Matti-Juhani Savolainen

Huoltokirjanpito-ohjelmiston vaatimusmäärittely ja soveltuvien menetelmien valinta

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

31.3.2015

Metropolia

Tekijä Otsikko Sivumäärä Aika	Matti-Juhani Savolainen Huoltokirjanpito-ohjelmiston vaatimusmäärittely ja soveltuvien menetelmien valinta 34 sivua 31.3.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelmat	tieto- ja viestintätekniikka
Suuntautumisvaihtoehto	ohjelmistotuotanto
Ohjaaja	yliopettaja Kari Aaltonen

Insinöörityössä oli tavoitteena selvittää vesiputousmallin ja nykyaikaisten ketterien kehitysmallien eroa ohjelmistojen vaatimusmäärittelyn osalta.

Vaatimusmäärittelyn kohteena projektissa oli Keski-Uudenmaan pelastuslaitokselle määriteltävä paineilmalaitteiden huoltokirjanpito-ohjelmisto. Insinöörityössä kartoitettiin kummankin kehitysmallin etuja ja heikkouksia sekä suoritettiin vertailu vaatimusmäärittelymallien soveltuvuudesta kyseiseen projektiin.

Työn tavoitteena oli luoda vaatimusmäärittely uudelle järjestelmälle, johon kirjattaisiin pelastuslaitoksen käytössä olevat paineilmalaitteet.

Vaatimusmäärittelyn toteuttaminen edellytti perehtymistä itse vaatimusmäärittelyprosessiin ja eri tapoihin toteuttaa sitä. Uuden järjestelmän määrittely edellytti nykyisten työskentelytapojen kartoittamista, unohtamatta nykyistä tietojärjestelmää. Tiedon keräämiseen käytettiin henkilöhaastatteluita ja käyttäjiltä kerättyjä dokumentteja, joiden avulla selvitettiin ohjelman käyttäjät sekä käyttäjien asettamat vaatimukset uudelle järjestelmälle.

Insinöörityön lopputuloksena projektissa päädyttiin käyttämään eri kehitysmallien vaatimusmäärittelytapojen yhdistelmää, joka on ketteriä malleja kattavampi, mutta ei kuitenkaan yhtä raskas toteuttaa kuin perinteinen vesiputousmallin vaatimusmäärittely. Vaatimusmäärittelyä päädyttiin täydentämään käyttöliittymäkuvin, jotka havainnollistavat itse vaatimusmäärittelyä tulevan ohjelmiston käyttäjille sekä ohjelmiston mahdolliselle toteuttajalle.

Tehty vaatimusmäärittely todettiin käyttäjien toimesta riittäväksi ja ominaisuuksiltaan sellaiseksi, että vaatimukset täyttävä uusi järjestelmä poistaisi nykyjärjestelmän ongelmat.

Avainsanat	ohjelmistojen vaatimusmäärittely, vesiputousmalli, ketterät kehitysmallit, agile, paineilmalaite

Author	Matti-Juhani Savolainen
Title	Requirements specification for maintenance record software
Number of Pages	34 pages
Date	31 March 2015
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor	Kari Aaltonen, Principal Lecturer

The goal of this thesis was to study the differences between waterfall and agile development models in the field of requirements specification. The target system for requirements specification was the maintenance record software of the compressed air equipment of the Keski-Uusimaa Fire and Rescue Department. The thesis studied the advantages and weaknesses of both models, including a comparison and a discussion on the suitability of these models for the project. The aim was to create a requirements specification document for the new system, which would record the currently available compressed air systems of the fire department.

The requirements specification document required studying the requirement definition process and the various ways to implement it carefully. The new upcoming system required defining and identifying the current working methods and current maintenance record system. The required data was collected by personal interviews and by documents that were gathered from users. The collected data made it possible to detect the stakeholders of the software and their requirements for the new system.

As a result of the project, it was decided that a combination of different development models would be used to generate the requirements specification document. Research showed that the document would be more comprehensive if a combination of the waterfall and agile methods were used than if only the agile methods were used. In addition, it seems that implementation is less time-consuming and easier than if only the traditional waterfall model is used.

It was decided that the requirements specification would be improved with interface pictures that illustrate the operation of the software to the users and potential partners, who could build the new system. The finished requirement specification was found to be sufficient, because software that meets the requirements of the new system would eliminate the problems in current system.

Keywords	requirements specification, waterfall model, agile development, compressed air devices

Sisällys

Lyhenteet

CSV	Comma-separated values. Tiedostomuoto, jolla tallennetaan taulukko-muotoista tietoa tekstitiedostoon, jossa eri kentät on eroteltu toisistaan pilkulla.
IEEE	Insitute of Electrical and Electronic Engineers. Kansainvälinen tekniikan alan järjestö.
QR-koodi	Quick response code. Ruutukoodi eli kaksiulotteinen kuviokoodi
RFID	Radio frequency identification. Radiotaajuinen etätunnistus.
SCRUM	Yleinen tapa toteuttaa ketterää ohjelmistokehitystä.

1 Johdanto

Työn tarkoituksena on tuottaa vaatimusmäärittely paineilmalaitteiden (esim. savusukelluksessa käytettävät happipullot, maskit, paineenalentimet ja hengitysventtiilit) huoltokirjanpito-ohjelmistoa varten. Työssä vertaillaan perinteisen vesiputousmallin ja uudemman, ketterän kehitysmallin soveltuvuutta em. ohjelmiston vaatimusmäärittelyn tuottamiseen. Työn tutkimusosa toteutetaan case- eli tapaustutkimuksena. Paineilmalaiteiden huoltoon ja kirjanpitoon sisältyy myös jonkin verran lain asettamia vaatimuksia, jotka tullaan vaatimusmäärittelyn yhteydessä ottamaan huomioon.

Vaatimusmäärittelyn tavoitteena on olla lopulta asiakirja, jonka pohjalta voidaan tulevaisuudessa toteuttaa määrittelyt täyttävä ohjelmisto. Ohjelmisto tulisi valmistuttuaan korvaamaan taulukkolaskentaohjelmistoilla ja pahvikorteilla ylläpidetyt järjestelmät. Ilman kelvollista vaatimusmäärittelyä asiat toteutetaan usein niitä sen syvällisemmin pohtimatta, jolloin ohjelmiston lopputulos ei ole paras mahdollinen, ainakaan käyttäjän kannalta. Ikävimmässä tapauksessa voikin käydä niin, että käyttäjä joutuu muuttamaan työskentelytapojaan ohjelmiston takia. Kuvattu lopputulos ei ole toivottua, sillä toimivan ohjelmiston tulisi tukea käyttäjän työprosessia, eikä muuttaa sitä sen takia, että ohjelmisto ei mukaudu käyttäjän työskentelytapoihin. Itse vaatimusmäärittelyn tekemisprosessi on yleensä yksilöllinen ja tapauskohtainen, joskin yhteneväisiä menetelmiä on aina löydettävissä.

2 Tutkimuksen tausta

Tässä kappaleessa määritetään ja rajataan työn tutkimusongelma sekä kerrotaan tutkimuksen toteutuksen suunnitelluista työvaiheista.

2.1 Tutkimusongelma

Keski-Uudenmaan pelastuslaitoksella huolletaan paineilmalaitteita keskitetysti yhdellä paloasemalla. Huoltojen lisäksi paineilmapulloja täytetään useammalla asemalla. Tiukentuvat painelaitteiden säädökset ja direktiivit edellyttävät muun muassa laitteilta parempaa kirjanpitoa kuin mitä on aiemmin totuttu tekemään. Nykyisellään laitteista on

pidetty kirjaa taulukkolaskentaohjelmistolla, jonka toiminnassa on havaittu puutteita, eikä nykyinen järjestely täytä käyttäjien toiveita toimivasta ratkaisusta. Säädettyjen säädöksien ja direktiivien mukainen toiminta edellyttää järjestelmän välitöntä jatkokehittämistä, jotta vaatimukset saadaan täytettyä. Pahimmassa tapauksessa säädösten vastainen toiminta saattaa johtaa siihen, ettei painelaitteita saada huoltaa, puhumattakaan paineilmapullojen täyttämisestä lukuisilla eri paloasemilla.

Tutkimusongelmaan liittyy olennaisena osana ongelmakenttä ja sen kartoittaminen. Ongelmakentällä tarkoitetaan ympäristöä, jossa ongelma esiintyy. Ratkaistava ongelma on osa suurta ongelmakenttää. Kun lopputyön aiheena on ohjelmiston vaatimusmäärittely, työn tavoitteena on mahdollistaa ongelmakentässä toimivan laitteen määrittely, joka ratkaisee havaitun ongelman. Jotta ongelmakenttä voidaan ymmärtää, tulee ymmärtää nykyisin käytössä olevan järjestelmän tavoitteet, toimintaa säätelevät lait, rajoitteet ja heikkoudet.

Tämän insinöörityön tutkimusongelmana on se, miten nykyiset työskentelytavat paineilmalaitteiden parissa muutetaan toimintaa tukevan ohjelmiston vaatimusmäärittelyksi. Ongelma voidaan pilkkoa pienemmäksi seuraavien tukikysymysten avulla, joihin etsitään ratkaisu.

1. Mitkä ovat painelaitteiden huoltokirjanpito-ohjelmiston vaatimukset?

    I. Mitkä ovat käyttäjävaatimukset?

    II. Mitä vaatimuksia laki asettaa?

2. Minkälainen vaatimusmäärittelyn prosessi on sopiva huoltokirjanpito-ohjelmiston määrittelylle?

3. Mitkä ovat huoltokirjanpito-ohjelmiston sidosryhmät?

4. Mitä etuja saavutetaan uudella järjestelmällä verrattuna vanhaan?

5. Mitkä ovat nykyisen järjestelmän ongelmakohdat?

2.2 Tutkimuksen toteutus

Tutkimus toteutetaan case- eli tapaustutkimuksena. Tutkimuksen kohteena on Keski-Uudenmaan pelastuslaitoksen paineilmahuolto sekä palomiehet, jotka suorittavat paineilmapullojen täyttötoimenpiteitä. Tavoitteena on selvittää se, mitä paineilmahuolto kokonaisuudessaan tekee ja saada selvitetyn tiedon pohjalta aikaan tarkka kuvaus paineilmalaitteisiin liittyvistä toimenpiteistä sekä talteen kirjattavista tiedoista. Toimenpiteisiin ja kirjattaviin tietoihin vaikuttavat painelaitteille asetetut säädökset ja direktiivit, joihin täytyy myös perehtyä.

Vaatimusmäärittelyn edellytyksenä on sidosryhmiltä, eli käyttäjiltä, saadut vaatimukset uudelle järjestelmälle. Parhaan vaatimusmäärittelytavan valitseminen edellyttää erilaisiin malleihin tutustumista, jonka jälkeen malleja voidaan vertailla ja valita niistä kyseessä olevaan tapaukseen paras ratkaisu. Koska vaatimusmäärittelyn toteuttaminen edellyttää myös vaatimusmäärittelyn työvaiheiden ymmärtämistä, tulee osana tutkimusta käydä vaatimusmäärittelynkin työvaiheet läpi. Ohjelmistojen vaatimusmäärittelystä on saatavilla lukuisia kirjoja ja verkkosivustoja, joiden avulla suoritetaan tutustuminen vaatimusmäärittelyn vaiheisiin.

Ongelmakentän kartoitus suoritetaan tutustumalla nykyisin käytössä olevaan järjestelmään. Tämän lisäksi haastatellaan huoltotoimenpiteitä tekeviä henkilöitä, paineilmahuollon laitehuoltajat mukaan lukien. Haastattelujen yhteydessä tulevan järjestelmän käyttäjiltä pyritään selvittämään yksiselitteiset vaatimukset uudelle järjestelmälle tutkimuksen tuloksena parhaaksi todetulla vaatimusmäärittelyn mallilla.

Lopuksi kirjatut sidosryhmien vaatimukset esitellään käyttäjille itselleen. Samalla pyritään selvittämään ovatko saadut vaatimukset sellaiset, että niiden avulla pystytään määrittämään käyttäjiä hyvin palveleva uusi järjestelmä. Vaatimusmäärittelyt esitellään myös ohjelmistoja toteuttavalle yritykselle, joka osaltaan arvioi, onko vaatimusmäärittely kelvollinen ja tarpeeksi yksiselitteinen. Saadun palautteen perusteella tarkastellaan, mitä tutkimuksen osalta olisi voinut tehdä paremmin vai oliko valittu määrittelytapa onnistunut.

Vaatimuksia arvioivaksi yritykseksi valikoitui Tampereella sijaitseva ohjelmistoyritys Ocllo Oy:n, johtuen siitä, että henkilökunta on tuttua ja kommunikointi on siten helpompaa kuin täysin vieraan yrityksen henkilöstön kanssa.

Toteutettava vaatimusmäärittely rajataan asiakkaan osuuteen. Vaatimusmäärittelyssä ei mennä syvälle tulevan järjestelmän teknisiin yksityiskohtiin vaan keskitytään siihen osaan, jonka asiakkaan, tässä tapauksessa pelastuslaitoksen, voi olettaa itse olevan kykenevä toteuttamaan, asiakkaan olematta kuitenkaan itse ohjelmistoyritys. Tutkimus ei kata tulevan järjestelmän toteutusta vaatimusmäärittelyn osalta, vaan päättyy aikaansaatuun vaatimusmäärittelyyn.

3 Vaatimusmäärittelyprosessi ja mallit

3.1 Vaatimusmäärittelyn tavoite

Vaatimusmäärittely on yksi ohjelmistojärjestelmienkin kehitystyön perustehtävistä (1, s. 2). Vaatimusmäärittelyn tähtäimenä on selvittää se, mitä laitteen, tai tämän työn tapauksessa, ohjelmiston, tulee tehdä. Lisäksi vaatimusmäärittelyn tavoitteena on kuvata se millaisia toimintoja laitteiden, ihmisten ja ohjelmiston yhteistyö edellyttää (2, s. 94). Vaatimusmäärittely kuvaa sen, millainen ohjelmisto on, ei sitä, kuinka se toteutetaan (3, s.10). Onnistunut vaatimusmäärittely kertoo ohjelmiston kehittäjille ohjelmiston reunaehdot, jättäen kuitenkin tilaa luovuudelle ja suunnittelulle. Vaatimusmäärittely tuleekin kirjoittaa kyseessä olevalla kohteelle sopivalla tarkkuudella ja abstraktiotasolla, jotta määrittely ei ole liian pitkä ja täydellinen (1, s. 166). Vastaavasti epäonnistunut tai puutteellinen vaatimusmäärittely saattaa aiheuttaa sen, että määrittelyjen pohjalta luotu ohjelmisto toimii puutteellisesti, eikä ratkaise sitä ongelmaa, jonka ratkaisuksi ohjelmistoa on alkujaan alettu kehittämään. Vaatimusmäärittelyyn kannattaakin varata tarpeeksi resursseja, aika mukaan lukien, sillä sitä voidaan verrata esimerkiksi rakennuksen perustuksiin. Se on tuki, jonka päälle rakennuksen muut osat rakennetaan. Onnistuneen vaatimusmäärittelyn tulee olla yksiselitteinen, jotta sitä ei voida ymmärtää väärin.

Vaatimusmäärittely kertoo sen, mitä yksittäisen ongelman ratkaisemiseksi täytyy tehdä. Ratkaistava ongelma on osa suurempaa ongelmakenttää. Ohjelmistoprojektin tavoitteena on tehdä ongelmakentässä toimiva laite (eli ohjelma), joka ratkaisee kyseessä olevan ongelman (1, s. 8). Siksi onkin oleellista huomioida ongelmakenttään vaikuttavat tekijät, kuten muun muassa ihmiset, fyysiset laitteet, toimintaympäristöt ja lait. Usein ongelmakenttä ei kuitenkaan ole stabiili, vaan ratkaisun vaatimukset muuttuvat ajan myötä (1, s. 13). Siksi tarvitaan edellä mainittua evoluution hallintaa. Ongelmakentän muutoksia voivat aiheuttaa esimerkiksi organisaatiomuutokset ja päivittyvä tek-

niikka. Voidaankin sanoa, että vaatimusmäärittelyä tehdessä täytyy ottaa huomioon sidosryhmiltä (esimerkiksi ohjelmisto käyttäjät) tulevien vaatimusten lisäksi ohjelman ja ihmisten toimintaympäristö ja siihen vaikuttavat tekijät.

Ohjelmistojen vaatimusmäärittelyjen tekemiseen on olemassa useita standardeja, joista yhden takana on Electrical and Electronics Engineers (IEEE). IEEE:n standardi jonka alkuperäinen versio julkaistiin alkujaan jo vuonna 1984, sisältää ohjeet siihen, kuinka luodaan yksiselitteisiä ja täydellisiä vaatimusmäärittelyitä. (3, s. 95.)

3.2 Vaatimukset

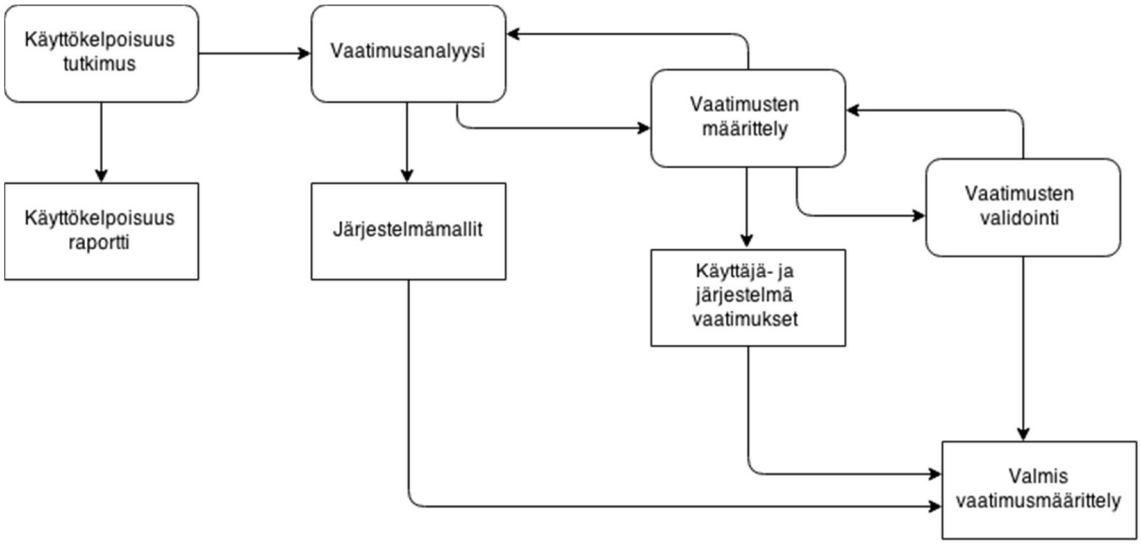
Vaatimukset voidaan jakaa toiminnallisiin (functional requirements) ja ei-toiminnallisiin vaatimuksiin (non-functional requirements). Toiminnalliset vaatimukset kertovat sen, mitä toimintaa järjestelmä suorittaa. Ne kuvaavat sitä, miten ja mitä käyttäjä järjestelmässä tekee ja miten järjestelmä siihen suhtautuu. Ei-toiminnalliset vaatimukset kuvaavat sen, miten toiminnalliset vaatimukset toteutetaan. Esimerkiksi laatuun ja arkkitehtuurin liittyvät vaatimukset ovat ei-toiminnallisia vaatimuksia. Julkisen hallinnon tietohallinnon neuvottelukunta esittää toiminnallisuuksien jakamista kolmeen eri ryhmään: toimintalähtöisiin vaatimuksiin (business requirements), käyttäjävaatimuksiin ja järjestelmän toiminnallisiin sekä ei-toiminnallisiin vaatimuksiin. (4, s. 8.)

International Institute of Business Analysis suosittaa A guide to the Business Analysis Body of Knowledge-teoksessaan, että vaatimukset jaettaisiin kuuteen ryhmään: arkkitehtuurisiin vaatimuksiin, toimintalähtöisiin vaatimuksiin, sidosryhmien vaatimuksiin, toiminnallisiin vaatimuksiin, laatuvaatimuksiin sekä toteutusvaatimuksiin. (5.)

3.3 Vaatimusmäärittelyn prosessi

Vaatimusmäärittelyn osavaiheita ovat vaatimusten kartoitus, spesifiointi, arviointi, riskienhallinta, laadunvarmistus sekä evoluution hallinta. Näistä vähiten käytettyjä ovat riskienhallinta, laadunvarmistus ja evoluution hallinta. (1, s. 167.) Klassinen vaatimusmäärittelyprosessi, joka on esitetty kuvassa 1, jakaa vaatimusmäärittelyn neljään vaiheeseen, joiden tuloksena tuotetaan lopullinen vaatimusmäärittely. Uudemmat proses-

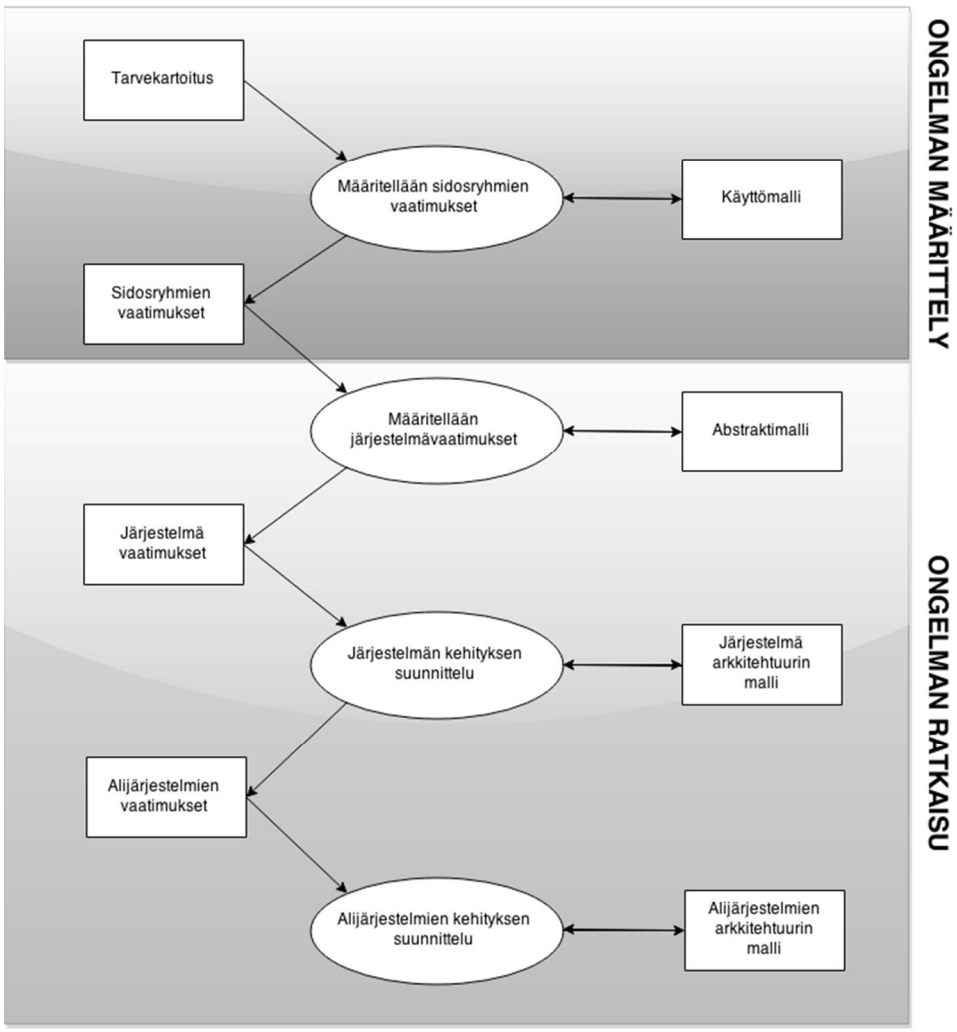
simallit sisältävät yleensä klassisen mallin neljä vaihetta, pilkkoen niitä pienimmiksi osasiksi.



Kuva 1. Klassinen vaatimusmäärittelyprosessi. (6).

Käyttökelpoisuustutkimuksessa tutkitaan, onko järjestelmän tuottaminen ylipäätään mahdollista ja järkevää. Tutkimuksen tuotoksena toimii käyttökelpoisuusraportti, jossa kerrotaan havaitut tulokset. Sen jälkeen siirrytään tekemään vaatimusanalyysiä, jonka avulla pyritään selvittämään, mitä vaatimukset ovat, ovatko ne selkeitä, ymmärrettäviä ja yksiselitteisiä. Kun vaatimukset ovat selvillä, ne kirjataan ylös vaatimusmäärittelyksi ja hyväksytään (vaatimusten validointi), taikka jos vaatimuksia ei sellaisinaan hyväksytä, niitä käsitellään, kunnes ne ovat hyväksyttäviä. Vaatimusanalyysin ja vaatimusten määrittelyn kautta saadaan tuotettua järjestelmämallit sekä käyttäjä- ja järjestelmävaatimukset, jotka yhdessä muodostavat valmiin vaatimusmäärittelyn.

Vaatimusmäärittelyä voidaan tehdä usein eri tavoin. Usein ohjelmiston kehitysmalli määrittää myös sen, millä tavoin vaatimusmäärittelyä tehdään. Kehitysmalleja ovat muun muassa perinteinen vesiputousmalli, erilaiset ketterät kehitysmallit ja prototyyppimenetelmä. (7, s. 95.) Tässä työssä käsitellään kahta ensimmäisenä mainittua, eli perinteistä vesiputousmallia ja ketteriä kehitysmalleja.



Kuva 2. Vaatimusmäärittelyn Requirements Engineering-teoksen mukaan. (8, s. 34)

Springer-kustantamon julkaiseman Requirements Engineer -kirjan mukaan koko vaatimusmäärittelyprosessi voidaan jakaa kahteen osaan (8, s. 34). Kuvassa 2 on esitetty kirjassa esitetty malli. Ongelman määrittelyvaihe sisältää sidosryhmien vaatimuksien ja käyttömallin selvityksen. Jäljelle jäävät vaiheet ovat ongelman ratkaisemista. Seuraavien alaotsikoiden alla on käyty läpi kuvassa 2 esitettyjä työvaiheita yleisellä tasolla.

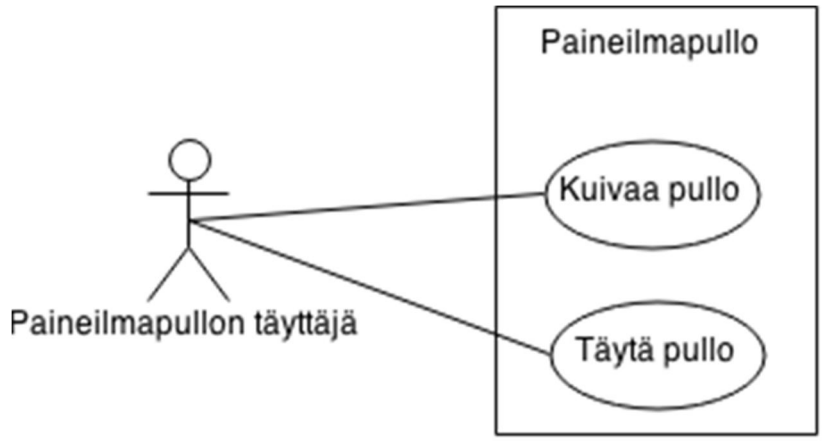
Työvaiheet sisältävät vaatimusten kartoituksen, spesifioinnin ja arvioinnin. On huomioitava, että kuvassa 2 osa työvaiheista on merkitty osaksi ongelman ratkaisua, jolla ei tarkoiteta kuitenkaan vielä ongelman ratkaisun toteuttamista. Ratkaisuvaiheessa luodaan määrittely ja suunnitelma, jonka pohjalta ongelman ratkaisu voidaan toteuttaa.

3.3.1 Tarvekartoitus ja sidosryhmien vaatimusten kartoitus

Tarvekartoitus on yleensä lyhyehkö tekstinpätkä, jolla asiakas kertoo tarvitsevansa ohjelmiston tekemään jotakin asiaa, jostakin syystä. Tämän pohjalta lähdetään selvittämään tarkempia vaatimuksia.

Yleisessä vaatimustenmäärityksprosessissa lähdetään aluksi selvittämään sidosryhmiltä sitä, mitä he haluavat järjestelmää käyttämällä saavuttaa, olettaen, että sidosryhmät ovat selvillä. Tässä vaiheessa vältetään vielä minkäänlaisen ratkaisun tarjoamista ongelmaan. Kartoitusvaiheessa kirjataan vain sidosryhmien toiveet ja vaatimukset. (8, s. 16.)

Hyväksi havaittu tapa esittää kysymys tulevan järjestelmän käyttäjille, on unohtaa koko järjestelmä ja kysyä sen sijaan käyttäjiltä, mitä he haluaisivat pystyä tekemään. Seuraava looginen kysymys olisi selvittää se, keneltä tai miltä tahoilta kysymys täytyy kysyä. Näihin kysymyksiin vastaamalla saadaan kartoitettua sidosryhmät, eli ihmiset ja tahot, joita tuleva ohjelmisto kiinnostaa suorasti taikka epäsuorasti. (8, s. 88.) On huomioitava, että sidosryhmienkin vaatimuksia saatetaan joutua rajoittamaan esimerkiksi taloudellisista syistä, jotta ohjelmiston kehityskustannukset pysyvät halutulla tasolla. Tässä vaiheessa on oleellista päästä asiakkaan kanssa yhteisymmärrykseen siitä, mitkä toiveet kirjataan sidosryhmien vaatimuksiksi.



Kuva 3. Esimerkki yksinkertaisesta käyttötapauskaaviosta.

Dokumentoinnin apuna voidaan käyttää esimerkiksi käyttötapauskaavioita, joissa pilkotaan yksittäinen toive halutusta lopputuloksesta pienemmiksi osiksi. Tyypillinen käyttötapauskaavio on esitetty kuvassa 3.

3.3.2 Järjestelmävaatimukset ja järjestelmäkuvauksen tekeminen

Järjestelmävaatimuksilla kerrotaan käsitteellisellä tasolla se, kuinka tuleva järjestelmä täyttää sidosryhmiltä saadut vaatimukset. Järjestelmävaatimuksissa ei tule ottaa kantaa siihen, kuinka järjestelmä suunnitellaan. Käytännössä tämä tarkoittaa sitä, että järjestelmävaatimukset tulisi kuvata yksittäisinä ”mustina laatikoina”, joiden tarkkoihin yksityiskohtiin ei oteta kantaa, kunhan ne vain suorittavat vaaditut sidosryhmien vaatimukset (8, s. 112). Järjestelmävaatimusten kartoitusta seuraa järjestelmän arkkitehtuurin kartoitus, jossa suunnitellaan se, miten kuvatut järjestelmävaatimukset ratkaistaan. (8, s. 16). Järjestelmävaatimuksien kuvaamiseen ja järjestelmäkuvauksen tekemiseen hyviä apukeinoja ovat muun muassa tila- ja prosessikaaviot yhdessä (8, s. 112.)

Järjestelmäkuvaus voidaan pilkkoa myös pienempiin osiin. Järjestelmäkuvauksen osia ovat sisäinen toiminnallisuus (internal functionality), ihmisvuorovaikutuksen toiminnallisuudet (human interaction functionality), käyttöliittymän toiminnallisuus (interface functionality), suojakeinojen toiminnallisuus (safeguard functionality) sekä järjestelmien välinen vuorovaikutus (systems transactions). Järjestelmäkuvauksen luomisprosessia seuraa hyväksymisprosessi, jonka perusteella järjestelmäkuvaus hyväksytään sellaisenaan tai siihen tehdään tarvittaessa muutoksia.

3.3.3 Alijärjestelmien määrittely ja arkkitehtuurimallin suunnittelu

Kun järjestelmäkuvaus on tehty, seuraava vaihe on tuottaa sen perusteella malli, jossa alijärjestelmät toteuttavat aiemmin luotuja määrityksiä. Tätä kutsutaan järjestelmän arkkitehtuurimallin suunnitteluksi. Oleellista on kuvata se, kuinka alijärjestelmät keskustelevat keskenään ja mitä toimintoja ne suorittavat. Kun alijärjestelmien toiminnallisuudet on kuvattu, voidaan niiden perusteella kirjoittaa jokaisen alijärjestelmän vaatimusmäärittely. Vaatimusten tulee kertoa, minkälainen toiminnallisuus alijärjestelmän tulee toimittaa, mitä liittymiä se käyttää taikka tarjoaa käyttöön. Vaatimusten tulee myös kuvata rajoitteet, joiden sisällä järjestelmän tulee toimia. Alijärjestelmän määrittely tehdään samalla tavalla kuin järjestelmävaatimuksienkin määrittely, erona on se, että yk-

sittäistä alijärjestelmää käsitellään täysin omana komponenttinaan, irrallaan muusta järjestelmästä. (8, s. 127.)

3.4 Riskienhallinta

Koska tämän opinnäytetyön tarkoitus ei ole perehdyttää lukijaansa kattavasti riskienhallintaan, tässä luvussa käydään hyvin pintapuolisesti läpi riskienhallintaa ohjelmistojen vaatimusmäärittelyprosessiin liittyen.

Suomen Riskienhallintayhdistys ry määrittelee, että riskienhallinnalla tarkoitetaan yleisesti toimintaa riskien ja niistä aiheutuvien vahinkojen vähentämiseksi (9). Riskienhallinta sisältää tilanteiden arviointia ja suunnittelua, unohtamatta käytännön tekoja.

Ohjelmistojen kehityksessä kohdataan usein odottamattomia ongelmia, jotka johtuvat esimerkiksi ympäristön muutoksista, aiheuttaen viiveitä taikka pahimmillaan estäen koko projektin onnistumisen. Riskienhallinta ohjelmistokehityksessä on haastavaa, mutta oikein toteutettuna se auttaa havaitsemaan ongelmat ennen niiden esiintymistä, parantaen näin projektin kokonaisvaltaista lopputulosta. (10, s.1.) Parhaimmillaan riskienhallinta säästää aikaa ja pienentää kustannuksia, vähentäen samalla ikävien yllätysten määrää.

Kuinka riskienhallinta sitten otetaan osaksi vaatimusmäärittelyä? Tähän on monia tapoja, joista yksi on GSRM (goal-driven software development risk management model), vapaasti suomennettuna tavoitteisiin perustuva ohjelmistojen riskienhallintamalli, jossa määritellään kehityksen kannalta oleelliset virstanpylväät, joihin täytyy päästä, jotta projekti onnistuu vaaditusti.

GSRM keskittyy projektin kannalta ei-teknisiin asioihin, kuten projektin suorittamisen estäviin rajoituksiin, sidosryhmiin, käyttäjiin, projektiin osallistuvien tahojen väliseen kommunikointiin. GSRM:n tarkastelun alla ovat myös teknisemmät asiat, kuten kehitysprosessi ja siinä käytettävät työkalut, joiden riskit pyritään selvittämään jo ennen vaatimusmäärittelyn aloittamista. Näin vähennetään virheellisten vaatimusten määrää sekä tarvetta palata jälkikäteen käsittelemään jo aiemmin luotuja vaatimuksia. (10, s.12.)

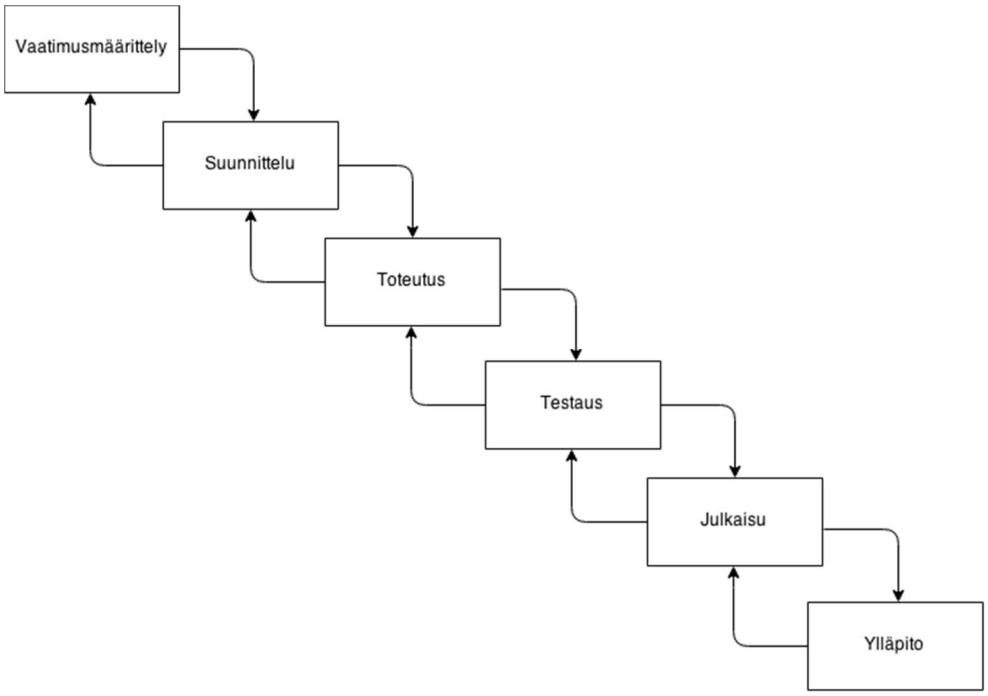
Riskienhallinta ei ole vain jollekin kehitysmallille tyypillinen ominaisuus, vaan se voidaan ottaa osaksi myös ketteriä kehitysmalleja, joista kerrotaan enemmän luvussa 3.6. Ketterässä kehityksessä ei useinkaan turvauduta raskaaseen dokumentointiin vaatimusten osalta, joten riskejä voidaan tutkia ja havaita esimerkiksi aikaisten prototyyppien avulla. Prototyypit ovatkin yksi parhaista tavoista saada käyttäjät mukaan määrittelemään järjestelmän ulosantia ja rajoitteita. (11.)

Ketterissä menetelmissä riskienhallintaa käsitellään yleensä useammin verrattuna perinteiseen vesiputousmallin käyttöön. Oikein toteutettuna ketterien menetelmien riskienhallinta on mukana vähintäänkin jokaisen sprintin (aikamääre, jonka aikana toteutetaan ajanjaksolle sovitut tehtävät) suunnittelutapaamisessa. Toki ketterissäkin menetelmissä tarvitaan silti jokin prosessi, jolla riskejä hallitaan. (12). Ville Ylimannelan tutkimuksessa yhdeksi tavaksi toteuttaa ketterän kehityksen riskienhallintaa ehdotetaan sprintin suunnittelun arviointivaihetta, jossa kartoitetaan tulevien ominaisuuksien riskitaso. Mikäli ominaisuudessa todetaan olevan riskejä, tarkemmalle riskianalyysille varataan erikseen aikaa ja selvitetään miten riskiä voidaan pienentää. Ylimannela painottaa myös suunnittelun ja tietoturvakoulutuksen tärkeyttä, jotta kaikki projektin osalliset, asiakasta ja tiimin johtajaa myöten, ovat tilanteen tasalla. (13, s. 5.)

Samaa Ylimannelan kuvaavaa metodia voidaan käyttää myös perinteisen vesiputousmallin vaatimusmäärittelyprosessissa. Kun vaatimuksia kirjataan ylös, tulisi niiden riskit kartoittaa ja laatia suunnitelma havaittujen riskien pienentämiseksi. Tärkeässä osassa niin ketteriä kehitysmalleja käyttävissä, kuin perinteisemmissäkin projekteissa ovat turvallisuusvaatimukset, joiden tulisi olla osa vaatimusmäärittelyä. Ne kertovat projektin kehittäjille, kuinka paljon aikaa riskienhallintaan tulee projektissa käyttää.

3.5 Vesiputousmalli

Vesiputousmalli on perinteinen ja samalla vanhin ohjelmistojen kehitystapa. Se edellyttää suunnittelua jo kehityksen varhaisessa vaiheessa. Raskas dokumentointi- ja suunnitteluvaihe varmistavat, että mahdolliset suunnitteluviat ilmenevät ennen toteutusvaihetta. Tämän takia vesiputousmalli soveltuukin hyvin projekteihin, joissa laadunhallinta on tärkeässä asemassa. (7, s. 95.) Malli on nimensä mukaisesti kuin vesiputous. Kuten kuvasta 4 on havaittavissa, malli etenee ylhäältä alas vaiheittain, seuraavasta työvaiheesta toiseen.



Kuva 4. Vesiputousmallin työvaiheet (7, s. 99).

Ensimmäisenä vaiheena on vaatimusmäärittely (requirement analysis) ja vesiputouksen lopussa on ylläpito (maintenance). Perinteinen vesiputousmalli perustuu siihen, että vaatimusmäärittely pyritään saamaan kerralla täydelliseksi. Tehdyn vaatimusmäärittelyn perusteella suunnitellaan järjestelmä (system design), jonka jälkeen suunnitelmaa aletaan toteuttamaan (implementation). Näitä työvaiheita seuraavat testaus (testing), julkaisu (deployment) ja ylläpitovaihe (maintenance). Vesiputousmallin eri vaiheet on havainnollistettu kuvassa 4.

Jos vesiputousmallia käytetään, se edellyttää, että vaatimukset ovat erittäin hyvin dokumentoituja, selkeitä ja yksiselitteisiä. Myös käytettävä teknologia täytyy ymmärtää hyvin. Vesiputousmallin käyttö edellyttää myös sitä, että projekti on kestoltaan lyhyt. (14).

Vesiputousmallin yksi ongelma on se, että jos projektin aika loppuu kesken, leikataan aikaa usein mallin loppupäästä eli testausvaiheesta, jolloin laatu usein kärsii. Ajanhallinta on myös haaste. Miten vesiputousmallia käyttäen voi tietää sen, kuinka paljon työtä on vielä ajallisesti jäljellä? Kun kehitys ja testaus suoritetaan projektin loppupäässä, miten käy jos testivaiheessa havaitaan kriittinen ongelma? Aikaa tarvitaan lisää, ellei ongelmiin ole osattu varautua. Ongelmia syntyy myös silloin, jos sidosryhmät toteavatkin vasta testausvaiheessa tietävänsä oikeat tarpeensa. Alkuperäisiä vaatimuk-

sia noudattavat ominaisuudet on ehditty toteuttaa kattavasti, joten niiden muuttaminen on työlästä.

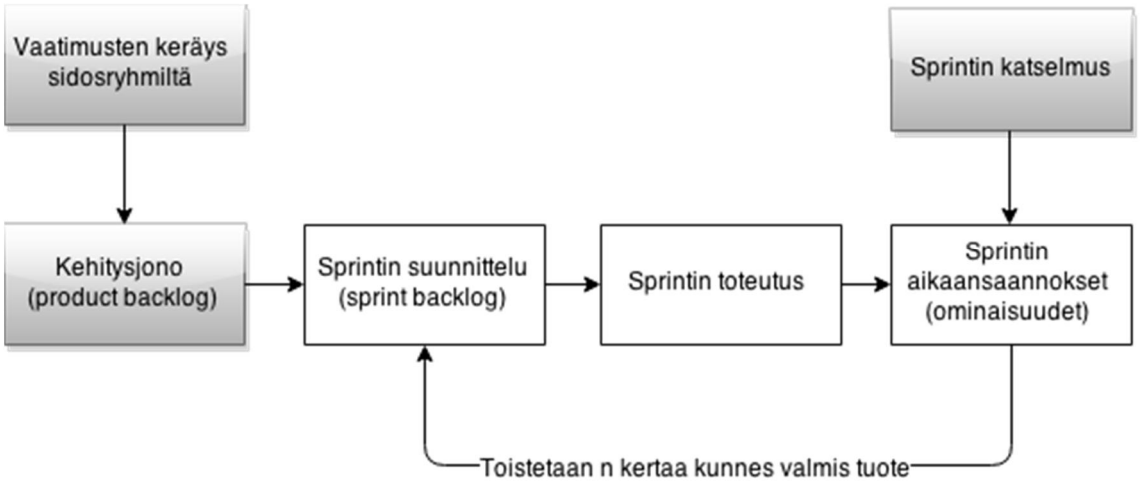
3.6 Ketterät mallit

Ketterä kehitysmalli ei ole niinkään sarja työkaluja, taikka selkeä malli, kuten vesiputousmalli, vaan ennemminkin filosofia. Ketterän kehityksen manifesti allekirjoitettiin vuonna 2001 seitsemäntoista henkilön toimesta. Tavoitteena oli kehittää älykkäämmät työskentelytavat ohjelmistoprojekteille. Manifestissa kirjoitetaan, että ohjelmistotyöskentelyn tekotapoja pitäisi jakaa ja muita auttaa niiden kehittämisessä. Myös yksilöitä ja kanssakäymistä tulisi arvostaa enemmän kuin menetelmiä taikka työkaluja. Samaten tulisi laittaa enemmän painoarvoa toimivalle ohjelmistolle kuin kattavalle dokumentoinnille. Manifestin mukaan myös asiakkaiden kanssa tulisi tehdä ennemminkin yhteistyötä, kuin käydä jäykkiä sopimusneuvotteluita. Ohjelmistojen kehittäjien tulisi myös pyrkiä vastaamaan muutoksiin, sen sijaan että pitäytyisivät tehdyssä suunnitelmassa. Manifestin lopussa todetaan, että kaikkia mainittuja asioita kuitenkin arvostetaan, mutta niitä joilla on suurempi painoarvo, tulisi arvostaa enemmän. (15.)

Ketterää filosofiaa toteuttavia kehitysmalleja ovat muun muassa Scrum, FDD (Feature Driven Development), Kanban, Lean ja XP (Extreme Programming). Edellä mainituista malleista käytetyin on Scrum ja siitä johdetut erilaiset kehitysmallit. (16, s. 2.). Koska erilaisista ketteristä kehitysmalleista on saatavilla valmiiksi jo valtavasti tietoa, ei tässä lopputyössä käydä tarkemmin läpi erilaisia kehitysmalleja, vaan käydään ketterää kehitystä läpi yleisellä tasolla.

Ketterässä kehitysmallissa ei ole kiveen hakattuja vaiheita vaan iteraatioita, joiden lopputuloksena on toimivaa ohjelmistokoodia, jota käytetään ratkaisemaan muuttuvia käyttäjävaatimuksia. Näitä iteraatioita toistetaan, kunnes saadaan valmis tuote. Ketterä kehitysmalli eroaakin siis ratkaisevasti vesiputousmallista siinä suhteessa, että vaatimukset ja niiden ratkaisut kehittyvät usein koko projektin elinajan. Sen sijaan, että yritettäisiin alkuun tehdä kattava dokumentti siitä, mitä ollaan tekemässä, ketterässä kehityksessä tukeudutaan esimerkiksi asiakkaan kirjoittamiin tarinakortteihin. Yksittäisen tarinakortin tehtävänä on määritellä kyseinen ominaisuus vastaamalla kysymyksiin kuka, mitä ja miksi mahdollisimman yksiselitteisesti, mutta kuitenkin tarpeeksi väljällä abstraktiotasolla. Käyttäjän kirjoittaman tarinan tulisi mahtua pienelle muistilapulle.

Usein ohjelmistokoodia käytetään dokumenttina ja muuta kirjallista dokumentointia vältetään, jos mahdollista. Käytännössä tämä tarkoittaa sitä, että dokumentointia pyritään tekemään juuri vain sen verran kuin on pakollista. Joissakin projekteissa dokumentointi on kuitenkin välttämätöntä esimerkiksi ohjelmiston liikearvon kannalta. Dokumentoinnin vähentäminen ei kuitenkaan ole täysin riskitöntä. Toisaalta perinteisen tekstidokumentoinnin vähentäminen antaa aikaa muille kehitystoiminnassa tarvittaville työtehtäville, mutta asioiden avaaminen niitä tarvitseville ihmisille ja tahoille puhumalla, saattaa vaatia kuitenkin lopulta enemmän aikaa. Dokumentoinnissa tulisikin löytää kyseiselle projektille sopiva keskitie, jolla dokumentointia suoritetaan tehokkuus maksimoiden. Riskiä vähennetään kertomalla ohjelmistosta ainoastaan tarpeellinen tieto. (17, s.183.)



Kuva 5. Yksinkertaistettu Scrum-mallin mukainen ketterä kehitysmalli (7, s. 6).

Kuvassa 5 on esitelty Scrum-mallissa käytetty työskentelytapa vaiheittain. Prosessi alkaa vaatimusten keräämisellä, joista koostetaan tuotteen kehitysjono (product backlog). Sprint on tyypillisesti 1–4 viikon mittainen ajanjakso, jonka aikana toteutetaan sprintille määritellyt ominaisuudet. Sprintin suunnitteluvaiheessa tuotteen kehitysjonosta poimitaan yksittäisiä ominaisuuksia sprintin toteutusvaiheessa tehtäväksi. Kun sprintille määritelty aika on kulunut, tarkastellaan sprintin aikaansaannokset (sprintin katselmus) ja aloitetaan seuraavan sprintin suunnittelu. Näin käynnistyy jälleen uusi sprintti, jonka aikana toteutetaan uudet ominaisuudet taikka korjataan edellisiltä sprinteiltä jääneitä puutteita. Sprinttejä toteutetaan kunnes ohjelmisto on vaaditulla tasolla.

Kuten edellä mainittiin, Scrum-mallin mukaisessa ja muissakin ketterissä malleissa on tyypillisesti käytössä taustaloki (backlog), joka sisältää listauksen kaikista järjestelmään

vaadituista ominaisuuksista. Vastaavasti sprintin taustaloki sisältää tiedot yksittäisen sprintin aikana toteuttavasti ominaisuuksista. Taustalokia ei kuitenkaan tule sekoittaa vaatimusmäärittelyyn. (17.) Taustalokin tarkoituksena on toimia listana tekemättömistä töistä, jota ohjelmiston kehittäjät käyttävät pysyäkseen kartalla tilanteesta.

Yksi ketterien kehitysmenetelmien erityispiirre on mukautunut kehitys. Kun projektia tehdessä huomataan, ettei aika tule riittämään kaikkien vaadittujen ominaisuuksien luomiseen, aletaan ominaisuuksia karsia, jos se vain on mahdollista. Jäljellä olevan ajan puitteissa toteutetaan silloin vain oleelliset kriittiset ominaisuudet.

3.7 Mallien vertailu

Vaatimusmäärittelyn osalta mallit eroavat toisistaan huomattavasti. Vesiputousmalli edellyttää kattavan vaatimusmäärittelyn tekemistä. Ketterissä menetelmissä ei ole suoranaista vaatimusmäärittelyä vaan ohjelmistoa muokataan kehityksen myötä. Yhtenäistä malleilla on vaatimusmäärittelyn osalta se, että kummassakin tapauksessa pohjatiedoiksi kelpaavat käyttötapauskaaviot.

Jos vaatimusmäärittely on huolella tehty, vesiputousmalli sallii ketteriä menetelmiä paremmin sen, ettei ohjelmiston sidosryhmiä tarvita enää vaatimusmäärittelyn jälkeen kehitystyön käynnistyttyä. Ketterät menetelmät saattavat usein edellyttää tarkistuksien tekemistä asiakkaalta, jos vaadittu ominaisuus on kuvattu vain muutamalla lauseella. Toisaalta usein saattaa käydä myös niin, ettei asiakas tiedä vaatimusmäärittelyvaiheessa täsmälleen oikeaa tarvettaan. Erheellisesti tehty vaatimusmäärittely johtaakin silloin vesiputousmallia käyttäessä pahimmillaan siihen, että loppukäyttäjä saa jotain aivan muuta kuin tarve vaatisi. Ketterien kehitysmallien voisi taas kuvitella vaativan asiakkailta aktiivisuutta ja oman toiminnan pohdintaa huomattavasti enemmän. Ketterilä malleja käyttäen saadaan sidosryhmiltä palautetta jo aikaisessa vaiheessa, jolloin suuntaa voidaan tarvittaessa korjata nopeastikin. Aikaisessa vaiheessa tehdyistä muutoksista ei silloin synny asiakkaille vastaavia kuluja kuin vesiputousmallin loppuvaiheissa pyydetyistä muutoksista.

Riskienhallinnan kannalta ketteriin menetelmiin liittyy toisenlaisia ongelmia kuin vesiputousmalliin. Kun sprintin tavoitteena on saada tehtyä määritetyt ominaisuudet, saattaa ongelmaksi muodostua tarvittavien henkilöresurssien saanti sprintin lopulla suoritetta-

vaan riskienhallintaan. Toisaalta voidaan myös pohtia sitä, kenellä on ketterää kehitysmallia käyttävässä projektissa valta sanella, mikä on merkittävä riski, joka vaatii asiakkaan hyväksynnän ja mikä taas on vähemmän merkittävä riski, joka voidaan jättää huomioimatta. (13, s. 2.) Käytettävällä kehitysmallilla ei kuitenkaan ole vaikutusta siihen, etteikö jotakin riskienhallintamallia voitaisi käyttää projektin aikana.

Ketterien mallien käyttö parantaa usein tuotteen laatua, koska testaus aloitetaan heti eikä vasta projektin lopulla, kuten vesiputousmallissa. Ajanhallinta on myös helpompaa ketteriä malleja käyttäen. Kun puolet ominaisuuksista on toteutettu, ollaan puolessa välissä projektia. Vesiputousmallissa on vaikeampi hahmottaa projektin puoliväliä.

VersionOne-yritys valmistaa ja myy ketteriin projekteihin käytettävää hallintatyökalua. Yritys on tehnyt kyselyn ketteriä menetelmiä käyttäville yrityksille vuonna 2012. 18 prosenttia yrityksistä vastasi, ettei yhdenkään ketterää kehitysmallia käyttäneen projektin voida katsoa epäonnistuneen. Yritykset, jotka olivat kokeneet epäonnistumisia, sanoivat sen johtuneen muun muassa yrityksen toimintamallista ja ulkoisesta painostuksesta palata käyttämään vesiputousmallia. (16, s. 2.) Tulosten perusteella voidaan myös todeta, että ketterän kehitysmallin käyttö saattaa aiheuttaa pelkoa esimerkiksi etukäteissuunnittelun, hallinnan ja dokumentoinnin puutteesta. (16, s. 11.) Vuonna 2012 The Standish Group-yrityksen julkaisema CHAOS-manifesto toteaakin ketterien projektien onnistuvan kolme kertaa vesiputousmallia todennäköisemmin. Lisäksi ketterät projektit pysyvät paremmin annetussa aika- ja hintahaarukassa. (19.)

Vuonna 2014 Cast-yrityksen julkaiseman CRASH-tutkimuksen (CAST Research on Application Software Health) mukaan yksittäisellä kehitysmallilla (perinteisellä vesiputousmallilla taikka ketterällä kehitysmallilla) tehdyt ohjelmistot sisältävät muita todennäköisemmin tietoturva-, suorituskyky-, luotettavuus- ja kustannusongelmia. Raportissa todetaan, että esimerkiksi ketteriä malleja käyttäessä voi helposti käydä niin, että ohjelmiston arkkitehtuuriin ei kiinnitetä tarpeeksi huomiota. Yksittäisen kehitysmallin ongelmat ratkaistaankin perinteinen vesiputousmalli ja ketterä malli yhdistämällä. Tutkimuksen ohjelmistoista parhaat arvostelupisteet saivatkin juuri kahta mallia yhdistelleet projektit. Tutkimuksen mukaan laadullisesti huonoimpaan ohjelmiston lopputulokseen päästään kun projektissa ei ole käytetty mitään tiettyä mallia. (20.)

3.8 Vaatimusmäärittelymallin valinta

Tulevan paineilmalaitteiden huoltokirjanpito-ohjelmiston vaatimusmäärittelymalliksi valittiin edellä mainittujen mallien yhdistelmä. Käyttäjiltä kerätään tarvekartoitus, jonka perusteella kuvataan nykyinen järjestelmä. Sidosryhmien vaatimukset kerätään ylös ja niitä havainnollistetaan käyttötapauskaaviolla. Koska työ ei kata ohjelmiston toteutusta, järjestelmävaatimukset kirjataan osaltaan ylös ja prototyypitys suoritetaan alustavin käyttöliittymäkuvin. Päämääränä on saada aikaan tarpeeksi kattava, mutta dokumentointia vain dokumentoinnin takia välttävä vaatimusmäärittely. Määrittely kertoo jo kuvin käyttäjille mitä on luvassa, jotta käyttäjät voivat helpommin kertoa onko heidän tarpeensa saatu täytettyä kattavasti.

Jos projekti toteutettaisiin ketterää kehitysmallia käyttäen, ohjelmistoa tekevälle yritykselle olisi silloin antaa jo valmiiksi käyttäjien tekemät tarinakortit. Lisäksi sprintteihin olisi jo antaa valmiiksi tarvittavaa tukimateriaalia esimerkiksi käyttöliittymäkuvien muodossa.

Toisaalta jos toimittaja suosisi vesiputousmallia, olisi jo tehtyjen dokumenttien perusteella helpompaa tehdä tarvittavat lisäykset, jotta dokumentti olisi kattava vaatimusmäärittely.

4 Case: paineilmalaitteiden huoltokirjanpito-ohjelmisto

Tutkimuksen päämääränä oli selvittää, mitä paineilmalaitteiden huoltokirjanpitojärjestelmältä vaaditaan ja miten löydetyt vaatimukset saadaan kuvattua jatkokehitystä varten, ja samalla arvioida, onko idea huoltokirjanpito-ohjelmistosta toteutuskelpoinen

4.1 Nykytila ja tavoitteet

Pelastuslaitoksella on käytössään useita erilaisia paineilmalaitteita. Näitä laitteita ovat kasvoille asetettavat maskit, hapen paineen sopivalle tasolle asettavat paineenalentimet, sekä paineilmapullot, jotka sisältävät nimensä mukaisesti paineistettua ilmaa. Laitteita käytetään esimerkiksi rakennuksen savusukelluksessa tulipalon yhteydessä. Pe-

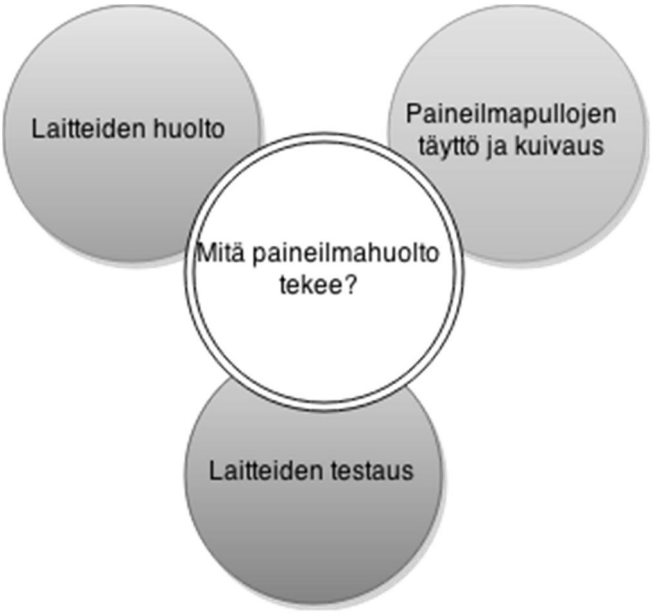
lastuslaitoksella on myös vesisukellukseen tarkoitettuja maskeja sekä muutamia varta vasten vesisukellukseen käytettäviä laitteita. Laitteita on havainnollistettu kuvassa 6.

Kuva 6. Pelastuslaitoksen käyttämiä paineilmalaitteita.

Työturvallisuuden takaamiseksi laitteet vaativat huoltojen seurantaa ja tarkistusta määräajoin. Seurannan nykytila on sekoitus sähköistä ja manuaalista järjestelmää. Paineilmahuollossa laitteiden perustiedot ja huollot on merkitty taulukkolaskentaohjelmiston avulla taulukkoon. Paineilmapulloja täytetään miltei jokaisella paloasemalla ja täyttöjä tekevät nimetyt palomiehet. Paineilmapullojen täytöt ja koeponnistukset kirjataan asemakohtaisesti paperille. Kirjauksista käy ilmi päivämäärä, työn suorittaja sekä pullon sarjanumero. Paineilmapulloja on kahta erilaista mallia, valmistusmateriaaliansa mukaan, teräksisiä sekä komposiittisia. Teräksiset paineilmapullot eroavat komposiittisista siinä, että teräksiset paineilmapullot tulee kuivata kymmenen täyttökerran jälkeen. Komposiitista valmistettuja pulloja ei tarvitse kuivata. Koska pullot pysyvät samalla paloasemalla, tarkistetaan niiden täyttökertojen määrä, kuivaustarve sekä uuden koeponnistuksen tarpeellisuus täyttö-toimenpiteen yhteydessä. Ongelmaksi muodostuu kuitenkin se, että on hyvin vaikeaa saada kokonaiskuvaa esimerkiksi siitä, kuinka paljon

seuraavana vuonna joudutaan pulloja koeponnistamaan. Käytössä oleva taulukkolaskentaohjelmistolla toteutettu laskentataulukko pitää kirjaa esimerkiksi paineenalentajien tulevista huolloista, mutta sen käyttäminen koetaan hankalaksi.

Laitteiden kunnon seuraamiseen liittyy oleellisesti myös niiden testaus. Maskit ja paineenalentimet testataan erillisen testerin avulla, jonka ohjelmisto pitää kirjaa suoritetuista testauksista.



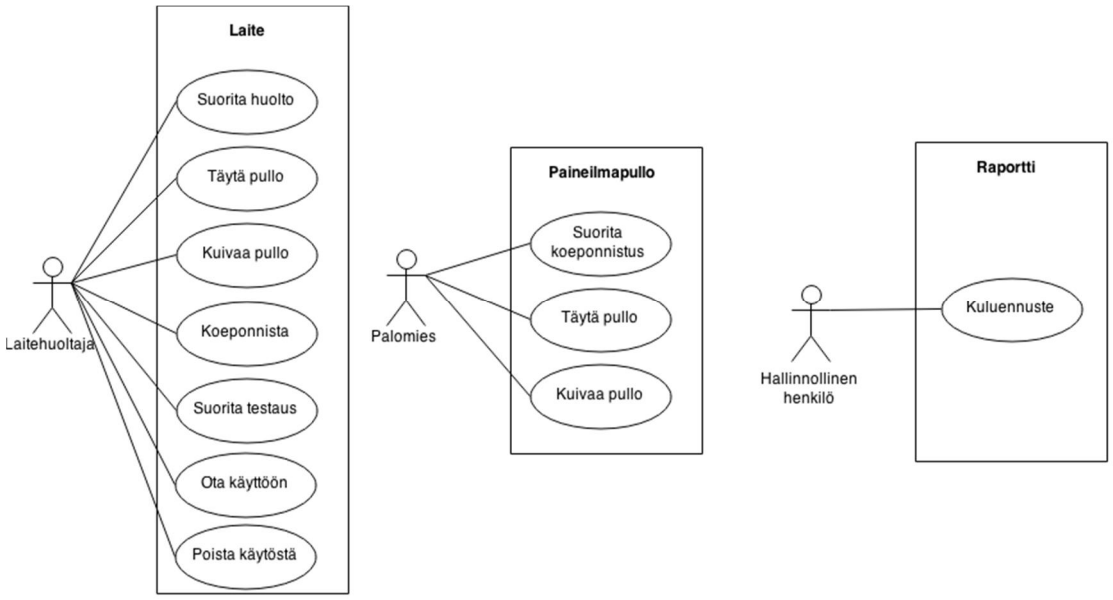
Kuva 7. Paineilmahuollon työtehtävät.

Näiden tietojen perusteella nykytilasta voidaan erottaa kolme merkittävää työvaihetta; laitteiden huolto, paineilmapullojen täyttö ja kuivaus sekä laitteiden testaus. Työvaiheet on esitelty kuvassa 7.

Työn tavoitteena on mahdollistaa yhtenäisen tietojärjestelmän luominen, josta olisi nähtävissä kaikki edellä mainituille laitteille suoritetut toimenpiteet. Kyseinen järjestelmä mahdollistaisi kaikkien toimenpiteiden kirjaamisen samaan paikkaan, jolloin käyttäjät pääsevät eroon käsin kirjatuista taulukoista. Tavoitteena on tuottaa tarpeeksi kattava vaatimusmäärittely, joka mahdollistaa edellä mainitun järjestelmän toteuttamisen.

4.2 Sidosryhmien kartoitus

Sidosryhmien kartoittamiseksi täytyi haastatella hallinnon henkilökuntaa sekä laitteita täyttäviä ja huoltavia henkilöitä. Työssä sidosryhmät rajattiin tietoisesti pelastuslaitoksen sisäpuolelle, vaikkakin oli jo tiedossa, että epäsuorasti sidosryhmiä voi olla pelastuslaitoksen ulkopuolellakin.



Kuva 8. Nykyjärjestelmän sidosryhmät ja toiminnallisuudet.

Järjestelmän sidosryhmiä ovat palomiehet, pelastuslaitoksen paineilmahuollossa laitteita huoltavat henkilöt (laitehuoltaja) sekä pelastuslaitoksen taloutta hoitavat henkilöt (hallinto). Sidosryhmät on havainnollistettu käyttötapauskaaviona kuvassa 8, josta ilmenee jokaisen sidosryhmän nykyjärjestelmässä tekemät toimenpiteet.

4.3 Vaatimusten kartoitus ja hyväksyminen

Tulevalle järjestelmälle asetetaan järjestelmävaatimukset (system requirements) sekä ohjelmistovaatimukset (software requirements), jotka koostuvat toiminnallisista ja ei-toiminnallisista vaatimuksista. Toiminnalliset vaatimukset kuvaavat sen mitä järjestelmä tekee. Ei-toiminnalliset vaatimukset kuvaavat sen miten toiminnalliset vaatimukset toteutetaan.

Keskeiset vaatimukset ohjelmistolle tulevat paineilmahuollosta, jossa laitteita huolletaan. Näihin vaatimuksiin vaikuttaa myös edellä mainittu painelaitelaki, koska paineilmahuollon tulee toimia lain edellyttämällä tavalla. Vaatimukset kerätään kaikilta pelastuslaitoksen sisäisiltä sidosryhmiltä.

Vanhan järjestelmän kuvauksen perusteella voidaan todeta, että jo yksinkertainenkin tietojen sähköistäminen parantaa laitteiden hallittavuutta sekä tiedon luotettavuutta. Näiden syiden takia erillistä käyttökelpoisuusraporttia ei kirjoitettu. Tulevasta järjestelmästä kirjoitettiin vaatimusmäärittely, johon liittyy oleellisena osana uutta järjestelmää havainnollistavat käyttöliittymäkuvat. Näiden kuvien avulla käyttäjiltä voitiin vahvistaa, että tuleva järjestelmä palvelee käyttäjien tarpeita, eikä muuta tarpeettomasti nykyistä toimintamallia. Ratkaisuehdotus tulevan järjestelmän toteuttamiseksi on kuvattu luvussa 5.

Vaatimusten oikeellisuus varmistettiin käyttäjiltä esittelemällä uuden järjestelmän käyttöliittymäkuvia. Tämän lisäksi uuden järjestelmän toteuttamisen mielekkyyttä käyttöliittymäkuvien ja vaatimusmäärittelyn perusteella kysyttiin ohjelmistoalan yritykseltä Ocllo Oy:ltä.

Pelastuslaitoksen osalta haastateltiin viittä ihmistä, joista kaksi työskentelee paineilmahuollossa. Ryhmän kaksi jäsentä työskentelevät pelastuslaitoksella muissa tehtävissä. Ryhmän viimeinen haastateltu henkilö työskentelee palomiehenä, täyttäen toisinaan paineilmapullojakin. Kaikki ryhmän haastatellut ihmiset olivat yksimielisesti sitä mieltä, että käyttöliittymäkuvauksen toteuttava ohjelma vastaisi tarpeisiin ja aiemmin tehtyyn vaatimusmäärittelyyn. Kehityskohteitakin löytyi muutama kappale.

Käyttäjät toivoivat, että tehty tilaus voitaisiin lähettää sähköpostilla suoraan varaosatoimittajalle. Tilaus olisi myös hyvä saada tallennettua ohjelmasta esimerkiksi PDF-muodossa. Toinen aivan uusi kehitysehdotus oli RFID-tekniikan (Radio Frequency Identification) huomiointi laitteiden merkinnässä. Esimerkiksi uusimmista älypuhelimista löytyy jo RFID-lukija, joka sallii RFID-tunnisteen luvun huomattavasti viivakoodeja helpommin, koska etäisyys luettavaan tunnisteeseen voi olla pidempi. Sovellukseen olisi siis hyvä luoda tuki älylaitteiden RFID-lukijoiden käyttämiseksi, vaikkakin alkuvaiheessa käytettäisiin vain viivakoodeja.

Ocllo Oy:stä haastateltiin kahta henkilöä, Timo Virtasta sekä Pasi Kangasta. Yhtenäinen näkemys henkilöillä oli se, että käyttöliittymäkuvat tarkentavine vaatimuksineen ovat oiva apu ohjelmiston toteuttamiselle. Esille tuotiin myös se fakta, että olipa vaatimusmäärittely millainen tahansa, ei sitä missään tapauksessa lähdettäisi toteuttamaan ilman yhteydenpitoa käyttäjiin, jotta lopputulos on varmasti käyttäjiänsä hyvin palveleva.

5 Ratkaisuehdotus

Tässä luvussa kerrotaan yleisellä tasolla ratkaisuehdotus uuden sähköisen paineilmalaitteiden huoltokirjanpidon toteuttamiseksi.

Uusi järjestelmä on Internet-selaimella taikka erillisellä mobiilisovelluksella käytettävä ohjelmisto, johon kirjaudutaan sisään vähintäänkin henkilökohtaista käyttäjätunnusta ja salasanaa käyttäen.

Ohjelmisto mahdollistaa painelaitteiden huoltotoimenpiteiden kirjaamisen huoltotehtäviä suorittaville henkilöille sekä painelaitteiden ylläpitotoiminnot kyseisistä laitteista vastaaville henkilöille. Laitteet tunnistetaan niissä olevan yksilöllisen sarjanumeron perusteella. Mikäli laitteella ei ole yksilöllistä sarjanumeroa, järjestelmästä voidaan tulostaa laitteelle järjestelmän antama yksilöllinen sarjanumero.

Ohjelmassa jokaisella painelaitteella on oma korttinsa, josta on nähtävissä laitteen perustiedot. Lisäksi yksittäiseen laitekorttiin voidaan liittää erilaisia toimenpiteitä, kuten esimerkiksi paineilmapulloilla niiden täyttäminen, kuivaus ja tarkistus. Myös huoltotoimenpiteet liitetään laitteeseen omina kortteinaan. Vanhoja laitteita voidaan poistaa käytöstä ja uusia laitteita voidaan myös lisätä järjestelmään helposti.

Oleellisesti ohjelmistoon liittyvät laitetyyppiin liitettävät hälytykset, joiden avulla on helposti huomattavissa laitteiden vaatimat määräaikaishuollot. Hälytyksiin voidaan liittää aikamääreitä, joiden perusteella laitelistauksissa ja laitekorteissa näytetään värikoodi (vihreä, keltainen, punainen) tilanteen mukaan. Esimerkiksi paineilmapullojen osalta värikoodina näytetään jäljellä oleva aika seuraavaan koeponnistuspäivään. Hengitysventtiilit täytyy huoltaa määräajoin (kahden vuoden välein), joten niiden osalta näytetään jäljellä oleva aika värikoodina. Kun seuraavaan huoltoon on vielä päälle vuosi,

näytetään vihreä väri. Kun aikaa on alle puoli vuotta, muuttuu väri keltaiseksi. Jos määräaika on jo mennyt umpeen, väri on punainen. Värikoodin lisäksi laitehuoltajille lähetään hälytysrajojen täyttyessä sähköpostiviesti, joka sisältää tiedot laitteista, joiden huolto on ajankohtainen. Näin huoltoa vaativista laitteista saadaan tieto ilman tarvetta kirjautua sisään ohjelmaan.

Määräaikaishuoltojen lisäksi esimerkiksi hengitysventtiileille vaaditaan kuuden vuoden välein suoritettava isompi huolto, jossa vaihdetaan enemmän osia. Järjestelmässä täytyykin olla keino merkitä huolto esimerkiksi merkinnällä 6-vuotishuolto, jotta voidaan laskea, mille laitteelle on tehty kyseinen huolto ja milloin se tulee taas ajankohtaiseksi.

Kun laitteelle suoritetaan huolto, luodaan ohjelmistoon uusi huoltokortti, joka liitetään sarjanumerolla laitteen laitekorttiin. Näin laitekortissa nähdään laitteelle suoritetut huoltotoimenpiteet. Huoltokorttiin merkitään huollon tiedot, kuten kuvaus suoritetusta huoltotoimenpiteestä, käytetyt varaosat, huollon suorittaja ja päivämäärä.

Ylläpitotoimintoja ovat laitteiden lisääminen ja poistaminen ohjelmasta, laitteiden korjausmerkintöjen kirjaaminen, huoltojen seuranta, käyttäjätunnuksien lisäys järjestelmään sekä niiden poistaminen järjestelmästä.

Lisäksi ohjelma tarjoaa varaston ylläpidon ja tuotteiden kuluttamisen varastosta kohdistamalla tuotteet yksittäisiin huoltotoimenpiteisiin. Koska varastoon täytyy myös pystyä tuomaan nimikkeitä, on ohjelmassa mahdollista luoda lähetteitä, joihin kirjataan hankitut tuotteet, jonka kautta tuotteet siirtyvät varastosaldolle. Varastokirjanpidon avulla nähdään reaaliaikainen tilanne varastossa vielä jäljellä olevista varaosista ja pystytään ennustamaan kuluja tulevaisuudessa.

Hallintoa varten järjestelmästä tulee saada ulos erilaisia raportteja esimerkiksi laitteiden elinkaarta ja kustannuksia silmällä pitäen.

6 Uuden ratkaisun vaatimusmäärittely

6.1 Toiminnalliset vaatimukset

Tässä luvussa käydään läpi uuden ratkaisun toiminnalliset vaatimukset.

6.1.1 Kirjautuminen järjestelmään

Järjestelmään kirjaudutaan sisään, jonka jälkeen käyttäjän on mahdollista valita itselleen yksi järjestelmään syötetyistä sijainneista. Käyttäjän tulee myös olla mahdollista vaihtaa oma sijaintinsa. Käyttäjän henkilöllisyys täytyy olla varmistettavissa.

- Järjestelmään voidaan kirjautua käyttäen vähintäänkin henkilökohtaista käyttäjätunnusta ja salasanaa.
- Käyttäjä voi valita kirjautuessaan itselleen sijainnin, jota käytetään huoltotoimenpiteiden yhteydessä laitteiden huoltojen suorituspaikkana.
- Käyttäjä voi itse vaihtaa salasanansa uuteen.
- Käyttäjä voi kirjautua ulos.

6.1.2 Järjestelmään kirjattavat laitteet

Tulevaan järjestelmään tulee pystyä luomaan uusia laitteita ja poistamaan niitä tarvittaessa. Laitteita poistetaan silloin, jos niitä luodaan vahingossa taikka jos laite poistetaan käytöstä. Käytöstä poistetut laitteet tulisikin mahdollisesti järjestää järjestelmään erillisen näkymän alle, eikä niitä tulisi poistaa pysyvästi.

Jo olemassa oleva laitekanta tulee voida tuoda uuteen järjestelmään. Nykyiset laitteet on yksilöity, joten uudetkin laitteet on voitava yksilöidä huoltokirjanpidon mahdollistamiseksi. Mikä laitteessa ei ole valmistajan omaa yksilöllistä sarjanumeroa taikka sarjanumero ei ole käyttökelpoinen, tulee järjestelmästä olla tulostettavissa järjestelmän oma yksilöllinen sarjanumero laitteelle joko QR- tai perinteisenä viivakoodina. Nykyisiä laitteita on satoja kappaleita, joten vanhan laitekannan syöttäminen käsin ei ole järkevää.

Järjestelmän haku-toimintoa käytettäessä käyttöä helpottaa, jos sarjanumeroa ei tarvitse syöttää järjestelmään käsin vaan esimerkiksi viivakoodinlukijalla. Viivakoodinlukijalla tarkoitetaan myös mobiililaitteen kameraa, joka osaa lukea sekä QR- että perinteisiä viivakoodeja.

Koska järjestelmään syötetään erilaisia laitteita, on laitetyyppien määrittäminen pakollista, jotta huoltotoimenpiteet voidaan kohdistaa oikealle laitetyypille. Laitetyypit mahdollistavat myös laitetyyppien avulla laitteiden määrän laskemisen. Laitteiden elinkaaren ja talouden hallinnan kannalta on oleellista tietää myös laitteiden hankintatiedot, muut tiedot mukaan lukien.

- Järjestelmään tulee voida tuoda ohjelmallisesti nykyinen laitekanta.
- Järjestelmästä voidaan poistaa laitteita ja siihen voidaan lisätä uusia laitteita.
- Järjestelmästä voidaan tulostaa yksilöllinen sarjanumero laitteelle.
- Järjestelmään kirjatut laitteet voidaan sijoittaa järjestelmässä tiettyyn sijoituspaikkaan (esimerkiksi tietylle paloasemalle).
- Järjestelmän tulostama sarjanumero voidaan lukea viivakoodinlukijalla.
- Laite voidaan hakea järjestelmästä käyttäen viivakoodinlukijaa.
- Järjestelmään on määritettävissä erilaisia laitetyyppejä.
- Järjestelmään on syötettävissä laitteiden perustiedot.

6.1.3 Järjestelmän ylläpitotoiminnot ja raportit

Järjestelmän mahdollistamia toimintoja halutaan rajoittaa niin, että käyttäjät pääsevät käsiksi vain heille tarkoitettuihin tietoihin ja toimintoihin käsiksi. Ylläpitotoiminnot on rajoitettu vain järjestelmän ylläpitäjien käyttöön, joita ovat paineilmahuollossa työskentelevät henkilöt. Hallintaosio tarjoaa pääsyn selaamaan laitteita ja muokkaamaan niiden tietoja, tarkastelemaan raportteja, muokkaamaan järjestelmän asetuksia, hallinnoimaan käyttäjiä, rooleja sekä sijainteja, lisäämään ja muokkaamaan laitetyyppejä, hallinnoimaan varastoa sen nimikkeitä sekä lähetteitä.

- Järjestelmässä on hallintaosio, johon pääsevät käsiksi vain siihen oikeutetut henkilöt.

- Hallintaosion kautta voidaan luoda ja poistaa käyttäjiä.

- Käyttäjille voidaan myöntää rooleja sekä tarvittaessa poistaa rooleja.

- Järjestelmään voi luoda uusia rooleja. Luotuja rooleja voi myös poistaa.

- Jokaiselle roolille on määritettävissä, mihin tietoihin ja toimenpiteisiin kyseisellä roolill pääsee käsiksi.

- Järjestelmästä saa raportin nykyisen ja tulevien vuosien arvioiduista laitteiden huoltomääristä, täyttömääristä, koeponnistuksista sekä kuluista.

- Järjestelmästä saa laitetyyppikohtaisesti listauksen vuositasolla tulevista hälytyksistä.

- Ylläpitotoimintojen kautta pääsee katselemaan laitteiden tietokortteja ja tarvittaessa muuttamaan yksittäisen laitteiden tietoja.

- Järjestelmästä on nähtävissä kaikki yksittäiselle laitteelle tehdyt huoltotoimenpiteet.

- Laitteiden perustiedot ovat ylläpitäjien muokattavissa.

6.1.4 Järjestelmän varasto-osio ja varaosat

Varaston tarkoituksena on näyttää jäljellä olevien varaosien määrä sekä varaston arvo. Tuotteiden hinta ei ole aina vakio, vaan hinnat vaihtelevat. Tilaus täytyy olla mahdollista syöttää alustavasti järjestelmään kustannusarvion saamiseksi. Kun tilaus vastaanotetaan, se kirjataan järjestelmään vastaanotetuksi. Varaosatoimittajilta on saatavissa yleensä vuosittain varaosalistat hintoineen. Järjestelmän varaosat tulee olla päivitettävissä toimittajalta saadun varaosalistan avulla.

- Kirjattuihin huoltotoimenpiteisiin käytetyt varaosat vähenevät varastosaldolta.

- Järjestelmässä on varastonhallinta esimerkiksi varaosien hallintaa varten.

- Varastonhallintaan voidaan lisätä uusia varaosia.
- Varastonhallintaan tuodaan tuotteita lähetteiden avulla.
- Varastonhallinnan tuotteita sekä tuotteiden hintoja voidaan päivittää.
- Varastonhallinnasta tulee voida poistaa varaosia.
- Ylläpitäjä voi määrittää varaosista koostuvia huoltopaketteja.

6.1.5 Järjestelmän hälytykset

Jokaiselle järjestelmään syötetylle laitetyypille tulee olla määritettävissä omat hälytysvälinsä, sekä se mitä huoltotyyppiä hälytysväli koskee. Esimerkiksi paineenalentajille tulisi voida määrittää hälytyksiä niiden määräaikaistarkastuksien perusteella. Mikäli paineenalentaja-laitetyypille asetettu määräaikaistarkistuksen väli (päivissä) on umpeutumassa, näytetään tästä selkeästi tieto laitteiden listauksessa sekä laitteen perustietokortissa. Hälytykset tulee olla mahdollista lähettää laitehuoltajille esimerkiksi sähköpostitse, jotta hälytykset havaitaan ilman järjestelmään kirjautumista.

- Järjestelmään voidaan asettaa erilaisia hälytyksiä koskien esimerkiksi paineilmapullojen täyttökertojen määrää ja koeponnistuksista kulunutta aikaa.
- Järjestelmä viestittää sähköpostitse laitehuoltajille tulevista huolloista.
- Järjestelmän hälytykset ovat asetettavissa laitetyyppikohtaisesti.
- Laitetyyppikohtaiset aktiiviset hälytykset näytetään laitekorttia katselevalle käyttäjälle.

6.2 Ei-toiminnalliset vaatimukset

Järjestelmän järkevän käyttämisen kannalta on oleellista, että käyttöliittymän kielenä on suomenkieli. Koska tuleva järjestelmä tulee korvaamaan aiemmat järjestelmät, tulee järjestelmän olla luotettava ja turvallinen. Luotettavuus edellyttää sitä, että järjestel-

mään tallennettu tieto on eheää ja siihen pääsevät kiinni vain tarkoitetut käyttäjät. Uudella järjestelmällä halutaan pois sulkea sen mahdollisuus, että esimerkiksi paineilmapulloja vain täyttävä henkilö pystyisi poistamaan vahingossa jonkin laitteen. Vain paineilmahuollon henkilökunnan eli järjestelmän ylläpitäjien tulee voida lisätä ja poistaa laitteita sekä tehdä tilauksia varaosista. Myöskään laitteiden tietoja ei saa oikeudettomasti päästä muuttamaan.

Lähtökohtaisesti pelastuslaitokselta edellytetään perustason tietoturvavaatimusten täyttämistä, joten uuden järjestelmän tulee täyttää myös perustason tietoturvavaatimukset.

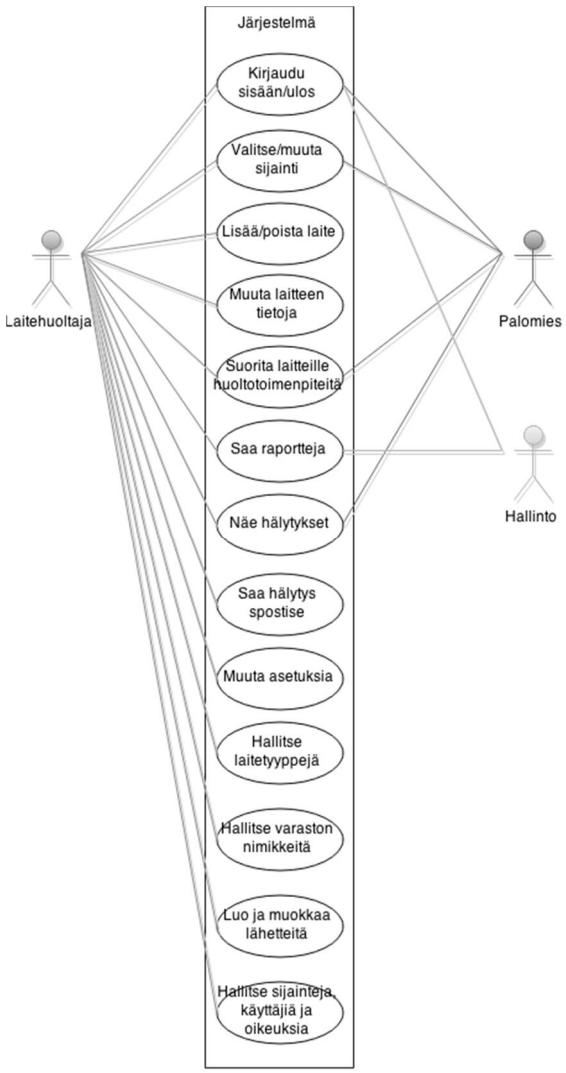
Järjestelmän käyttäminen ei kaikilla henkilöillä ole läheskään päivittäistä, joten ohjelman tulisi olla mahdollisimman helppokäyttöinen, niin että jo muutaman järjestelmän käyttökerran jälkeen käyttäjä osaa jo omatoimisesti hoitaa tehtävänkuvaansa kuuluvat toimenpiteet järjestelmässä. Järjestelmässä saa olla käyttöopas, mutta lähtökohtaisesti on toivottua, että järjestelmä olisi niin yksinkertainen käyttää, ettei sen käyttöönotto edellytä ohjekirjan lukemista.

Järjestelmän tulisi mahdollistaa esimerkiksi CSV-muotoisten tiedostojen avulla tietojen tuonti ja jo olemassa olevien tietojen päivitys järjestelmään, jotta tietoja ei tarvitse syöttää käsin.

- Käyttöliittymän on oltava suomenkielinen.
- Järjestelmään tallennetun tiedon tulee olla eheää ja luotettavaa.
- Eri henkilöt näkevät vain omaan käyttöönsä tarvittavat tiedot.
- Käyttäjät pystyvät suorittamaan vain roolilleen sallittuja toimenpiteitä järjestelmässä.
- Järjestelmää voi käyttää mobiililaitteilla.
- Järjestelmän tulee täyttää perustason tietoturvavaatimukset.
- Järjestelmän käyttöliittymän tulee olla helppokäyttöinen.
- Järjestelmän käyttöön ei vaadita käyttöopasta.
- Varastonhallinnan tuotteet/nimikkeet on voitava päivittää csv-muotoisella tiedostolla.
- Laitteet on voitava tuoda ohjelmaan esimerkiksi csv-tiedostona.

6.3 Käyttötapaukset

Uuden järjestelmän osalta haluttiin selkeästi kuvata se, mihin toiminteisiin eri roolit pääset järjestelmässä käsiksi.



Kuva 9. Laitehuoltajan, palomiehen ja hallinnon työntekijöiden toimet järjestelmässä.

Kuvassa 9 on havainnollistettu eri käyttäjärooleille tyypilliset käyttäjätapaukset. Huomattavana erona laitehuoltajan roolissa verrattuina muihin rooleihin on se, että laitehuoltajalla on mahdollisuus käyttää kaikkia järjestelmän mahdollistamia toiminteita, kun palomiehet käyttävät vain murto-osaa järjestelmän ominaisuuksista. Kuvassa on esitetty myös järjestelmän keskeiset toiminnallisuudet eri käyttäjärooleille.

6.4 Laitekortin perustiedot

Vaatimusten ohella kartoitettiin myös laitteista kirjattavat perustiedot, jotka tulisi täyttää jokaisen laitteen osalta.

Laitteen oleellisia perustietoja ovat laitteen kategoria, valmistaja, malli sekä sarjanumerot. Kategorian avulla nähdään millainen laite on, eli onko kyseessä paineilmapullo vai hengitysventtiili. Valmistaja ja malli ovat yhtä lailla tärkeitä tietoja. Sarjanumeroiden avulla laitteet yksilöidään. Laitteella voi olla yksi tai useampia sarjanumeroita, joiden perusteella laite on tunnistettavissa. Oletusarvoisesti pyritään käyttämään valmistajan omaa, jo tehtaalla laitteelle annettua sarjanumeroa. Mikäli sarjanumeroa ei ole valmistajan toimesta, tulee käyttää uudesta järjestelmästä saatua sarjanumeroa. Vanhemmissa laitteissa voi olla myös esimerkiksi kaiverruksina yksilöiviä tietoja, joten tämäkin tieto täytyy voida tallentaa.

Laitelistauksien osalta on tärkeää tietää missä laitteet ovat. Tämän takia tarvitaan kenttä, johon voidaan kirjata laitteen sijainti, eli tässä käyttötapauksessa paloasema.

Laitteen käyttötietojen osalta on oleellista tietää onko laite vielä käytössä vai onko se mahdollisesti poistettu käytöstä. Laitteiden käyttöönottamisen päivämäärä tulee voida tallentaa, jotta tiedetään kuinka vanhoja laitteet ovat. Laitteen iän lisäksi on tarve voida arvioida sanallisesti laitteen kuntoa esimerkiksi termein hyvä, välttävä tai huono.

Tarkastustietojen osalta on oleellista tietää kuka on viimeksi laitteen tarkastanut. Näiden tietojen tulisi päivittyä itsestään, sitä mukaa kun laitteelle tarkastuksia tai huoltotoimenpiteitä tehdään. Muita aikaan liittyviä perustietoja ovat tiedot siitä milloin laite on järjestelmään lisätty sekä milloin laitekorttia on viimeksi päivitetty ja kenen toimesta.

Kun laite on hankittu, tulee kirjata ylös valmistajan mahdollisesti laitteelle ilmoittama käyttöikä, jotta vanhentuvat laitteet voidaan poistaa käytöstä ajallaan tai huoltaa tarvittaessa. Kustannuksien kirjaamiseksi tulee kirjata ylös laitteen ostohinta sekä ostopäivämäärä, unohtamatta hankintapaikkaa.

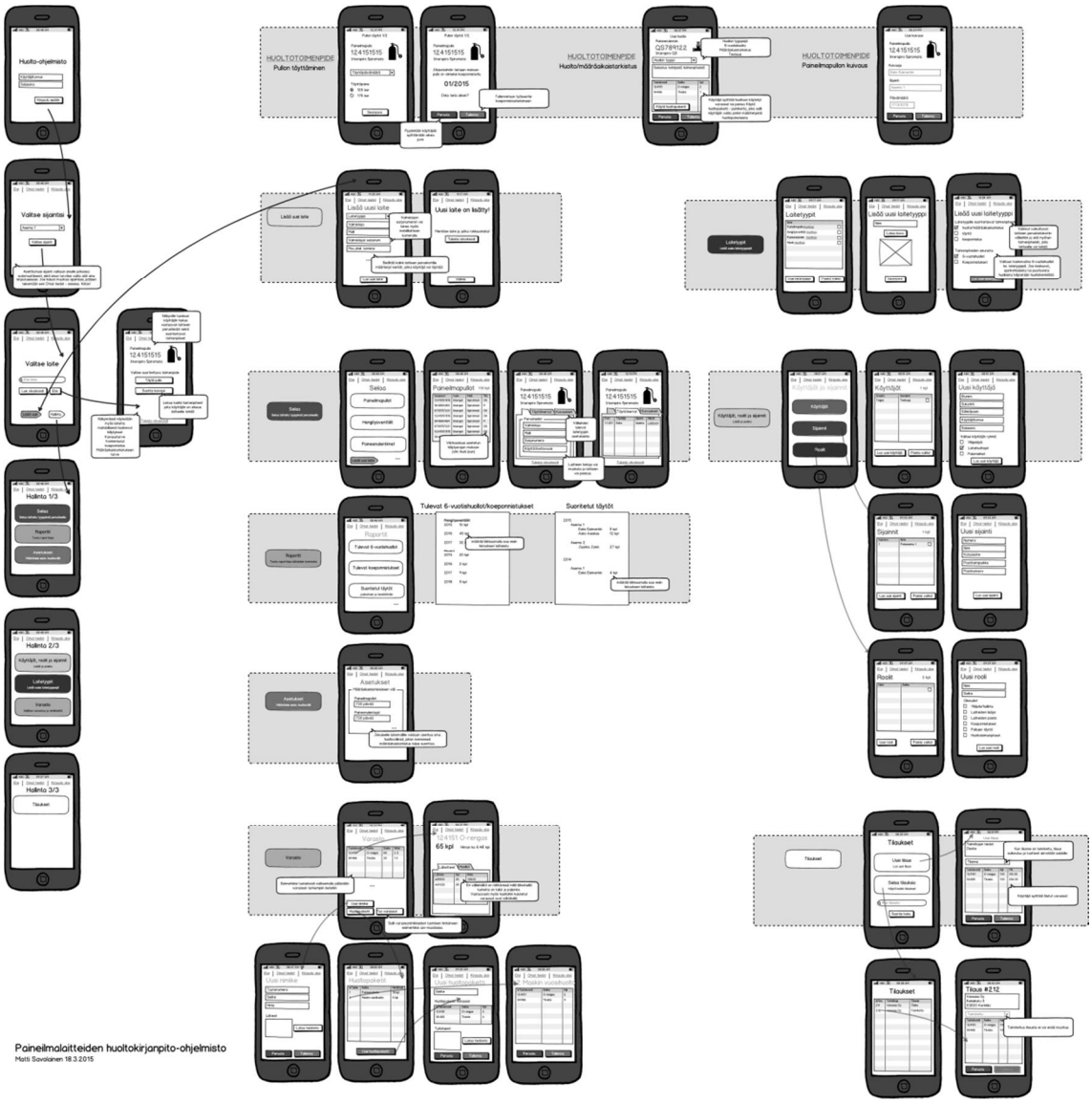
Edellä mainittujen seikkojen perusteella voidaan todeta laitekortin perustietojen olevan seuraavat:

- Kategoria
- Valmistaja
- Mall
- Valmistajan sarjanumero
- Muu yksilöllinen tunniste
- Järjestelmän sarjanumero
- Sijoituspaikka (paloasema)
- Laitteen tilanne
- Käyttöönoton päivämäärä
- Kunto
- Laitteen edellinen tarkastaja
- Lisätty tietokantaan
- Lisännyt tietokantaan
- Päivitetty
- Päivittänyt
- Valmistajan ilmoittama käyttöikä
- Ostopäivämäärä
- Hinta
- Hankintapaikka

6.5 Käyttöliittymäkuvat

Vaatimusmäärittelyn esittämiseksi sidosryhmille sekä asioiden havainnollistamiseksi vaatimusmäärittelyn perusteella suunniteltiin järjestelmän käyttöliittymän prototyyppi, joka sisältää kaikki käyttäjien käyttämät näkymät.

Käyttöliittymäkuva piirrettiin Balsamiq Mockups-ohjelmistoa käyttäen. Tavoitteena oli luoda keskeiset ominaisuudet esille tuova ”rautalankamalli”, joka tukee vaatimusmäärittelyä.



Kuva 10. Käyttöliittymäkuvat.

Kuvassa 10 on esitelty käyttöliittymän prototyyppi. Tyypillisessä käyttötapauksessa käyttäjä aloittaa järjestelmän käytön kirjautumalla sisään. Kirjautuminen on havainnollistettu kuvan 10 vasemmassa yläreunassa. Kirjautumisen jälkeen siirrytään vaiheittain ohjelmistossa eteenpäin. Kuvan vasemmassa reunassa on esitelty sovelluksen päävalikko kuutena erillisenä näkymänä, joista kolme ensimmäistä olisivat tavallisen käyttäjän nähtävissä.

Harmaalla pohjalla olevat kokonaisuudet havainnollistavat kunkin yksittäisen valikon takaa aukeavaa toiminnallisuutta.

Tehty käyttöliittymäkuvitus osoittautui todella oivaksi keinoksi esitellä vaatimusmäärittelyä ja oli samalla oiva työkalu vaatimusmäärittelyn tekemisen tukena.

7 Yhteenveto

Insinöörityö lähti liikkeelle tarpeesta ajanmukaistaa ja yhtenäistää pelastuslaitoksen paineilmalaitteiden huoltokirjanpitoa. Vaatimusmäärittely sujui käyttäjien kanssa ongelmitta, kiitos paineilmahuollon työntekijöiden, jotka osasivat kertoa selkeästi sen mitä he toimintansa tueksi tarvitsisivat.

Opinnäytetyön alkupuolella läpi käydyt vaatimusmäärittelyn työvaiheet selvensivät huomattavasti vaatimusmäärittelyn prosessia ja niitä päästiin hyödyntämään myös tämän työn aikana, joskaan kaikkia työvaiheiden suorittamista ei tämän työn osalta nähty järkeväksi. Kahden eri ohjelmistokehitysmallin pohjalta koostettu hybridimalli osoittautui ainakin tämän mittakaavan projektissa järkeväksi valinnaksi. Mallin lopullinen palaute saadaan tosin vasta sitten kun ohjelmisto on aikanaan mahdollisesti toteutettu ja mahdolliset lisätarkennukset vaatimusmäärittelyyn on annettu. Tämän työn ja aiemman työkokemuksen perusteella näkisin puhtaasti perinteisen vaatimusmäärittelyn perusteella toteuttavan ohjelmistoprojektin olevan hyvin riskialtis jo siinä suhteessa, että tilaaja saa jotain muuta kuin alkujaan oletti. Unohtamatta sitä, että täysin kattavan vaatimusmäärittelyn tuottaminen on hyvin raskas prosessi.

Niin käyttäjiltä kuin ohjelmistokehittäjiltäkin tuli positiivista palautetta vaatimusmäärittelyn käyttöliittymäkuvista. Varsinkin sidosryhmille, eli käyttäjille, oli huomattavasti helpompi esitellä tulevaa ohjelmistoa, kun ei tarvinnut tukeutua vain tekstiin. Myös kysymyksiä oli selkeästi vähemmän, kiitos havainnollistavan kuvituksen. Balsamiq Mockups-ohjelmisto koettiin loistavaksi ohjelmaksi piirtää käyttöliittymäkuvia helppokäyttöisyytensä ansioista.

Vaatimusmäärittelyssä suuri vastuu on sidosryhmillä, joiden toiminta pitäisi pystyä ymmärtämään hyvin, jotta vaatimukset osataan kirjata ylös. Tapauksesta riippuen tämä voi olla hyvinkin haastavaa. Tämän insinöörityön tapauksessa suuren kiitoksen ansaitsee Keski-Uudenmaan pelastuslaitoksen paineilmahuollon esimies Sami Ranta, joka on oman alansa rautainen ammattilainen. Ranta osasi kertoa selkeästi osastonsa tarpeet ja visiot, joka auttoi huomattavasti työn toteuttamista. Vastaan ei tullut tilannetta,

jossa sidosryhmän tarvetta olisi joutunut arvailemaan. Tämän perusteella voidaankin todeta, että vaatimusmäärittelyä tehdessä, sidosryhmästä kannattaa mahdollisuuksien mukaan valita haastateltavaksi henkilö, jolla on jo visio siitä, miten asioita voitaisiin parantaa. Lopputulos kun saattaa olla täysin erilainen, jos haastateltava kohde ei ole kiinnostunut omien työskentelytapojensa kehittämisestä.

Insinöörityön tekeminen vahvisti ennakkoaavistuksia vesiputousmallin raskaudesta, lisäten samalla huomattavasti tekijänsä tietoutta ja kokemusta itse vaatimusmäärittelyn tekemisestä. Nähtäväksi jää vielä, toteutetaanko vaatimusmäärittelyn perusteella uusi vanhan korvaava nykyaikainen paineilmalaitteiden huoltokirjanpito-ohjelmisto.

Lähteet

1 Paakki, Jukka. 2011. Ohjelmistojen vaatimusmäärittely. Verkkodokumentti. Helsingin yliopisto. <http://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf>. Luettu 18.3.2015.

2 Donaldson, Scott , Siegel, Stanley G. 2000. Successful Software Development 2nd Edition. Kirja. Prentice Hall PTR. Luettu 18.3.2015.

3 The Institute of Electrical and Electronics Engineers. 1998. IEEE Recommended Practice for Software Requirements Specification. Sähköinen dokumentti. Luettu 1.3.2015.

4 Julkisen hallinnon neuvottelukunta. 2009. JHS 165 ICT-palvelujen kehittäminen: Vaatimusmäärittely. Verkkodokumentti. JUHTA. <http://www.jhs-suositukset.fi/c/document_library/get_file?uuid=b8118ad7-8ee4-459a-a12b-f56655e4ab9d&groupId=14>. Luettu 1.3.2015.

5 Schedlbauer, Martin. 2012. The quest for good requirements. Verkkodokumentti. BusinessAnalystTimes. <http://www.batimes.com/articles/the-quest-for-good-requirements.html>. Luettu 2.3.2015.

6 Betterton, Robert D. 2001. Design Web Applications the Systems Engineering Way. Verkkodokumentti. University of Maryland. <http://www.rdbprime.com/researchPapers/SoftwareEngineering/SE_TermPaper.html>. Luettu 2.3.2015.

7 Munassar Nabil Mohammed; Govardhan A. 2010. A Comparison Between Five Models Of Software Engineering. Verkkodokumentti. International Journal of Computer Science Issues. <http://www.ijcsi.org/papers/7-5-94-101.pdf>. Luettu 2.3.2015.

8 Hull, Elizabeth; Jackson, Ken; Dick, Jeremy. 2005. Requirements Engineerind Second Edition. Kirja. Springer Science+Business Media. Luettu 12.2.2015.

9 Suomen riskienhallintayhdistys ry. 2012. Mistä riskienhallinnassa on kysymys. Verkkodokumentti. Suomen riskienhallintayhdistys ry. <http://www.pk-rh.fi/index.php?page=riskienhallinta>. Luettu 12.1.2015.

10 Islam, S; Houmb ,S.H. 2010. Integrating Risk Management Activities into Requirements Engineering. Verkkodokumentti. IEEE Xplore .<http://ieeexplore.ieee.org.ezproxy.metropolia.fi/xpl/articleDetails.jsp?tp=&arnumber=5507389>. Luettu 13.1.2015.

11 Cerejo, Lyndon. 2010. Design Better And Faster With Rapid Prototyping. Verkkodokumentti. Smashing Magazine. <http://www.smashingmagazine.com/2010/06/16/design-better-faster-with-rapid-prototyping/>. Luettu 17.1.2015.

12 Horvath, Kristof. 2014. Risk Management in Agile and Waterfall Enviroments. Verkkodokumentti. Intland Software.

<http://intland.com/blog/sdlc/risk-management-in-agile-and-waterfall-environments/>. Luettu 19.1.2015.

13 Ylimannela, Ville. 2010. A model for risk management in agile software development. Verkkodokumentti. Tampere University of Technology. <http://www.cloudsw.org/under-review/a6f468c9-4857-4206-96ee-f67df0583d41/file_initial_version>. Luettu 22.1.2015.

14 Tutorialspoint. SDLC Waterfall Model. Verkkodokumentti. <http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm>. Luettu 1.2.2015.

15 Useita allekirjoittajia. Manifesto for Agile Software Development. Verkkodokumentti. <http://agilemanifesto.org/>. Luettu 2.2.2015.

16 VersionOne.com. 2013. 7th Annual State of Agile Development Survey. Verkkodokumentti. <http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf>. Luettu 2.2.2015.

17 Shore, James. 2007. The Art of Agile Development. Kirja. O'Reilly.

18 Agile Alliance. 2013. Backlog. Verkkodokumentti. <http://guide.agilealliance.org/guide/backlog.html>. Luettu 5.3.2015.

19 Cohn, Mike. 2012. Agile Succeeds Three Times More Often Than Waterfall.Mountain Goat Software. Verkkodokumentti. <http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall>. Luettu 5.3.2015.

20 Cast Software. 2014. New report confirms Agile/Waterfall Mix Produces the Best Code Quality. Cast Software. Verkkodokumentti. <http://www.castsoftware.com/news-events/press-release/press-releases/new-report-confirms-agile-waterfall-mix-produces-the-best-code-quality>. Luettu 10.3.2015.