkinnälliset laitteet. Euroopan parlamentin ja neuvoston direktiivi 98/79/EY, annettu 27. päivänä lokakuuta 1998, in vitro -diagnostiikkaan tarkoitetuista lääkinnällisistä laitteista. (OJ L 331, 1998-12-07, sivut 1–37). http://www.sfs.fi/julkaisut/newapproach/invitro.html.

Koskimies, K. & Mikkonen, T. 2005. Ohjelmistoarkkitehtuurit. Helsinki: Talentum Media Oy. ISBN: 952-14-0862-6.

Laki 1505. Laki terveydenhuollon laitteista ja tarvikkeista 1505/94 ja asetus terveydenhuollon laitteista ja tarvikkeista 1506/1994.

MDD. Direktiivi 93/42/ETY. Terveydenhuollon laitteet ja tarvikkeet. Neuvoston direktiivi 93/42/ETY, annettu 14. päivänä kesäkuuta 1993, lääkinnällisistä laitteista, (OJ L 169, 1993-07-12, sivut 1–43). http://www.sfs.fi/julkaisut/newapproach/terve.html.

Pöyhönen, I. & Hukki, K. 2004. Riskitietoisen ohjelmiston vaatimusmäärittelyprosessin kehittäminen. VTT Tiedotteita 2263. Espoo: VTT. 36 s. + liitt. 9 s. ISBN 951-38-6496-0; 951-38-6497-9. http://www.vtt.fi/inf/pdf/tiedotteet/2004/T2263.pdf.

Pöyhönen, I., Kylmälä, K., Harju, H., Kemppainen-Kajola, P., Kuhakoski, K., Spankie, G. & Ventä, O. 2002. Vaatimukset ohjelmistoa sisältäville lääkintälaitteille. Hallinta ja menetelmät vaatimustenmukaisuuden osoittamiseksi. VTT Tiedotteita 2150. Espoo: VTT. 135 s. + liitt. 40 s. ISBN 951-38-6060-4; 951-38-6061-2. http://www.vtt.fi/inf/pdf/tiedotteet/2002/T2150.pdf.

XP:n avainkäytännöt. http://www.xprogramming.com/what_is_xp.htm.

Liite A: Kuvaus teknisen tiedoston sisällöstä

Julkaisussa puhutaan ohjelmiston suunnitteludokumentaatiosta, joka on osa tuotteen teknistä tiedostoa (DMR).

Terveydenhuollon laitteita ja tarvikkeita käsittelevä direktiivi (MDD) edellyttää valmistajaa ylläpitämään tuotetta koskevaa teknistä tiedostoa, jonka avulla valmistaja kykenee osoittamaan tuotteensa määräystenmukaisuuden valvontaviranomaisille. Tiedoston rakenne ja vastuut tietotuotannosta on tämän vuoksi valmistajan ratkaistava ja otettava käyttöön ennen tuotteen markkinointia. Määrättyjen osien teknisestä tiedostosta on oltava viranomaisten ja valmistajan valitseman ilmoitetun laitoksen käytettävissä.

Tekninen tiedosto on keino, jolla valmistaja voi varmentaa, että hänen tuotteensa ovat standardien ja direktiivien vaatimusten mukaisia, sekä kykenee osoittamaan tuotteensa vaatimustenmukaisuuden ilmoitetulle laitokselle (NB).

Teknisen tiedoston tulee sisältää kaikki tuotteeseen liittyvä suunnittelu-, valmistus-, käyttö-, suorituskyky- ja riskienhallintadokumentaatio.

Direktiivin (MDD) mukaan tekninen tiedosto sisältää erityisesti seuraavat seikat:

- tuotteen yleinen kuvaus, mukaan lukien suunnitellut vaihtoehdot
- suunnittelupiirustukset, suunnitellut valmistusmenetelmät sekä kaaviot esimerkiksi osista, osakokoonpanoista ja piireistä
- tarvittavat kuvaukset ja selitykset edellä mainittujen piirustusten ja kaavioiden sekä tuotteen toiminnan ymmärtämiseksi
- riskianalyysin tulokset sekä luettelo kaikilta osin tai osittain noudatetuista 5 artiklassa tarkoitetuista standardeista ja kuvaus tehdyistä ratkaisuista direktiivin olennaisten vaatimusten täyttämiseksi, jos 5 artiklassa tarkoitettuja standardeja ei noudateta kaikilta osin
- steriileinä markkinoille saatettavista tuotteista kuvaus käytetyistä sterilointimenetelmistä
- suunnittelulaskelmien ja suoritettujen tarkastusten tulokset; jos tuote on liitätettävissä toiseen (toisiin) laitteeseen (laitteisiin) toimiakseen käyttötarkoituksensa mukaisesti, näyttö siitä, että se täyttää olennaiset vaatimukset ollessaan liitettynä johonkin näistä laitteista, jolla on valmistajan ilmoittamat ominaisuudet
- testausselosteet ja tarvittaessa direktiivin (MDD) liitteessä X tarkoitetut kliiniset tiedot
- merkinnät ja käyttöohjeet.

Lisää tietoa teknisestä tiedostosta ja sen sisältövaatimuksista saa seuraavista lähteistä:

- GHTF:n www sivut, http://www.ghtf.org
- Lääkelaitoksen julkaisu ”Terveydenhuollon laadunhallinta, Lääkintälaitejärjestelmien turvallisuus”, http://www.nam.fi/uploads/julkaisut/Julkaisu_01_2004_04-07-06.pdf
- Terveydenhuollon laitteita ja tarvikkeita koskeva direktiivi (93/42/ETY:1993).

Julkaisija

Julkaisun sarja, numero ja raporttikoodi

VTT Tiedotteita 2320
VTT–TIED–2320

Tekijä(t)
Pöyhönen, Ilpo

Nimeke

Lääkintälaitteiden ohjelmistot
Suunnittelun kehityskohteita vesiputous- ja XP-mallin näkökulmasta

Tiivistelmä

Ohjelmistojen merkitys potilaan hoidossa kasvaa jatkuvasti. Ohjelmistot ohjaavat lääkintälaitteen toimintoja tai laskevat ja diagnosoivat mitattua potilasinformaatiota monimutkaisten algoritmien avulla. Tästä on seurannut se, että myös päätökset potilaan hoidosta tai hoitamatta jättämisestä perustuvat lisääntyvässä määrin ohjelmistojen laskemiin tuloksiin (laboratoriotulokset, hoitosuunnitelmat, diagnosointi ja tehohoito).

Ohjelmistovirheet ovat luonteeltaan systemaattisia, mikä voi aiheuttaa useiden potilaiden altistamisen haitalliselle tapahtumalle ennen kuin ohjelmistovirhe havaitaan. Ohjelmistojen turvallisuuden ja vaatimustenmukaisuuden arviointi on sen tähden avainasemassa. Ohjelmistotuotantoprosessi on lääkintälaitteiden ohjelmistokehityksen kulmakivi. Prosessin toiminta ja tuotokset ratkaisevat sen, täyttääkö ohjelmisto valmiissa tuotteessa sille asetetut viranomais- ja suorituskykyvaatimukset ja voidaanko suunnittelu tehdä kustannustehokkaasti ja toistettavasti.

Kehittämiskohteiden valinta ja yhdenmukaistettujen standardien soveltuvuuden arviointi omaan toimintaan edellyttää yritykseltä kriittistä ja perinpohjaista tarkastelua. Tässä julkaisussa esitetään keinoja, joiden avulla yrityksissä voidaan kehittää yrityskohtaista ohjelmistotuotantoprosessia.

Tarkastelun kohteena ovat standardien ja riskienhallinnan tarjoamaan tukeen perustuvat menettelytavat, ja kehittämiskohteet kuvataan perinteisen vesiputousmallin näkökulmasta. Lisäksi tarkastellaan lyhyesti myös XP-mallin soveltuvuutta lääkintälaitteiden ohjelmistojen suunnitteluun. Esitettyjen keinojen tarkoituksena on kehittää yrityksen suunnitteluprosessia siten, että suunnittelussa tunnistetaan tarvittavat lakisääteiset vaatimukset ja kukin suunnitteluvaihe tuottaa myös suunnitteludokumentaation, jota tuotteen hyväksyntäprosessi edellyttää.

Avainsanat
medical devices, software engineering, safety related software, cost-effectiveness, design, requirements, specifications, traceability, risk management, risk-informed life cycle models

ISBN
951–38–6766–8 (nid.)
951–38–6767–6 (URL: http://www.vtt.fi/publications/index.jsp)

Avainnimeke ja ISSN
VTT Tiedotteita – Research Notes
1235–0605 (nid.)
1455–0865 (URL: http://www.vtt.fi/publications/index.jsp)

Projektinumero
G3SU00058

Julkaisuaika	Kieli	Sivuja	Hinta
Helmikuu 2006	Suomi, engl. abstr.	61 s. + liitt. 2 s.	B

Projektin nimi
TRUST

Toimeksiantaja(t)
VTT

Yhteystiedot
VTT
PL 1300, 33101 TAMPERE
Puh. vaihde 020 722 111
Faksi 020 722 3365

Myynti
VTT
PL 1000, 02044 VTT
Puh. 020 722 4404
Faksi 020 722 4374

Published by

Series title, number and
report code of publication

VTT Research Notes 2320
VTT–TIED–2320

Author(s)
Pöyhönen, Ilpo

Title

Software of medical devices
Targets fot development of the design from the point of view of the waterfall and XP model

Abstract

The significance of the software in the patient's care increases continuously. The software controls the functions of the medical device or they calculate and diagnose measured patient information using complex algorithms. Due to this the decisions on the patient's care or leaving without care are increasingly based on the results calculated by the software (laboratory researchs, treatment plans, diagnosing and intensive care).

The software errors have a systematic character which can cause the exposure of several patients to a harmful event before the software errors are detected. Therefore the evaluation of the safety and compliance of requirements of software is in the key position. The software engineering process is the cornerstone of the medical device software development. The operation of the process and results determine the fact whether the software meets the requlatory and performance requirements in the final product and can a design be made cost-effectively and repeatably. Choice of targets for development and evaluation of the suitability of harmonized standards to own operation requires a critical and thorough examination of the company. In this report means are presented the means which the companies can use to develop a company-specific software engineering process.

The subject of the examination are procedures which are based on the support offered by the standards and the risk management and the target for development are described from the point of view of the traditional waterfall model. Also, the suitability of the XP model for the design of the software of medical devices is briefly examined. The purpose of presented methods is to develop the design process of the company so that in the design the necessary requlatory requirements are identified and each design phase produces the design documentation which is required during approval process of the product.

Keywords
medical devices, software engineering, safety related software, cost-effectiveness, design, requirements, specifications, traceability, risk management, risk-informed life cycle models

ISBN
951–38–6766–8 (soft back ed.)
951–38–6767–6 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN
VTT Tiedotteita – Research Notes
1235–0605 (soft back edition)
1455–0865 (URL: http://www.vtt.fi/publications/index.jsp)

Project number
G3SU00058

Date	Language	Pages	Price
February 2006	Finnish, engl. abstr.	61 p. + app. 2 p.	B

Name of project
TRUST

Commissioned by
VTT Technical Research Centre of Finland

Contact
VTT Technical Research Centre of Finland
P.O. Box 1300, FI-33101 TAMPERE, Finland
Phone internat. +358 20 722 111
Fax +358 20 722 3365

Sold by
VTT
P.O.Box 1000, FI-02044 VTT, Finland
Phone internat. +358 20 722 4404
Fax +358 20 722 4374

Lääkintälaitteiden ohjelmistokehitysprosessi on avainasemassa, kun arvioidaan lääkintälaitteiden ohjelmistojen suorituskykyä, laatua ja turvallisuutta. Julkaisussa kuvatussa hankkeessa tutkittiin keinoja kehittää lääkintälaitteiden ohjelmistokehitysprosessia kustannustehokkaammaksi siten, että suunnitteluprosessia kehittämällä saataisiin parannettua myös lopputuotteen eli ohjelmiston laatua ja turvallisuutta. Lääkintälaitteiden ohjelmistojen kehityskohteita pohditaan julkaisussa perinteisen vesiputousmallin ja XP-mallin näkökulmasta.

VTT TIEDOTTEITA 2320

Lääkintälaitteiden ohjelmistot. Suunnittelun kehityskohteita vesiputous- ja XP-mallin näkökulmasta

VTT
PL 1000
02044 VTT
Puh. 020 722 4404
Faksi 020 722 4374

VTT
PB 1000
02044 VTT
Tel. 020 722 4404
Fax 020 722 4374

VTT
P.O. Box 1000
FI-02044 VTT, Finland
Phone internat. + 358 20 722 4404
Fax + 358 20 722 4374

ISBN 951-38-6766-8 (nid.)
ISSN 1235-0605 (nid.)

ISBN 951-38-6767-6 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1455-0865 (URL: http://www.vtt.fi/inf/pdf/)