jamk.fi

Yleisen heitinohjelmistojen suunnittelutyökalun vaatimukset ja määrittely ohjelmistoergonomian periaatteiden mukaisesti

Aleksi Borén

Opinnäytetyö
Kesäkuu 2016
Tekniikan ja liikenteen ala
Teknologiaosaamisen johtamisen koulutusohjelma
Ylempi ammattikorkeakoulututkinto

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä	Julkaisun laji	Päivämäärä
Borén, Aleksi	Opinnäytetyö, ylempi AMK	3.6.2016
	Sivumäärä 62+ 26	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Yleisen heitinohjelmistojen suunnittelutyökalun vaatimukset ja määrittely ohjelmistoergonomian periaatteiden mukaisesti

Tutkinto-ohjelma

Teknologiaosaamisen johtaminen, Ylempi AMK

Työn ohjaajat

Hautanen Juha ja Siistonen Matti

Toimeksiantaja

Työn tekijän oma aihe

Tiivistelmä

Nykyaikaiset omasuojaheittimet tarjoavat suuren määrän erilaisia toimintoja, joiden tehokas hyödyntäminen edellyttää tarkkaa ohjelmointia. Ohjelmoimiseen käytettävät työkalut ovat usein käytettävyydeltään huonoja ja johtavat jopa omasuojaheittimien ominaisuuksien vajaaseen hyödyntämiseen.

Suunniteltaessa omasuojalaitteiden heiteohjelmat riittävän yksityiskohtaisesti jo ennen varsinaisen omasuojaheitinkohtaisen ohjelmointityökalun käyttämistä, voitaisiin vähentää siinä olevien käytettävyyden heikkouksien vaikutuksia ohjelmoinnin lopputulokseen. Niinpä opinnäytetyössä selvitettiinkin mahdollisuutta suorittaa tämä suunnittelu kokonaan erillisellä suunnittelutyökalulla ja mitkä olisivat silloin tämän työkalun vaatimukset.

Työn ensimmäisessä vaiheessa arvioitiin ohjelmiston tarpeellisuutta ja muodostettiin vaatimusmäärittely neljän eri tutkimuksen perusteella. Niin kyselyssä, ryhmähaastattelussa kuin havainnointitutkimuksessakin käytettiin samaa, mahdollisen uuden työkalun käyttäjiä hyvin edustavaa ryhmää. Erilaisten omasuojaheittimien ominaisuuksista hankittiin tietoa perehtymällä julkisiin lähdemateriaaleihin. Toisessa vaiheessa muodostettiin ohjelmistosta toiminnallinen määrittely, joka vastasi asetettuihin vaatimuksiin. Muodostetun toiminnallisen määrittelyn toimivuuden varmistamiseksi, sille suoritettiin käyttäjätesti.

Työn tuloksena muodostettiin vaatimusmäärittely sekä ensimmäinen versio toiminnallisesta määrittelystä ohjelmistolle, joka vähentäisi nykyisten omasuojaheittimien ohjelmointityökalujen käytettävyyden ongelmista aiheutuvia haittoja. Todettiin että tällaisella työkalulla voitaisiin parantaa omasuojaheittimien ohjelmoinnin lopputulosta ja helpottamaan ohjelmoijien työtä. Työkalulta vaaditut ominaisuudet tiedetään ja se on toteutettavissa tuotetun toiminnallisen määrittelyn pohjalta.

Avainsanat (asiasanat)

ohjelmisto, ohjelmistoergonomia, ohjelmointi, omasuojaheitin, toiminnallinen määrittely, vaatimusmäärittely

Muut tiedot

jamk.fi

Description

Author	Type of publication	Date
Borén, Aleksi	Master's thesis	3.6.2016
		Language of publication: Finnish
	Number of pages 62 + 26	Permission for web publication: x

Title of publication

Requirements and design of a universal dispense program design tool utilizing software ergonomics methods

Degree programme

Master of Engineering, Technological Competence management

Supervisors

Hautanen Juha and Siistonen Matti

Assigned by

-

Abstract

Modern countermeasure dispensers offer a great variety of features for the end user. This in turn demands very detailed work while programming them, but the tools provided, often lack the usability to support this. This may result in ineffective use of the provided features.

If the dispense programs could be designed in good detail, before introducing the actual programming tools, it could decrease the effects of the usability problems of the tools in the end result. Therefore, the possibilities to do this planning with dedicated planning software, were assessed and the requirements for this tool were gathered.

In the first phase, the necessity for this tool was evaluated and a requirements specification was formed, based on four separate studies. The same representative group was used in all of the studies, including the enquiry, group audition and observing the use of the actual tool. Information about the different features of countermeasure dispensers was gathered, studying freely available material especially online. In the second phase, a functional design document was created, considering the requirements placed at the first phase. To assure that the functional design had met its goals, the design was put into a user test.

As a result, a requirements specification document and a first version of the functional design document for the new software program were produced. It was acknowledged that using this kind of software, would increase the quality of the end result in programming and help the programmers in their work. The requirements for the program are known and the program may be engineered using the produced functional design document.

Keywords/tags (subjects)

Functional design document, programming, requirements specification, countermeasure dispenser, software, software ergonomics

Sisältö

Kuvat

Taulukot

Termit

Tähän on koottu erilaisia tässä työssä esiin tulevia termejä, jotka eivät ole alaa tuntemattomille välttämättä ennestään tuttuja. Listauksen tarkoituksena on ainoastaan helpottaa tämän työn lukemista, eikä sisältää kaikkia käytettyjä termejä.

Alustustiedosto: Tiedosto, joka sisältää pysyviä asetuksia tai valintoja, joita jokin ohjelma tai laite lukee ja käyttää oman toimintansa määrittelemiseen.

Heitejakso: Yksi heitejakso kuvaa täydellisesti valitun heitemallin heittotapahtumat, sisältäen vähintäänkin tarkat heittoajat sekä mahdollisesti myös valinnan heitteen heittopaikasta. Samassa heiteohjelmassa voidaan, omasuojaheittimestä riippuen, suorittaa yksi tai useampia heitejaksoja rinnakkaisesti. Rinnakkaiset heitejaksot voivat tyypillisesti sisältää myös keskenään samaa heitemallia.

Heiteohjelma: Yksittäinen, ajallisesti rajattu, heitteiden heittämiskokonaisuus, joka sisältää yhtä tai useampaa heitemallia erilaisina yhdistelminä. Heiteohjelma on siis yksi kokonaisuus jota voidaan suorittaa peräkkäin, mutta tyypillisesti ei samanaikaisesti. Yksi heiteohjelma voi omasuojaheittimestä riippuen, sisältää yhden tai useita rinnakkaisia heitejaksoja, jotka yhdessä muodostavat heiteohjelman.

Kognitiivinen kapasiteetti: Ihmisen kyky käsitellä tietoa, eli vastaanottaa, käsitellä, muistaa ja käyttää tietoa.

*Käyttäjäkokemus: ”*Henkilön havainnot ja vasteet jonkin tuotteen, järjestelmän tai palvelun käytöstä tai sen ennakoidusta käytöstä. Huomio 1: Käyttäjäkokemus sisältää kaikki käyttäjän tunteet, uskomukset, mieltymykset, havainnot, fyysiset ja psyykkiset vasteet, käyttäytymisen ja aikaansaannokset, jotka tapahtuvat käytön aikana ja sen jälkeen.” (ISO 9241) Opinnäytetyön tekijän vapaasti suomentamana.

OFP (Operational Flight Program): Omasuojalaitteen laiteläheinen perusohjelmisto, joka toteuttaa laitteen perustoiminnallisuuden ja mahdollistaa UDF:n suorittamisen.

Ohjelmistoergonomia: Ohjelmistotuotannon tutkimusala, jonka tarkoituksena on parantaa ohjelmistojen sopivuutta ihmisten käyttöön ja vähentää niiden käytöstä aiheutuvia haittoja. Tavoitteena on sopeuttaa tietotekniset järjestelmät ihmisen toimintakykyyn ja kykyyn käsitellä tietoa niin että ihminen pystyy käyttämään niitä mahdollisimman tehokkaasti, ilman tarpeetonta rasitusta.

Omasuojaheite: Soihtu, silppupaketti tai muu vastaava kertakäyttöinen, omasuojaheittimestä heitettävä tai vapautettava väline. Yleensä yhden kokonaisen omasuojaheitteen sisältö heitetään kokonaisuudessaan ulos heittotapahtuman yhteydessä, mutta myös useampia heitelatauksia sisältäviä omasuojaheitteitä on olemassa. Omasuojaheitteen tarkoitus on estää, hidastaa, vaikeuttaa tai estää omasuojaheitteitä käyttävään laitteeseen kohdistuvaa asevaikutusta.

Omasuojaheitin: Laite joka heittää tai vapauttaa soihtuja, silppupaketteja tai muita vastaavia kertakäyttöisiä omasuojaheitteitä.

Omasuojaheittimen ohjaustiedosto (myöhemmin ohjaustiedosto): Omasuojaheittimeen ladattava ohjelmisto, joka sisältää määrittelyt omasuojaheittimen toiminnasta eri tilanteissa. Sisältää mm. määrittelyt siitä millaisia heitesarjoja missäkin toimintatilassa ja -tilanteessa kutakin uhka-asejärjestelmää vastaan heitetään. Tämä on omasuojaheittimen UDF, eli User Data File (katso jäljempänä). Joskus tästä käytetään myös nimitystä MDF (Mission Data File).

UDF (User Data File, myös MDF Mission data File): Omasuojalaitteen käyttäjätiedosto, jossa määritellään miten laitteen halutaan reagoivan tai toimivan uhkatilanteessa. (Kts. Omasuojaheittimen ohjaustiedosto.)

Vaatimusmäärittely: Kokoelma erillisistä vaatimuksista joita tuotteelle on asetettu.

1 Johdanto

Nykyaikaisia sotilaslentokoneita suojataan vastustajan asevaikutukselta monin erin tavoin. Yksi näistä tavoista on erilaisten omasuojalaitteiden sisällyttäminen lentokoneeseen. Omasuojalaitteiden tehtävä on nimensä mukaisesti suojata omaa lavettia vastustajan asevaikutukselta. Omasuojalaitteita on olemassa eri tarkoituksiin monen tyyppisiä ja yleensä niistä valitaan juuri kuhunkin tarkoitukseen sopiva yhdistelmä ja ne liitetään toimimaan yhteistyössä keskenään. Tässä työssä keskitytään tarkastelemaan omasuojaheittimiä, jotka ovat yksi näistä monista eri omasuojalaitteista. Sotilaslentokoneeseen asennetun omasuojaheittimen tarkoituksena on harhauttaa vastustajan asejärjestelmän seurantaa tai maksimoida laukaistun ohjuksen ohitusetäisyys ja siten lisätä oman lentokoneen selviytymismahdollisuuksia taistelutilanteessa. Tämän saavuttamiseksi omasuojaheitin on etukäteen ohjelmoitava vastaamaan optimaalisesti kulloiseenkin uhkan ja lentotilan muodostamaan uhkatilanteeseen. Jokaista eri uhkatilannetta varten taas täytyy suunnitella ja ohjelmoida oma heitekokonaisuus, joka toimii parhaiten juuri siinä tilanteessa. Lisää itse omasuojalaitteista ja erityisesti omasuojaheittimistä on kerrottu osassa kaksi.

Nykyisellään omasuojaheittimien heiteohjelmat ohjelmoidaan heiteohjelmistogeneraattoreilla (myöhemmin generaattori), joiden käyttöliittymät ovat pääosin hyvin laiteläheisiä ja monimutkaisia. Generaattorien käyttö on usein epäintuitiivista ja vaatii paljon opiskelua sekä muistamista. Erityisen ongelmallista generaattoreissa on, etteivät ne tyypillisesti tue ohjelmoinnin kattavuuden varmistamista, vaan ohjelmoinnin varmistaminen jää täysin käyttäjän tekemien tarkastusten ja testien varaan. Lisäksi generaattorit ovat visuaalisesti alkeellisia, eivätkä tue nopeaa tietosisällön havainnointia. Nämä saattavat vaikuttaa ohjelmoinnin lopputulokseen ja siten sotilaslentokoneiden selviytymiseen taistelussa. Kokemuksen mukaan omasuojaheittimien laitevalmistajat ovat kuitenkin haluttomia parantamaan omien generaattoreidensa käytettävyyttä.

1.1 Työn tavoitteet ja tutkimusasettelu

Koska nykyiset heiteohjelmistogeneraattorit eivät tue käyttäjän työskentelyä, tässä työssä selvitetään mahdollisuuksia vaikuttaa ohjelmoinnin lopputulokseen, erillisen suunnitteluohjelman avulla. Keskeinen tavoite onkin selvittää vaatimukset ja toteuttamismahdollisuudet ohjelmalle, jolla pystyttäisiin erityisen hyvän ohjelmistoergonomian avulla parantamaan ohjelmoinnin lopputulosta. Työssä keskitytään suunnittelemaan ohjelmalle mahdollisimman hyvä käyttäjäkokemus, eli kaikki käyttäjälle näkyvät ominaisuudet pyritään saamaan mahdollisimman vaivattomiksi käyttää. Lisäksi pyritään tukemaan käyttäjän työskentelyä hyvien visuaalisten keinojen sekä ohjeiden avulla. Käyttäjälle näkymättömien ominaisuuksien ja teknisten yksityiskohtien määrittelyä tehdään mahdollisimman vähän.

Tämä opinnäytetyö jakautuu käytännössä kahteen eri osaan, joista ensimmäisessä etsitään vastauksia seuraaviin kysymyksiin: Onko tässä työssä määriteltävän ohjelman kaltaiselle tuotteelle todellista tarvetta ja olisiko siitä hyötyä käyttäjälle? Lisäksi selvitetään mitä käyttäjät haluavat tuotteen avulla saavuttaa ja mitä vaatimuksia käyttäjät sille asettavat. Saatujen vastausten perusteella päätellään toisen osan tarpeellisuus ja mahdollisuudet sen toteuttamiseen. Mikäli ohjelman määritteleminen nähdään saatujen tulosten perusteella järkeväksi, tuotetaan ohjelmalle vaatimusmäärittely. Ensimmäisen osan tutkimus keskittyy siis vastaamaan seuraavaan kysymykseen: Tuottaako tällainen ohjelma hyötyä sen käyttäjille ja millä ehdoilla?

Opinnäytetyön toisessa vaiheessa ohjelmalle tuotetaan toiminnallinen määrittely. Sen tuottamisessa käytetään perustana ensimmäisessä osassa tuotettua vaatimusmäärittelyä. Toiminnallisen määrittelyn tavoitteena on ensisijaisesti määritellä ohjelman tuottama käyttäjäkokemus, ei niinkään sen toteuttamista tai teknisiä yksityiskohtia. Tämä tapahtuu pääasiallisesti määrittelemällä ohjelman käyttölogiikka ja käyttöliittymä riittävällä tarkkuudella. Tässä hyödynnetään ohjelmistotekniikan perusteita sekä ohjelmistoergonomisia periaatteita. Toisaalta ohjelman käytännön toteuttamiseen tai siinä käytettäviin välineisiin ei oteta kantaa, mikäli se ei osoittaudu pakolliseksi.

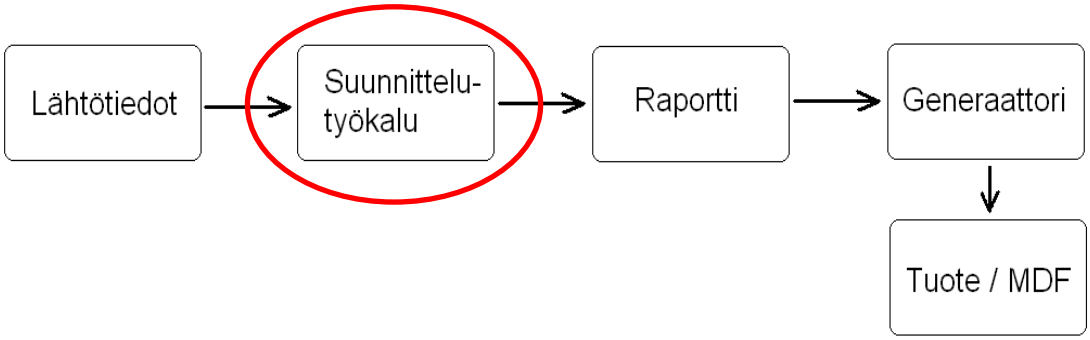
Edellä kerrotuista syistä ohjelman toiminnallinen määrittely ei anna suoria vastauksia ohjelmointityön varsinaiseen toteutukseen tai esimerkiksi ohjelmiston käyttämän tietokannan ominaisuuksiin. Siksi se ei vielä mahdollista ohjelmiston varsinaisen ohjelmointityön toteuttamista, vaan se vaatii vielä erillisen toteutussuunnitelman muodostamisen, mikä ei sisälly tähän työhön. Toteutussuunnitelmassa määritellään tässä työssä tuotettavan toiminnallisen määrittelyn lisäksi useita toteutukseen liittyviä, mutta loppukäyttäjälle näkymättömiä teknisiä yksityiskohtia. Tämän määrittelyn pohjalta tiedetään kuitenkin melko yksityiskohtaisesti se, miltä ohjelman halutaan näyttävän ja miten sen halutaan toimivan loppukäyttäjän näkökulmasta, eli keskitytään määrittelemään ohjelmiston käyttäjäkokemus. Tämän toisen osan perusteella pyritään siis vastaamaan kysymykseen: Pystytäänkö ohjelman toiminnallisessa määrittelyssä täyttämään käyttäjien vaatimukset ja samalla saavuttamaan hyvä ohjelmistoergonomia?

1.2 Tarpeet suunnitteluohjelman taustalla

On olemassa joukko valmiita työkaluja, joilla voidaan suunnitella ja simuloida erilaisia heitesekvenssejä sekä arvioida niiden toimivuutta simulaatioilla. Tämän jälkeen toimiviksi todetut heitesekvenssit voidaan suoraan syöttää käytössä oleviin työkaluihin heiteohjelmiston tuottamiseksi. Voisi ajatella ettei näiden kahden työvaiheen väliin tarvita tässä työssä määriteltävän kaltaista ohjelmaa. Erilaisia heitesekvenssejä suunniteltaessa ja erityisesti niitä simuloitaessa huomataan kuitenkin tyypillisesti, että heitesekvenssien vaikuttavuus vaihtelee hyvin voimakkaasti riippuen yksittäisistä muuttujista varsinaisessa simulaatioskenaariossa. Niinpä yhden, jokaisessa tilanteessa, suurella todennäköisyydellä, riittävän hyvin toimivan, heiteohjelman löytäminen on erittäin vaikeaa tai jopa mahdotonta. Yksittäiseen muuttujiltaan rajoitettuun tilanteeseen toimiva heiteohjelma löydetään helpommin. Niinpä erilaisten heittimien ominaisuuksien rajoissa, voidaan muodostaa hyvinkin suuri määrä erilaisia skenaarioita, joissa muuttujien tila on rajattu ja siihen on löydetty kohtuullisen optimoitu heitesekvenssi.

Erilaisia, juuri tiettyyn tilanteeseen optimoituja, heitesekvenssejä voi olla erittäin suuri määrä ja niiden hallitseminen ilman erillistä työkalua, voi olla hyvinkin haastavaa. Jos käytössä oleva generaattoriohjelma on lisäksi hankala käyttää, eikä

tue ohjelmoinnin kattavuutta, voi tämän työn aiheena olevan ohjelmiston kaltainen tuote edistää hyvinkin merkittävällä tavalla omasuojaheittimen ohjelmoijan työtä. Näistä syistä opinnäytetyössä päätettiin tutkia, voisiko tällaisen tuotteen määritellä ja voisiko se onnistuneesti yhdistää teoreettisen tietämyksen ja omasuojalaitteiden ohjelmoinnin toisiinsa, niin että se kokonaisuutena parantaisi ohjelmoinnin lopputulosta. Tämän ohjelmiston sijoittuminen osaksi tuotantoketjua, jossa teoreettinen tieto muutetaan käytännön tuotteeksi, on kuvattu alla kuvassa 1.



Kuva 1. Määritettävän työkalun sijoittuminen ohjaustiedoston muodostamisen prosessissa.

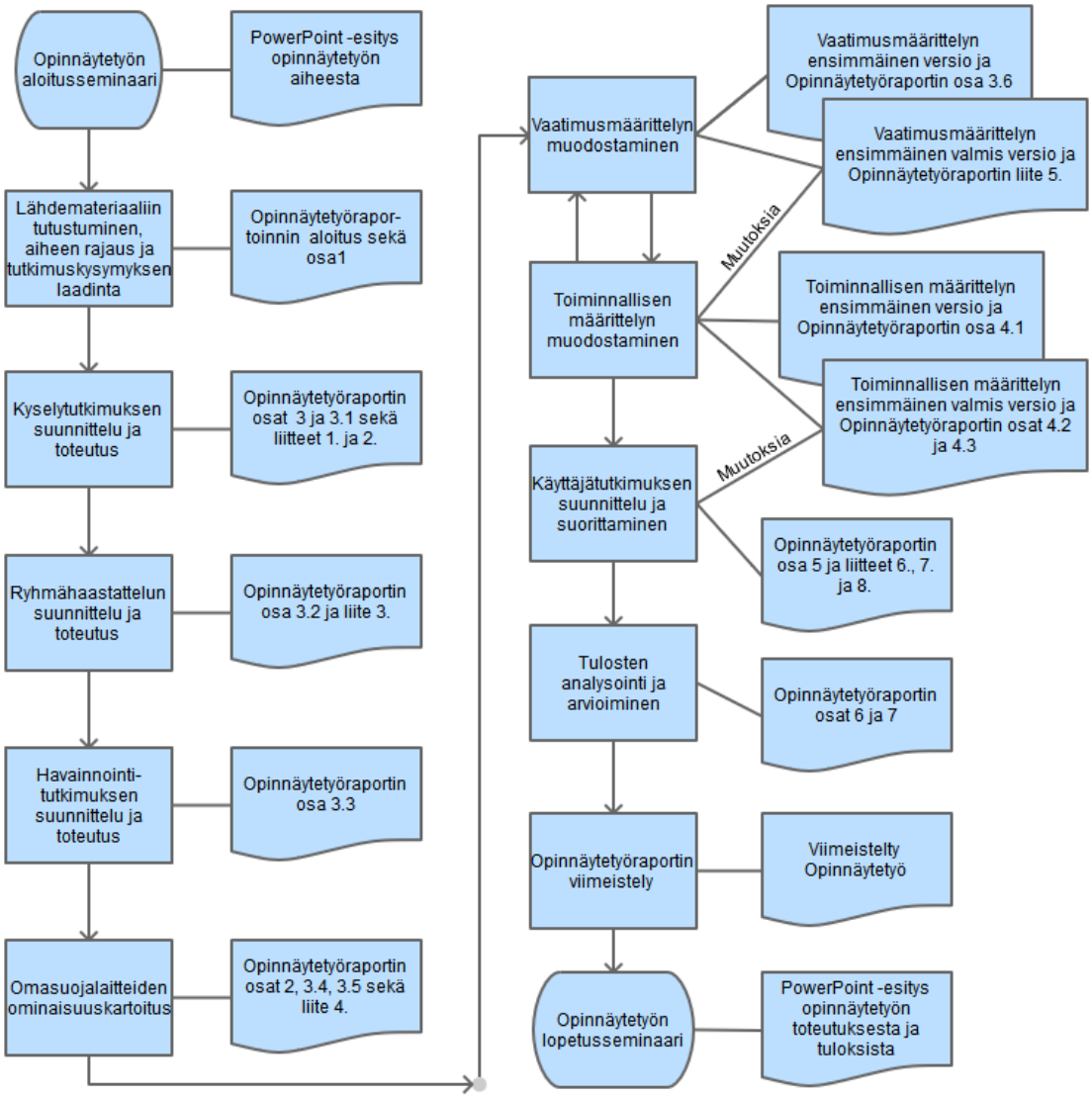
1.3 Opinnäytetyön raportointi

Tässä raportissa käytetään ns. synkronista raportointitapaa, jonka mukaisesti teoreettinen tietämys esitetään yhdessä kyseiseen aiheeseen liittyvien tutkimuskuvausten ja saatujen tulosten kanssa. Tällä pyritään parantamaan raportin luettavuutta sekä yhdistämään teoriatieto ja käytännön toteutus tiiviiksi kokonaisuudeksi. Koska työ on raportoitu toteuttamisjärjestyksessä, aiheen käsittelyjärjestys saattaa poiketa oletetusta.

Tässä työssä tuotettu vaatimusmäärittely on julkinen asiakirja, mutta toiminnallinen määrittely on luottamuksellinen. Kumpaakaan määrittelyä ei ole liitetty sellaisenaan tämän työn liitteeksi, mutta Vaatimusmäärittelyn sisällysluettelo on tämän työn liitteenä 5. Niinpä joidenkin vaatimusten sekä toiminnallisten ratkaisujen esittely itse raportissa helpottaa sellaista lukijaa, jolla ei ole mahdollisuutta lukea itse vaatimusmäärittelyä tai toiminnallista määrittelyä. Toisaalta käyttöliittymäkuvien ja

muiden toiminnallisen määrittelyn yksityiskohtien esittelyä on luottamuksellisuussyistä jouduttu rajoittamaan.

Seuraavassa kuvassa 2. on esitetty kaaviona koko opinnäytetyöprosessi. Tämä kuvaus voi osaltaan havainnollistaa myös tämän raportin käsittelyjärjestystä.



Kuva 2. Opinnäytetyön kokonaisprosessi.

Kuvassa 2. nuolilla osoitetaan siirtymistä opinnäytetyön toteutusvaiheesta toiseen. Suunnattomilla yhdysviivoilla kuvataan kulloiseenkin toteutusvaiheeseen liittyvät tuotteet tai opinnäytetyöraportin osat, jotka ovat syntyneet pääasiallisesti kyseisen vaiheen työskentelyn tuloksena. Lisäksi kuvaan on merkitty aloitus ja lopetusseminaarit, jotka eivät sisälly opinnäytetyöraporttiin, mutta ovat tärkeä osa opinnäytteen kokonaisprosessia.

2 Omasuojaheitin

Erityisesti sotilaalliseen tarkoitukseen käytettävien kulkuvälineiden ja laitteiden suojaamiseen on jo Toisesta maailmansodasta lähtien käytetty monia erilaisia ns. elektronisen sodankäynnin menetelmiä. Elektronisella sodankäynnillä tarkoitetaan pääasiallisesti toimintaa, jossa pyritään kiistämään vastustajan mahdollisuudet sähkömagneettisen spektrin hyödyntämiseen, samalla kun mahdollistetaan sen käyttö omiin tarkoituksiin. (Stimson 2014. 509.) Yksittäisen kulkuvälineen suojaamiseen tarkoitettuja elektronisen sodankäynnin menetelmiä käyttäviä laitteita kutsutaan yleensä omasuojalaitteiksi. Omasuojalaitteiden tarkoituksena on suojata omaa kulkuvälinettä vastustajan sähkömagneettista spektriä hyödyntäviltä laitteilta ja niiden aiheuttamilta vaikutuksilta. Esimerkiksi lentokoneeseen asennetun tutkahäirintälähettimen tarkoituksena on harhauttaa vastustajan tulenjohtotutkaa niin, että vastustajan mahdollisuus käyttää tutkan toimintaan perustuvaa asevaikutusta estyy tai vaikeutuu.

Myös omasuojaheitin on osa laajaa mm. lentokoneiden suojaamiseen tarkoitettua omasuojavälineiden valikoimaa. Omasuojaheitin on laite, joka heittää tai vapauttaa ilmaan kertakäyttöisiä heitteitä sen osana olevista makasiineista, joihin yksittäiset omasuojaheitteet on asennettu. Perinteisesti heittäminen tapahtuu sytyttämällä sähköimpulssilla heittopanos, joka on pieni räjähde. Heittopanoksen räjähdys pakottaa varsinaisen heitteen suurella nopeudella ulos makasiinista. Myös muita tapoja heitteiden heittämiseen on käytössä. Omasuojaheittimen mallista sekä tehdyistä valinnoista riippuen sen makasiineihin voidaan asentaa eri määriä ja keskenään erilaisia heitetyyppejä ja heitemalleja. Omasuojaheittimen tehtävänä on pitää kirjaa siihen ladatuista ja sillä edelleen käytettävissä olevista heitteistä ja raportoida se lentokoneen muille järjestelmille, sekä heittää heitteet täsmälleen halutulla tavalla ja ajan hetkellä.

2.1 Omasuojaheittimien toiminta

Jotta omasuojaheitin pystyisi suorittamaan kaikki tehtävänsä, on niissä yleensä kaksi erillistä ohjelmistoa. Ensimmäinen ohjelmisto on tavallaan laitteiston oma käyttöjärjestelmä, OFP (Operational Flight Programme). OFP hallinnoi itse laitteiston

sisäistä toimintaa ja sitä ei voi tyypillisesti muuttaa käyttäjän toimesta, vaan laitteen valmistaja tai toimittaja vastaa OFP:n ohjelmoimisesta. Toinen ohjelmisto UDF (User data File tai joskus myös MDF Mission Data File) on ikään kuin ajettava sovellus, joka toteuttaa haluttuja toiminnallisuuksia ja on yleensä vapaasti käyttäjän muokattavissa. UDF:ään ohjelmoitava tietosisältö on useimmiten luottamuksellista, eikä sitä siten voi tuottaa kukaan muu kuin laitteiston käyttäjä. Tämä jako kahteen erilliseen ohjelmiston osaan, OFP:hen ja UDF:ään, on hyvin tyypillinen myös muissa omasuojalaitteissa ja UDF:stä käytetään Suomessa yleensä nimitystä ohjaustiedosto.

Omasuojaheittimeen ladattavalla ohjaustiedostolla kerrotaan omasuojaheittimelle täsmällisesti, kuinka sen halutaan toimivan kussakin ennalta määritellyssä tilanteessa. Tällaisia etukäteen määriteltäviä ominaisuuksia ovat esimerkiksi mitä heitteitä, millä ajanhetkillä ja mihin suuntaan heittimen halutaan heittävän, kun lentokonetta uhkaa tietty asejärjestelmä, tietystä suunnasta ja oman koneen korkeus on alle kyseisen uhkajärjestelmän maksimiampumakorkeuden. Kaikki nämä edellä mainitut ominaisuudet on suunniteltava etukäteen ja ohjelmoitava tarkasti ohjaustiedostoon, jotta omasuojaheitin tuottaisi halutun toiminnallisuuden. Ilman tällaista etukäteen tehtyä ohjelmointia ja ohjaustiedoston lataamista itse omasuojaheittimeen, se ei välttämättä tuota mitään toiminnallisuuksia, eikä heitteiden heittäminen ole siis mahdollista. Omasuojaheittimen ohjaustiedoston muodostamiseen ja heiteohjelmiin liittyvistä valinnoista kerrotaan tarkemmin kappaleessa 3.4, jossa käydään läpi erilaisten omasuojaheittimien ominaisuuksia.

2.2 Erilaisten omasuojaheittimien ominaisuuksia

Kuvassa 3. nähdään erään tyypillisen lentokoneissa käytettävän omasuojaheittimen perusosat. Kuvassa ylinnä on heitemakasiini, johon yksittäiset omasuojaheitteet ladataan. Keskellä on varmistin, jonka avulla voidaan estää sähkönsyöttö heitemakasiinille tai koko heittimelle ja siten estää heitelaukaisu esimerkiksi maassa. Alinna on ohjaamoon sijoitettava ohjausyksikkö, joka pitää kirjaa käytettävissä olevista heitteistä ja näyttää heitetilanteen ohjaajalle, pitää muistissaan tiedot ohjelmoiduista heiteohjelmista ja antaa heiteohjelmien mukaiset sähköiset heittokäskyt tai impulssit heitemakasiinille. Eri omasuojaheittimet koostuvat luonnollisesti erilaisista osista ja lisäksi niissä saattaa olla mm. erillinen osa

heittoimpulssien muodostamista varten, erillinen ohjaustietokone jne. Heitemakasiinien määrä riippuu lavetista, jota on tarkoitus suojata, sekä omasuojaheittimen sisäisistä rajoitteista. Hävittäjäluokan lentokoneissa melko tyypillinen makasiinimäärä on neljä, kuvan 3. mukaista, 30 heitepaikan makasiinia.

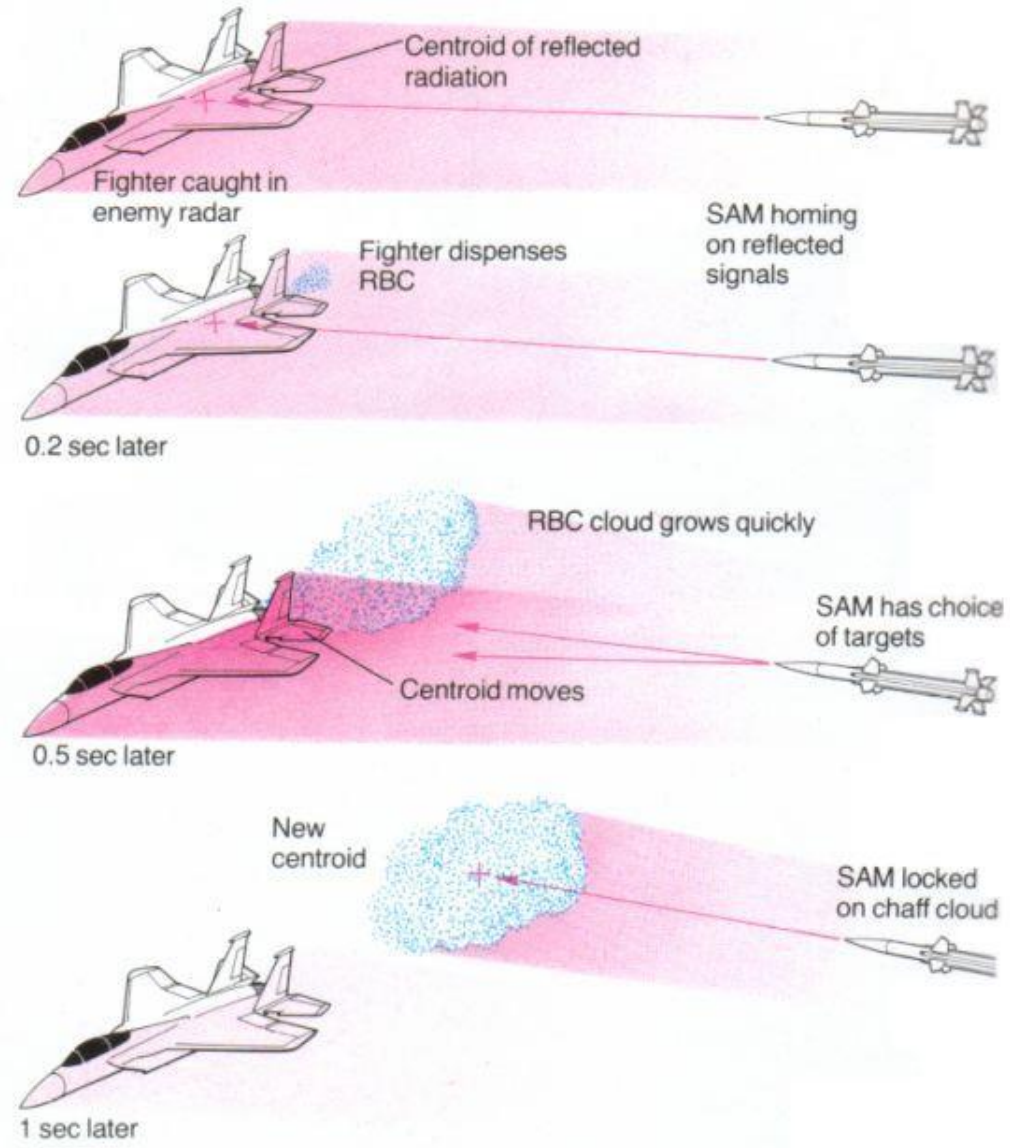
Kuva 3. VICON 78 omasuojaheitin. (VICON 78 Series 455 Countermeasures Dispensing Systems, 1)

Kuvassa 3. näkyvä makasiini on tarkoitettu NATOn standardiheitteille, joiden koko on 1 x 1 x 8 tuumaa, missä 8 tuumaa tarkoittaa heitteen pituutta. Perinteisimpiä omasuojaheitetyyppejä ovat tutkasignaalia heijastavat silppuheitteet sekä infrapuna- eli lämpösäteilyä aktiivisesti tuottavat soihdut. Lisäksi on olemassa myös joukko muita heitetyyppejä, erilaisiin käyttötarkoituksiin. Omasuojaheittimellä on kuitenkin aina tarkoitus suojata omaa lavettia vastustajan asejärjestelmän vaikutukselta. Erilaisten omasuojaheittimien ominaisuuksia on listattu liitteessä 4.

2.3 Silppuheitteet

Silpun käyttö tutkahavaintojen estämiseen sekä tutkien harhauttamiseen keksittiin jo 1940 -luvun alussa. Toisessa maailmansodassa mm. ennen Normandian maihinnousua kuljetuslentokoneista levitettiin valtavia määriä alumiinifoliosta leikattua silppua Englannin kanaalin ylle, estämään saksalaisia näkemästä tutkillaan lähestyviä lentokoneita, joilla maihinnousu aloitettiin. (Chaff (countermeasure) 2015.) Vapautettaessa ilmaan, silppukuidut erkaantuvat toisistaan ja leviävät satunnaisesti. Kun silpun pituus on puolet siihen kohdistetun sähkömagneettisen säteilyn aallonpituudesta ja säteily kohdistuu silppuun poikkisuoraan, se alkaa toimia kuten puoliaaltodipoli. Tällöin silppu alkaa resonoimaan ja lähettää uudelleen vastaanottamansa säteilyn, samalla aallonpituudella. Tällöin todennäköisesti osa silpun lähettämästä säteilystä kohdistuu myös alkuperäisen säteilyn lähettäneeseen tutkaan. Jos silpusta tutkaan palautuvan signaalin voimakkuus on riittävä, se pystyy peittämään takanaan olevat oikeat maalit tai houkuttelemaan tutkan seuraamaan silppupilveä varsinaisen kohteen sijasta. (Stimson 2014. 558.)

Seuraavassa kuvassa 4. on esitys silpun toimintaperiaatteesta. Kuvassa lentokonetta kohti on laukaistu ohjus, joka hakeutuu lentokoneesta heijastuvaan tutkasäteilyyn. Lentokone heittää silppuheitteen, joka avautuu nopeasti ilmavirrassa muodostaen lentokoneen heijastuspinta-alaa suuremman heijastuspinta-alan. Tällöin ohjus siirtyykin hakeutumaan kohti silppupilveä, josta se saa suuremman heijastuspinta-alan ansiosta voimakkaamman signaalin. Näin ohjus hakeutuu lentokoneen sijasta kohti silppupilveä, eikä ohjus pysty vaikuttamaan lentokoneeseen.



Kuva 4. Silppuheitteen toimintaperiaate. (Jackson, D. 2004)

Myös nykyaikainen silppu perustuu tyypillisesti alumiinin käyttöön, mutta se on useimmiten tarkasti tiettyihin pituuksiin katkottua erittäin ohutta lasikuitua, jonka päälle on höyrystetty ainoastaan muutaman atomikerroksen vahvuinen alumiinipinnoite. Tällöin erittäin pieneen tilaan saadaan mahtumaan valtava määrä näitä silppukuituja ja jo yksittäisillä heitteillä saadaan aikaan voimakas ns. tutkaheijaste. Koska kuitujen pituus määrittää sen taajuusalueen, jolla silppu aiheuttaa voimakkaimman tutkaheijasteen, silpun pituudet määritellään valittujen uhkajärjestelmien tutkan taajuuden perusteella.Logistisesti on järkevää rajoittaa erilaisten heitteiden lukumäärää, joten samalla silppuheitteellä pyritään harhauttamaan useampia eri tutkia, jotka toimivat keskenään eri taajuudella. Niinpä samaan silppuheitteeseen pakataan tyypillisesti useampaa eripituista kuitua, kuten

kuvassa 5. Tällöin yhdellä heitetyypillä saadaan riittävän voimakas tutkaheijaste useammalle eri tutkatyypille.

Kuva 5. Silppupatruunoita, joissa eri silppupituuksia. (US Navy Naval Research Laboratory, 2012)

2.4 Soihtuheitteet

Soihtuja käytetään erityisesti harhauttamaan suojattavaa lavettia kohti ammuttuja infrapunasäteilyn havaitsemiseen perustuvia ohjuksia. Kun soihtu heitetään ilmaan, se joko sytytetään palamaan tai se alkaa muulla tavoin säteilemään voimakasta infrapunasäteilyä halutulla taajuusalueella. Tällä yritetään aiheuttaa oman lavetin infrapunaherätettä voimakkaampi, tai muuten sopivampi, heräte ohjuksen hakupäälle ja harhauttaa ohjuksen seuranta pois kohteesta. Tämän saavuttamiseksi myös soihdun ominaisuuksia pyritään säätämään niin, että sen tuottama säteily olisi aallonpituudeltaan ja intensiteetiltään, sekä intensiteetin voimistumisnopeudeltaan, kestoltaan ja muilta ominaisuuksiltaan, halutun kaltainen. (Pollock 1993, 289-295.) Käyttötilanne ja -olosuhteet, kuten esimerkiksi lentokoneesta heitettäessä lentonopeus ja heittokorkeus, vaikuttavat voimakkaasti heitteen tuottaman säteilyn

intensiteettiin ja lentorataan (Pollock 1993, 305). Nämä taas vaikuttavat merkittävästi heitteillä saavutettavaan vaikutukseen, joten erilaisiin tilanteisiin on suunniteltava erilaisia heiteohjelmia, halutun vaikutuksen aikaansaamiseksi. Esimerkiksi nopeammin lennettäessä yksittäisen heitteen tuottaman säteilyn intensiteetti laskee, joten heitteitä pitää heittää enemmän, saman kokonaisintensiteetin saavuttamiseksi.

Perinteisin soihtutyyppi on ns. MTV-soihtu, jonka käyttöä nähdään kuvassa 6. MTV – soihtu on yhtenäiseksi puristettua magnesiumia, teflonia ja vitonia sisältävää massaa, joka palaa erittäin kuumana ja näkyy hyvin myös paljaalle silmälle. Tämä saattaa paljastaa lentokoneen esimerkiksi yöllä, joten silloin saatetaan haluta heittää soihtuja, jotka eivät paljasta konetta pimeässä. Toisaalta helposti nähtävällä heitteellä voidaan yrittää vaikuttaa ohjusta käyttävään ihmiseen ja estää häntä laukaisemasta ohjusta.

Kuva 6. F-18 monitoimihävittäjä heittää MTV -soihtuja omasuojaheittimellä. (ALE-47 Airborne Countermeasures Dispenser System)

3 Vaatimusmäärittelyn muodostaminen

Työn ensimmäisenä tavoitteena oli selvittää ohjelman tarpeellisuus ja muodostaa sille vaatimusmäärittely. Jotta ohjelmisto voitaisiin määritellä, täytyy sille olla

olemassa perusteet. Näitä perusteita ovat esimerkiksi ohjelmiston käyttötarkoitus ja tulos johon sen käytöllä pyritään. Lisäksi tuotteen tulevilla käyttäjillä on sen toimintaan ja käyttöön liittyviä vaatimuksia, joiden pitää täyttyä jotta tuotteelle olisi kysyntää. Näiden erilaisten tuotteeseen kohdistuvien odotusten kokoamista ja jäsentämistä toteuttamiskelpoisiksi vaatimuksiksi, kutsutaan vaatimusmäärittelyksi. Ohjelmistoprojektien epäonnistumiset johtuvat yli 60-prosenttisesti huonosti hoidetusta vaatimustenkäsittelystä, joten tähän prosessiin panostaminen on yksi onnistuneen ohjelmistoprojektin perusedellytyksiä (Haikala 2011, 61).

Vaatimusmäärittelyssä selvitetään vähintään tuotteen tarpeellisuus ja toteuttamiskelpoisuus sekä asetetaan tuotteelle tavoitteet ja vaatimukset. Edellisten lisäksi muodostetaan myös käsitys siitä, millainen tuote täyttäisi nämä asetetut vaatimukset. Usein tämä kokonaisuus jaetaan kahteen osaan, joista ensimmäisessä kerätään asiakkaiden tuotteelle asettamat vaatimukset ja toisessa määritellään tuote joka täyttää nämä vaatimukset. Tärkein vaatimusmäärittelystä syntyvä tuote on toiminnallinen määrittely, joka on kuvaus tuotteelle asetetuista tavoitteista ja vaatimuksista sekä siitä, miten tuotettava tuote ne täyttää. (Mts. 78.)

Jotta vaatimusmäärittely kyettiin muodostamaan osana tätä työtä, piti käyttäjien odotukset ja erilliset vaatimukset tuotetta kohtaan selvittää. Näitä varten perehdyttiin käyttäjätutkimuksen menetelmiin ja tutkimuksen tekemisen periaatteisiin. Käyttäjien vaatimusten ja todellisten tarpeiden tarkka selvittäminen ennen vaatimusmäärittelyn tuottamista nähtiin opinnäytetyössä erittäin tärkeäksi. Niinpä vaatimusten kokoamiseksi päätettiin suorittaa neljä erillistä tutkimusta, ennen saatujen tulosten yhdistämistä vaatimusmäärittelyksi. Työ aloitettiin kyselyllä, jonka jälkeen vastaajille suoritettiin ryhmähaastattelu ja lopulta havainnoitiin nykyisen omasuojaheittimen ohjelmointityökalun käyttäjän työskentelyä. Tuloksia täydennettiin vielä perehtymällä erilaisiin julkisiin lähdemateriaaleihin, lähinnä erilaisten omasuojaheittimien teknisten ominaisuuksien selvittämiseksi.

3.1 Kyselytutkimus

Lyhyen asiaan perehtymisen perusteella muodostettiin käsitys, että vastaavaa omasuojaheittimien ohjelmoinnin suunnitteluun tarkoitettua apuvälinettä ei ole

ennestään saatavilla. Myöskään tietoa tämänkaltaisen tuotteen tarpeellisuudesta ja hyödystä omasuojaheittimien ohjelmoijille ei ollut saatavilla, joten ensimmäinen tehtävä oli näiden selvittäminen. Tämän tiedon saamiseksi toteutettiin käyttäjäkysely omasuojaheittimien ohjelmointiin perehtyneille ihmisille. Samalla kyselyllä selvitettiin myös tavoitteita, joita käyttäjät haluaisivat tuotteen avulla toteuttaa sekä vaatimuksia, jotka tuotteen tulisi vähintäänkin täyttää.

Koska tavoitteena ei ollut muodostaa uutta tieteellistä tietämystä, vaan tuottaa vastauksia juuri tähän määriteltävään ohjelmistotuotteeseen liittyvistä, käyttäjien toiveista ja erityisistä vaatimuksista, kysely toimi erityisen hyvänä tapana kerätä mahdollisimman paljon lähtötietoa aiheesta. ”Käyttäjät voivat kyselyjen avulla kertoa omista toiveistaan ja tulevista tarpeistaan mutta myös pettymyksistään.” (Saariluoma 2004, 42-43) Avoimessa kyselytutkimuksessa vastaajilta kysytään kysymyksiä, joihin he voivat vastata omasanaisesti. Tällöin saadaan hyvin monipuolisesti tietoa tutkittavasta aiheesta. Kyselytutkimuksessa on kuitenkin myös monia riskejä. Kyselyn kysymykset ja sanavalinnat voivat olla harhaanjohtavia, vaikeasti ymmärrettäviä tai piilevästi asenteellisia. Tällöin on riskinä, etteivät tutkimuksen tekijät saa vastauksia juuri haluamiinsa kysymyksiin tai kysymyksenasettelu vääristää vastauksia. (Saariluoma 2004, 43-44.)

3.1.1 Kyselyn muodostaminen

Käyttäjien ohjelmistolle asettamien käytännön vaatimusten ennakointi oli hyvin hankalaa ja toisaalta pyrittiin välttämään liiallisten ennakkoasenteiden ohjaavaa vaikutusta. Niinpä kyselytutkimuksen kysymykset päätettiin jättää melko väljiksi ja antaa vastaajille mahdollisuus kertoa omista toiveistaan ja käyttää omaa mielikuvitustaan. Lisäksi joissakin kysymyksissä toistettiin samaa teemaa hieman eri sanoin, jotta vastaajan ymmärrys kysymyksenasettelusta ei estäisi mahdollisimman monipuolisten vastausten saamista. Käyttäjille esitetyt kysymykset on esitetty seuraavaksi Taulukossa 1.

1) Mikä voisi olla työkalun tuottama tuote?
2) Mitä nykyisten ohjelmien ongelmia tällainen ohjelmisto voisi korjata?
3) Mitä muita asioita työkalulla pitäisi voida tehdä?
4) Mitä työkalulla pitää vähintäänkin pystyä tekemään, jotta siitä olisi sinulle hyötyä?
5) Mitä vaatimuksia asettaisit työkalulle?
6) Mikä on mielestäsi kriittisin hyöty mikä kyseisestä työkalusta olisi saatavissa?
7) Lisäisikö vai vähentäisikö työkalu työmääräsi uusien ohjaustiedostojen tuottamisessa?
8) Parantaisiko työkalu luottamustasi ohjelmoinnin onnistumiseen?
9) Millaisia raportteja työkalun pitäisi tuottaa?
10) Miten uskoisit ohjelman käyttöönoton vaikuttavan heiteohjelmien tuottamiseen?
11) Miten voisit hyödyntää työkalua työssäsi?
12) Parantaisiko työkalu työsi lopputulosta?
13) Onko yleiselle heiteohjelmistojen suunnittelutyökalulle omasta mielestäsi tarvetta?

Taulukko 1. Käyttäjille esitetyt kysymykset.

Jotta koko ohjelmistomäärittelyn toteuttamisen mielekkyyttä voitiin arvioida, kysyttiin käyttäjien mielipiteitä koko ohjelmiston tarpeellisuudesta ja hyödyllisyydestä heidän työssään. Tätä selvitettiin kysymyksillä 7, 8, 12 ja 13, jotta asiaa voitiin tarkastella useammasta eri näkökulmasta ja lisätä vastausten luotettavuutta. Varsinaisten ohjelmistolle asetettavien käyttäjävaatimusten selvittämiseksi kysyttiin kysymykset 3, 4 ja 5. Koska koko ohjelmiston tarpeellisuus perustuu nykyisin käytössä olevien ohjelmistojen käytettävyyden heikkouksiin, kysyttiin kysymys 2, jolla selvitettiin nykyisten ohjelmistojen käytössä havaittuja ongelmia. Koska erilaiset raportit ovat hyvin yleinen lopputuote suunnittelutyökaluille, yritettiin käyttäjät saada ajattelemaan asiaa erilaisesta näkökulmasta, ilman perinteiden ohjaavaa vaikutusta. Tämän vuoksi kysymyksillä 1 ja

3 kysyttiin vielä erikseen muodostettavan työkalun tuottamista lopputuotteista. Jotta raportteja mahdollisina lopputuotteina ei kuitenkaan kokonaan unohdettaisi, kysyttiin niiden mahdollista muotoa kysymyksellä 9. Käyttäjille ohjelmistosta muodostuvien mielikuvien selvittämiseksi kysyttiin kysymykset 6 ja 11. Näillä haluttiin saada selville, ovatko eri käyttäjille muodostuvat mielikuvat hyvin erilaisia ja toisaalta, miten kukin käyttäjä henkilökohtaisesti odottaa ohjelmiston vaikuttavan hänen työskentelyynsä.

3.1.2 Kysymyksenasettelun varmistaminen

Koska kysymysten muodostamisessa ei käytetty mitään valmista pohjaa, päätettiin sopiva kysymysasettelu varmistaa vielä erikseen. Tämä tehtiin vertaamalla muodostettuja kysymyksiä Ulrichin ja Eppingerin muodostamaan listaan käyttäjän tarpeiden tunnistamisen kokonaisuuden päämääristä. Tämä lista on esitetty alla vapaasti opinnäytetyön tekijän suomentamana.

a. Varmistaa tuotteen kohdistuminen tarpeeseen.
b. Varmistaa että myös piilevät tarpeet on tunnistettu.
c. Kerätä kaikki faktat ja tallentaa niiden hankintamenettelyt.
d. Varmistaa ettei mitään kriittistä tarvetta ole unohdettu.
e. Kehittää yhteisesti jaettu ymmärrys tarpeista.

Taulukko 2. Käyttäjän tarpeiden tunnistaminen Ulrichin ja Eppingerin (2000, 60) mukaan. Vapaasti lähdemateriaalista suomennettu.

Verrattaessa muodostettuja kysymyksiä Ulrichin ja Eppingerin (2000, 60) muodostamaan listaan, havaittiin että kohtaan a. vastataan kyselyssä useillakin kysymyksillä. Niinpä vastaus ohjelman yleiseen tarpeellisuuteen oli varmistettu. Kohdan b. mukaisia käyttäjien piileviä tarpeita ei voida todennäköisesti luotettavasti selvittää kyselytutkimuksella. Käyttäjien piilevät tarpeet tulevat esiin todennäköisesti ainoastaan todellista käyttötilannetta tai käyttösimulointia tarkastelemalla tai haastattelututkimuksen yhteydessä. Niinpä näiden piilevien käyttäjätarpeiden selvittäminen korostuikin toteutettaessa ryhmähaastattelua sekä havainnointia. Jotta kohdan c. mukaisesti kaikki faktat tulivat kerätyiksi ja tallennetuiksi, koostettiin

kyselyn vastaukset yhteen luetteloon. Myös myöhemmissä tutkimusvaiheissa saadut vastaukset koostettiin, samalla tavalla, tutkimuskohtaisesti. Koska kattavaa ennakkokäsitystä ohjelmiston käyttäjien kriittisistä tarpeista ei pystytty muodostamaan, kohdan d. mukaisten kriittisten tarpeiden tunnistaminen ei ollut mahdollista, ilman kaikkien eri tutkimuksissa kerättyjen tarpeiden läpikäyntiä. Niinpä tämä, kohdan d. mukainen varmistaminen, tapahtuikin vasta varsinaista vaatimusmäärittelyä muodostettaessa.

Koska kyseessä ei ollut asiakasprojekti, ei nykyisiä käyttäjiä voitu sitouttaa tähän työhön, eikä heidän työaikaansa käyttää enempää kuin mitä erilliset lyhyet tutkimukset edellyttivät. Niinpä tätä työtä ei tehty varsinaisesti yhteistyössä käyttäjien kanssa, kuten Ulrich ja Eppinger esittävät, joten taulukon 2. kohdassa e. esitettyä jaettua ymmärrystä ei voinut muodostua. Tämän jaetun ymmärryksen muodostaminen on kuitenkin olennaista, mikäli tämän työn tuloksena toteutetaan asiakasprojekti varsinaisen ohjelmiston tuottamiseksi. Ilman tuota jaettua ymmärrystä, ei tämän kaltaista ohjelmaa voi saada vastaamaan riittävällä tasolla käyttäjien todellisiin tarpeisiin, eikä se siten voi lunastaa sille asetettuja tavoitteita.

3.1.3 Kyselyn toteutus ja tulokset

Kun lopullinen kysymyslista oli muodostettu, tehtiin kyselystä erillinen yhdeksän kuvan diaesitys, jossa käytiin läpi kyselyn taustat sekä perusasioita mahdollisesta ohjelmistosta. Varsinaiset kysymykset sijoitettiin erilliseen Word -asiakirjaan, johon vastaajien toivottiin kirjoittavan vastauksensa. Pelkästään kyselyn ymmärrettävyyden ja yleisen toimivuuden varmistamiseksi, se testattiin ennakkoon. Kyselyn diaesitys sekä kyselykaavake annettiin omasuojalaitteita tuntemattoman humanistin tarkasteltavaksi ja sitä muokattiin hänen antamansa välittömän palautteen perusteella.

Korjatut diaesitys sekä kyselykaavake lähetettiin sähköpostitse, saatesanojen kera, yhteensä seitsemälle ihmiselle, joita pyydettiin vastaamaan kyselyyn. Tämä diaesitys yhdessä saatesanojen kanssa on esitetty liitteessä 1. Kysely lähetettiin ainoastaan henkilöille, joilla tiedettiin olevan ainakin muutaman vuoden mittainen, suhteellisen viimeaikainen ja omakohtainen kokemus omasuojalaitteiden ohjelmoinnista. Kokemus juuri omasuojaheittimien ohjelmoinnista vaihteli vähäisestä, yli kymmenen

vuoden jatkuvaan työskentelyyn omasuojaheittimien ohjelmoinnin ja heitteiden parissa. Tällä rajauksella pyrittiin varmistamaan, että vastaajat edustavat hyvin ohjelman tulevia loppukäyttäjiä ja esitettyihin kysymyksiin saadaan mahdollisimman relevantteja vastauksia.

Ikävä kyllä vastauksia saatiin lopulta vain kaksi. Onneksi molemmilla vastaajilla oli vahva viimeaikainen ja omakohtainen kokemus omasuojaheittimien ohjelmoinnista, joten tulokset olivat hyvin käyttökelpoisia. Kyselyllä saadut suorat vastaukset ovat nähtävissä liitteessä 2. Yksittäisiä vastauksia jouduttiin luottamuksellisuussyistä hieman muokkaamaan, mutta vastausten asiasisältö on säilytetty. Vastausten pohjalta muodostettiin myös alla näkyvä Taulukko 3, jossa saadut vastaukset avataan selväsanaisiksi ja samoja sisältöjä koskeneet vastaukset on yhdistetty. Osa saaduista vastauksista oli epäselviä ja niitä haluttiin tarkentaa ryhmähaastattelussa, eivätkä ne siksi ole mukana taulukossa.

Arvioita ohjelman yleisestä hyödyllisyydestä ja käyttötavoista

- Ohjelman uskottiin parantavan omasuojaheittimen ohjelmoinnin lopputulosta ja ohjaustiedostojen laatua sekä lisäävän käyttäjän luottamusta ohjelmoinnin onnistumiseen.
- Tällaiselle työkalulle koettiin olevan tarvetta, mikäli se on käytettävyydeltään erityisen hyvä ja tuottaa lisäarvoa käyttäjän työlle.
- Tällaista työkalua haluttaisiin käyttää omasuojaheittimen ohjaustiedostojen suunnittelussa, toteutuksessa ja dokumentoinnissa.
- Ohjelman kriittisimmiksi hyödyiksi uskottiin ohjelmoinnin helpottuminen ja siirtyminen yksityiskohtien hiomisesta kohti kokonaisuuden hallintaa.
- Arviot ohjelman vaikutuksista käyttäjän kokonaistyömäärään olivat täysin ristiriitaisia.
- Ohjelmaa haluttaisiin käyttää tuotettavan heitinohjelmiston testipisteiden määrittämiseen.
- Ohjelman tuottamia raportteja haluttaisiin käyttää myös omasuojaheittimen toiminnasta, sen käyttäjille muodostettavan kuvauksen tai ohjeistuksen muodostamiseen. Erityisesti toivotaan havainnollistavia kuvia MDF:n toiminnasta.
- Ohjelman haluttaisiin olevan suoraan visuaalisempi käyttöliittymä varsinaiselle MDF -generaattoriohjelmistolle.

Ohjelmalle asetettavia vaatimuksia

- Ohjelman haluttaisiin tuottavan erilaisia yhteenvetoja suunnitelluista ohjaustiedostoista ja erityisesti siihen ohjelmoiduista uhkista tai uhkaluokista.
- Ohjelmalta vaaditaan mahdollisimman havainnollista, graafista käyttöliittymää joka visuaalisuudellaan helpottaa ja nopeuttaa heiteohjelmistojen suunnittelua sekä samalla parantaa niiden laatua.
- Ohjelman on oltava helppokäyttöinen, intuitiivinen ja omata korkea luotettavuus.
- Käyttöohjeiden pitää olla ohjelmassa helposti saatavilla.
- Ohjelmiston pitää olla yhteensopiva muiden ohjelmien kanssa.
- Ohjelmiston pitää mahdollistaa seuranta syistä, jotka ovat johtaneet tehtyihin ratkaisuihin. Esimerkiksi miten muodostettuihin heiteohjelmiin on päädytty.
- Aiemmin luotuja kokonaisuuksia, kuten yksittäisiä heiteohjelmia, on voitava hyödyntää uudelleen uusissa tuotteissa tai saman tuotteen sisällä. Tämän tekemiseksi eri komponentit on tallennettava tietokantaan, jossa niitä voidaan hallinnoida.
- Ohjelmiston pitää muistaa käyttäjän tekemät valinnat esim. hakemistojen suhteen.

Taulukko 3. Kyselytutkimuksen tulokset

Vastauksissa korostuivat toistuvasti toiveet ohjelman helppokäyttöisyydestä ja käyttöliittymän runsaasta visuaalisuudesta. Ohjelman haluttaisiin toimivan suoraan helppokäyttöisempänä käyttöliittymänä MDF –generaattoriohjelmistolle ja mahdollistavan testipisteiden muodostamisen tuotettavalle ohjaustiedostolle. Myös mahdollisuus tuottaa kuvia heiteohjelmista käyttäjäohjeistusta varten, koettiin tärkeäksi. Saatujen vastausten pohjalta muodostettiin ennakkokäsitys asioista ja kysymyksistä, joihin etsittiin vastauksia ryhmähaastattelussa. Muodostettua vastausluetteloa käytettiin olennaisena osana varsinaisen vaatimusmäärittelyn muodostamisessa.

3.2 Ryhmähaastattelu

Jotta kyselytutkimuksella saatuja vastauksia voitaisiin ymmärtää paremmin ja varmistaa, että kaikki olennaiset käyttäjävaatimukset saataisiin kerättyä jo alkuvaiheessa, suoritettiin seuraavaksi ryhmähaastattelu. Ryhmähaastatteluun osallistui samoja henkilöitä, joille kohdassa 3.1 toteutettu kyselytutkimus oli osoitettu. Koska ryhmähaastattelun osallistujat oli ennalta valittu ja heillä oli tiettyä ennalta määritettyä osaamista, oli kyseessä fokusryhmähaastattelu. Tässä osassa käytetään kuitenkin lyhyempää nimitystä ryhmähaastattelu.

Haastattelussa tutkija tapaa haastateltavan ja yrittää keskustelun ja kysymysten avulla selvittää haastateltavan näkemyksiä, haastattelijan valitsemasta aiheesta. Strukturoimattomassa haastattelussa haastattelija ei ole etukäteen tarkasti määritellyt haastateltavalle esitettäviä kysymyksiä tai haastattelutilannetta, vaan haastateltava voi itse valita käsiteltävien asioiden painotuksia. Haastattelulla saadaan samoja tietoja kuin kyselyllä, mutta haastattelussa puheena olevaa asiaa voidaan tarvittaessa syventää. (Saariluoma 2004, 45-46.)

Haastattelun ongelmana ovat erityisesti työläys sekä saatavien tietojen luotettavuus. Muut haastattelun osallistujat sekä haastattelija voivat vaikuttaa saatuihin tuloksiin, esimerkiksi johdattelemalla tai haastattelija saattaa ymmärtää keskustelun väärin. Lisäksi kaikki saadut tiedot on kirjattava tarkasti ylös ja analysoitava jälkikäteen. Jotta haastattelu tuottaa luotettavaa tietoa, täytyy itse haastattelutilanteen olla luottamuksellinen ja haastateltavan on voitava ilmaista omat näkemyksensä vapaasti ilman haastattelijan vaikutusta. Tärkeintä on haastateltavan kuunteleminen ja rohkaiseminen kertomaan oma näkemyksensä. (Mts. 45-46.)

3.2.1 Ryhmähaastattelun valmistelu

Jotta ryhmähaastattelussa nousisi esiin mahdollisimman monipuolisesti erilaisia näkökulmia ja mielipiteitä, haluttiin se suorittaa strukturoimattomana haastatteluna, jossa osallistujien vapautta ilmaista itseään rajoitettaisiin mahdollisimman vähän. Samalla haluttiin kuitenkin varmistaa, että vaatimusmäärittelyn kannalta olennaisiin asioihin saadaan vastauksia. Niinpä haastattelua varten muodostettiin jäljempänä nähtävän taulukon 4. mukainen haastattelurunko, jonka tarkoitus oli lähinnä toimia

haastattelijan tukena, auttamassa olennaisten vastausten saamisessa. Tarkoitus ei kuitenkaan ollut sitoa haastattelijaa tai haastateltavia juuri kyseisten kysymysten pohtimiseen tai niihin vastaamiseen. Haastattelun alussa ohjelmasta käytettiin nimitystä työkalu, keskittämään osallistujien ajatukset aluksi työhön ja sen tekemiseen. Haastattelussa sanoja työkalu, ohjelmisto ja ohjelma käytettiin synonyymeinä toisilleen. Tätä haastattelurunkoa ei esitelty haastateltaville.

- Miten te näette korkean luotettavuuden? Mitä se tarkoittaa? (esim. hyvä saatavuus, virheetön lopputulos, hyvä toistettavuus yms.)
- Millainen olisi ”Raportti halutun heitesekvenssin määrittelyn siirtymisestä”?
- Millä tavalla työkalun pitäisi ohjata käyttäjän toimintaa?
- Pitäisikö työkalun antaa jatkokehitysehdotuksia ja millaisia ne voisivat olla?
- Lisäisikö vai vähentäisikö tällainen työkalu kokonaistyömäärääsi ja miksi?
- Minkä ohjelmien kanssa työkalun pitäisi olla yhteensopiva?
- Millä tasolla ja millä tavoin ohjelmistossa pitäisin seurata muutoksia, niiden tekijää ja ajankohtaa?
- Mitkä olisivat suurimmat hyödyt ohjelmiston käyttämisestä.
- Millaisia kokonaisuuksia aiemmin tuotetuista ohjaustiedostoista pitäisi voida uudelleenkäyttää. (Esim. kokonainen heiteohjelma, kokonainen uhka ja kaikki sen vasteet, kokonainen ohjaustiedosto vai jokin muu tai pienempi osa.)
- Haluaisitko nähdä mieluummin yhden uhkan kaikki eri heitesekvenssit kerrallaan yhdellä erillisellä sivulla vai jotakin muuta?
- Millaisiin osiin jakaisit ohjelman? (Esimerkiksi näkymä jossa näkyvät kaikki uhkat ja niiden valmiusaste ja siitä avautuvat erilliset uhkakohtaiset ikkunat.)

Taulukko 4. Ryhmähaastattelun haastattelurunko.

Apuna haastattelurungon muodostamisessa käytettiin kohdan 3.1 kyselytutkimuksesta saatuja vastauksia ja sen pohjalta nousseita uusia kysymyksiä, sekä seuraavassa taulukossa 5. esitettyä kysymyssarjaa. Nämä taulukon kysymykset ovat Cooperin, Reimannin ja Croninin (2007, 56) mukaan, erityisen kiinnostavia kysymyksiä ohjelmistotuotteen loppukäyttäjille. Taulukon viimeinen kohta jätettiin ryhmähaastattelussa pienemmälle painoarvolle, koska se päätettiin ottaa havainnointitutkimuksen erityiseksi havainnointiaiheeksi. Lisäksi taulukon

ensimmäiseen kohtaan oli vastattu riittävällä tasolla jo kyselytutkimuksen yhteydessä, kuten myös jollakin tasolla useimpiin muihinkin kysymyksiin. Niinpä haastattelussa haluttiin tarkentaa joitakin kyselytutkimuksen tuloksia ja keskittyä kysymyksiin kaksi ja neljä.

1. Milloin, miksi ja miten käyttäisitte tuotetta.
2. Mitä käyttäjien pitää tietää tehdäkseen työnsä.
3. Mitä nykyinen työkalu tekee ja mitä sillä ei voi tehdä?
4. Tavoitteet ja motivaatio tuotteen käyttämiseen.
5. Mitä käyttäjät ajattelevat työstään ja tehtävistään kuten myös mitä odotuksia heillä on tuotteesta.
6. Ongelmia ja turhautumista aiheuttavat asiat nykyisessä välineessä.

Taulukko 5. Ohjelmistotuotteen käyttäjälle esitettävät kiinnostavat kysymykset, vapaasti lähdemateriaalista suomennettu (Cooper ym. 2007. 56)

3.2.2 Ryhmähaastattelun toteutus ja tulokset

Ryhmähaastatteluun osallistui kaikkiaan neljä henkilöä, joista jokaiselle aihe oli entuudestaan tuttu. Kokemuspohja omasuojaheittimien ohjelmoinnista vaihteli perustiedoista kokeneeseen ohjelmoijaan. Haastattelutilanteessa tehdyt huomiot kirjattiin kannettavalla tietokoneella. Haastattelutilanteen jälkeen kirjattujen huomioiden asiasisältö laajennettiin muistinvaraiseen tulkintaan ja tutkijan omakohtaiseen kokemukseen pohjautuen. Tässä vaiheessa nopeasti kirjatut huomiot muokattiin myös yksityiskohtaisemmiksi ja helpommin ymmärrettäviksi. Nämä on listattu kokonaisuudessaan liitteessä 3.

Kaikki ryhmähaastattelussa kirjatut havainnot ovat hyvinkin yksityiskohtaisia ohjelman toiminnallisia vaatimuksia. Kuitenkin itse ohjelman perusominaisuuksista käytiin keskustelua hyvinkin periaatteellisella tasolla, mitä ei kirjattu välittömiin havaintoihin. Haastattelijalle muodostui ymmärrys, että käyttäjät eivät suinkaan halunneet ohjelmaan lisää ominaisuuksia ja mahdollisuuksia vaikuttaa itse ohjelman toiminnallisuuteen, vaan oikeastaan päinvastoin. Ohjelman haluttaisiin siis itsessään

olevan mahdollisimman yksinkertainen ja selkeä, eikä siinä haluta olevan mitään, varsinaisen työn lopputuloksen kannalta, ylimääräisiä ominaisuuksia.

Määritettävällä Ohjelmalla ei siis tarvitse pystyä tekemään kaikkea mahdollista, vaan ainoastaan erityisen hyvin ne tehtävät mihin se on tarkoitettu. Ohjelman pitäisi siis, ohjelmistoergonomian periaatteiden mukaisesti, rajoittaa käyttäjälle tarjottavia mahdollisuuksia, niin ettei se rasita liiaksi käyttäjän kognitiivista kapasiteettia. Koska ihmisen kognitiivinen kapasiteetti, eli kyky käsitellä runsasta määrää samanaikaista tietoa, on rajallinen, ei hänelle pidä tarjota hyödyttömiä vaihtoehtoja tai informaatiota. Toinen ryhmähaastattelussa esille noussut esimerkki samasta aiheesta, oli käyttöliittymä, jonka pitää olla sellaisenaan niin hyvä, ettei käyttäjän tarvitse muokata sitä mitenkään. Haastattelun perusteella käyttöliittymän ei siis tarvitse olla muokattavissa, silloin kun se on jo valmiiksi toimiva.

Kirjatut havainnot, sekä edellä käsitelty ohjelman ominaisuuksien pelkistäminen, hyväksyttiin pääosin sellaisinaan mukaan varsinaiseen vaatimusmäärittelyyn. Saatujen vastausten perusteella, ohjelman yksinkertaisuus ja selkeys vaikuttivat olevan hyvin tärkeitä ominaisuuksia ja johtavan käyttäjän kannalta parempaan ohjelmistoergonomiaan. Niinpä ohjelman toiminnallisuuksien pelkistäminen ja äärimmäinen selkiyttäminen, päätettiin ottaa koko ohjelmiston määrittelyn lähtökohdaksi.

3.3 Havainnointitutkimus

Tämän työn lähtökohtana olivat, nykyisin käytössä olevissa heiteohjelmageneraattoreissa havaitut ohjelmistoergonomiset ongelmat. Niinpä näiden ongelmien selvittäminen oli erityisen tärkeää, vaatimusmäärittelyn muodostamista varten. Jo aiemmin kohdassa 3.2.1 päätettiin tutkia tätä aihetta juuri havainnointitutkimuksen keinoin, koska tällä menetelmällä aiheesta uskottiin saatavan parhaiten tietoa. Siksi toteutettiin vielä kolmas, erillinen tutkimus yhdellä nykyisten heitinohjelmistogeneraattorien tottuneella käyttäjällä. Tämä toteutettiin ns. osallistuvana havainnointitutkimuksena, jossa seurattiin käyttäjän toimintaa, hänen käyttäessään nykyistä ohjelmistogeneraattoria.

”Havainnoinnin ideana on kerätä tietoa tutkittavasta ilmiöstä sen luonnollisessa ympäristössä.” (Ronkainen, Pehkonen, Lindblom-Ylänne & Paavilainen 2011, 115) Havainnointi jaetaan lähteestä riippuen useampiin erilaisiin tyyppeihin, joista tässä tapauksessa valittiin osallistuva havainnointi. Osallistuvassa havainnoinnissa tutkija osallistuu itse toimintaan, jossakin roolissa ja pääsee siten lähemmäksi tutkittavaa aihetta ja saa siitä puhtaasti ulkopuolista havainnoijaa omakohtaisemman käsityksen. Havainnoinnin systemaattisuuden kannalta, tehtyjen havaintojen kirjaaminen esimerkiksi esivalmisteltuun havaintolomakkeeseen tai havaintopäiväkirjaan on tärkeää. Etukäteen on myös päätettävä mitä pyritään havainnoimaan ja mikä on havainnoinnin erityinen kohde. (Ronkainen ym. 2011, 115.) ”Kohdistettu haastattelu yleensä yhdistetään osallistuvalla tai tarkkailevalla havainnoinnilla saatuun havainnointiaineistoon.” (Vilkka 2006, 44-45)

3.3.1 Havainnointitutkimuksen toteutus

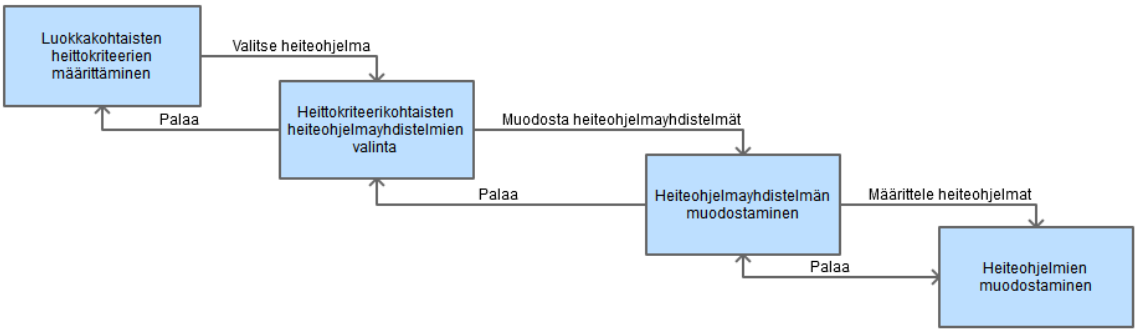
Havainnointitutkimus päästiin toteuttamaan pian ryhmähaastattelun jälkeen, eikä siihen muodostettu erillistä tiedonkeräyssuunnitelmaa. Tutkimuksessa keskityttiin havainnoimaan ja kirjaamaan ylös erityisesti asioita, jotka hankaloittavat tai hidastavat työskentelyä tai vaikuttavat kielteisellä tavalla työn lopputulokseen tai luotettavuuteen, tutkimuksessa käytetyllä generaattorilla. Käyttäjälle kerrottiin, että tutkimuksessa havainnoidaan erityisesti kussakin vaiheessa eteen tulevia ongelmia, sekä hänen kognitiivista kapasiteettiaan kuormittavia tekijöitä. Kyseisessä tutkimuksessa ohjelmoitava laite oli BAE-Systemsin ALE-47 – omasuojaheitinjärjestelmä. Yksityiskohtaisia tietoja varsinaisesta generaattoriohjelmistosta ei ole luottamuksellisuussyistä mahdollista liittää tähän tutkimukseen. Niinpä tehdyistä havainnoista kerrotaan ainoastaan yleisellä tasolla.

Osallistuvan havainnoinnin mukaisesti tutkija osallistui itse työskentelyyn avustajan roolissa, mikä helpotti työskentelyn seuraamista ja toiminnan ymmärtämistä. Luottamuksellisuussyistä työskentelyä ei tallennettu mitenkään, eikä työskentelyn jälkianalyysi tallenteen perusteella ollut siis mahdollista. Niinpä havainnointi suoritettiin reaaliajassa työskentelyn ohessa ja tehdyt havainnot kirjattiin ylös erillisiin muistiinpanoihin. Työskentely varsinaisen käyttäjän avustajana vaikeutti havaintojen tekemistä ja kirjaamista, koska ne olivat samanaikaisia mutta toisistaan

erillisiä toimintoja. Lisäksi tehtävän työn ja käytetyn ohjelmiston tuttuuden takia, jotkin käytettävyysongelmat saattoivat jäädä huomaamatta. Havaintoja kuitenkin tehtiin ja ne tukivat aiemmin haastattelu- ja kyselytutkimuksilla saatuja tuloksia. Käyttämällä automaattista tallennusjärjestelmää ja tekemällä tallenteesta kattava jälkianalyysi, olisi saavutettu varmastikin enemmän ja laadukkaampia havaintoja.

3.3.2 Havainnointitutkimuksen tulokset

Erityisen selkeästi tässä havainnointitutkimuksessa tulivat esiin seuraavissa kappaleissa kerrotut asiat. Ensimmäinen, ja oikeastaan koko generaattoriohjelmiston käytettävyyttä eniten haittaava, tekijä olivat runsaat erilliset ikkunat. Lähes kaikkia toimenpiteitä varten avattiin kokonaan uusi ikkuna, jossa tiedot kyseiseen kokonaisuuteen syötettiin. Esimerkiksi pelkästään yhden heitesekvenssin määrittelemiseksi, piti ensin kulkea neljän peräkkäin avattavan ikkunan kautta, joista jokaisesta piti myös poistua käänteisessä järjestyksessä. Tämän jälkeenkään edellisessä kohdassa määritelty, yksittäinen heitesekvenssi ei ollut vielä käytössä, vaan se piti vielä erikseen käydä liittämässä halutun uhkan yksittäisiin heittotilanteisiin, yksi kerrallaan. Missään ikkunassa käyttäjälle ei myöskään kerrottu, mitä hänen tulisi tehdä saadakseen heitesekvenssin määriteltyä. Tätä prosessia on havainnollistettu kuvassa 7. Kuvaa on tekijänoikeudellisista ja luottamuksellisista syistä, yksinkertaistettu ja muokattu, joten se ei sellaisenaan kuvaa suoraan mitään todellista ohjelmistoa tai laitteistoa.



Kuva 7. Luokkakohtaisen heiteohjelman määrittäminen.

Edellä kuvatussa prosessissa, käyttäjälle ei myöskään tarjottu mitään keinoa varmistua tekemänsä ohjelmoinnin kattavuudesta tai sen puutteista. Käytännössä siinä vaiheessa, kun käyttäjä uskoi heitesekvenssin olevan valmis, hänen täytyi vielä

käydä kertaalleen tarkistamassa kaikki tekemänsä työ, varmistuakseen että oli muistanut täyttää kaikki vaaditut tiedot.

Edellä generaattoriohjelman runsaita erillisiä ikkunoita ja automaattisten varmistusten ja ohjeiden puuttumista, pidettiin käytettävyyden ongelmina. Käytettävyys kuvaa minkä tahansa tuotteen ominaisuuksia, jotka yhdessä vaikuttavat siihen, kuinka sujuvasti tuotteen käyttäjä pystyy saavuttamaan haluamansa päämäärän tuotetta käyttäessään. Kyse on siis ihmisen ja jonkin laitteen välisestä vuorovaikutuksesta ja sen sujuvuudesta. Hyvä käytettävyys helpottaa käyttäjän työtä, kun taas huono käytettävyys hidastaa tai jopa estää työn tekemisen. Käytettävyys koostuu useista eri osatekijöistä, joita ovat opittavuus, muistettavuus, tehokkuus, pieni virhealttius ja miellyttävyys (Kuutti 2003, 13). Edellisessä esimerkissä esitellyt, käytettävyyden ongelmat, koskevat siis kaikkia käytettävyyden osatekijöitä. Sinänsä tutkittu generaattoriohjelmisto ei itsessään osoittautunut virhealttiiksi, mutta käyttäjän oli helppo tehdä virheitä, joita hänen oli vaikea havaita. Niinpä ohjelmaa ei voida kokonaisuutena pitää käytettävyydeltään, myöskään virhealttiuden osalta, erityisen onnistuneena.

Toinen erityisen ongelmallinen käyttäjätilanne muodostui, kun yksittäinen heiteohjelma piti liittää haluttuun uhkaan ja käyttötilanteeseen. Tässä uhkan, heittotilanteen ja heiteohjelman liittämisvaiheessa, käyttäjälle ei kerrottu miten liittäminen tapahtuu, eikä se ollut käyttäjälle ilmeistä. Testitilanteessa edes erittäin kokenut käyttäjä, ei onnistunut liittämään haluttua vastetta haluamaansa uhkaan, koska käyttöliittymä ei tukenut riittävästi hänen havainnointiaan. Lisäksi käyttöliittymä oli muuhun ohjelmistoon nähden erilainen. Tässä tilanteessa ohjelmiston kokenut käyttäjäkään ei osannut suorittaa toimenpidettä, ilman käyttöohjeen lukemista.

Edellisissä kappaleissa kuvatut havainnot ovat selkeitä ohjelmistoergonomisia ongelmia. Testattu ohjelmisto on tarpeettoman monimutkainen ja tieto sen sisällä hajaantuu eri ikkunoihin niin, että kokonaisuuden hahmottaminen vaikeutuu. Testatun ohjelmiston käyttöliittymä ei tue työskentelyä erityisesti loogisuuden tai helpon hahmotettavuuden kautta. Lisäksi käyttöliittymä on erilainen ohjelmiston eri osissa. Työn seuraavassa vaiheessa määritettävässä ohjelmistossa, tällaisiin ongelmiin voidaan vastata, tekemällä määritettävän ohjelman käyttöliittymä

mahdollisimman loogiseksi ja helpoksi hahmottaa. Hyvässä käyttöliittymässä eri toiminnot on ryhmitetty selkeästi omiksi kokonaisuuksiksi, jotka tarjotaan käyttäjälle kokonaisuutena, eikä pilkota niitä eri ikkunoihin tai sijoiteta niitä käyttöliittymässä erilleen. Lisäksi käyttäjän hahmottamista tuetaan värein, avustavin tekstein sekä muin keinoin. Myös työn kulku pyritään järjestämään etenemään loogisesti, länsimaisen lukemistavan mukaisesti, yläoikealta alas, vasemmalle edeten. Hyvä otsikointi auttaa käyttäjää hahmottamaan erillisiä kokonaisuuksia.

Tavoitteena on, että tässä työssä määritettävän ohjelmiston käyttöliittymästä tehdään niin ergonominen ja helposti sisäistettävä, että sitä kykenee käyttämään ilman varsinaista koulutusta. Kuitenkin ohjelmiston käyttöliittymään lisätään vielä hyvä ja helppokäyttöinen ohjeistus, jotta käyttäjä selviää varmasti myös mahdollisista ongelmatilanteista. Mikäli jokin toiminto tai kohta jää näistä huolimatta vaikeaselkoiseksi, voidaan siihen lisätä ohjeteksti, joka avautuu käyttäjän nähtäväksi automaattisesti. Tämä tapahtuu, kun käyttäjä on valinnut kyseiseen kohtaan liittyvän kentän, mutta ei ole antanut mitään syötettä muutaman sekunnin kuluessa, tai pitää hiiren osoitinta pitkään samalla paikalla. Tarvittaessa kaikki kyseiseen ikkunaan liittyvät ohjeet saa esiin klikkaamalla kysymysmerkkiä ikkunan oikeassa yläkulmassa, tai avaamalla ohjelmiston erillisen käyttöohjeen valikosta tai painamalla F1 -näppäintä. Näiden avulla voidaan varmistaa ohjelmiston sujuva käyttö, ilman vaaraa kokonaisuuden hahmottamisen ongelmista tai riittävän ohjeistuksen puuttumisesta aiheutuvasta työskentelyn estymisestä.

3.4 Omasuojaheittimien ominaisuuskartoitus

Jotta tuotettava ohjelmistomäärittely olisi tavoitteiden mukaisesti yleinen, siinä piti huomioida mahdollisimman kattavasti erilaisten omasuojaheitinjärjestelmien ominaisuuksia. Lisäksi työssä tuli yrittää nähdä myös mahdollisia tulevaisuuden kehityskulkuja, jotka voisivat muuttaa joitakin omasuojaheittimiltä vaadittavia ominaisuuksia. Jotta kaikki nämä erilaiset ominaisuudet voitiin huomioida ohjelmistomäärittelyä muodostettaessa, täytyi selvittää mitä erilaisia ominaisuuksia eri omasuojaheittimillä on. Koska erilaisia omasuojaheitinlaitteistoja ja niiden valmistajia on kohtuullisen suuri määrä, on niillä myös yhteneviä ja toisistaan poikkeavia ominaisuuksia melko suuri määrä.

Useimmiten omasuojalaitteiden yksityiskohtaiset ominaisuudet eivät ole julkista ja yleisesti saatavilla olevaa tietoa, eivätkä valmistajat tyypillisesti halua kertoa niistä, muille kuin potentiaalisille asiakkaille. Niinpä tässä työssä keskityttiin hankkimaan näitä ominaisuustietoja ainoastaan julkisista lähteistä, kuten esittelymateriaaleista ja vastaavista. Tällaista materiaalia oli saatavilla kohtuullisen hyvin suoraan internetistä. Aiheen tuttuuden vuoksi myös iso osa ominaisuuksista oli jo valmiiksi tiedossa, vaikkakin lähdemateriaaleihin tutustuttaessa löydettiin myös uusia ja yllättäviä ominaisuuksia. Esimerkkinä tällaisesta muihin omasuojaheittimiin verrattuna hyvin erilaisesta heittimestä on alla kuvassa 8. näkyvä SAAB Ab:n valmistama BOL –heitin. Siinä heitteitä ei heitetä omista heitepaikoistaan perinteisesti heittopanoksella ilmaan, vaan vapautetaan peräkkäin mekaanisesti ilmavirtaan suoraan taaksepäin.

Kuva 8. BOL -omasuojaheitin Australian ilmavoimien F-18 –hävittäjässä. (BOL FOR F/A-18 ADVANCED COUNTERMEASURE DISPENSER)

Hankitun tiedon pohjalta muodostettiin lista erilaisista ominaisuuksista, joita ohjelmiston tulee kyetä tukemaan, jotta se olisi aidosti yleiskäyttöinen. Lista ominaisuuksista on tämän työn liitteenä 4. Koska eri lentokoneilla on toisistaan poikkeavia ominaisuuksia, jotka vaikuttavat omasuojaheittimien toimintaan, myös ne täytyi huomioida. Niinpä myös lentokoneiden omasuojalaitteisiin vaikuttavat ominaisuudet selvitettiin ja lisättiin mukaan ominaisuuslistaukseen. Myös koneeseen asennetut muut omasuojalaitteet antavat tietojaan omasuojaheittimien käyttöön tai

ohjaavat niiden toimintaa, joten nämäkin otettiin mukaan liitteen 4. listaukseen. Ominaisuuslistauksesta muodostettiin siis mahdollisimman kattava, mutta työn myöhemmissä vaiheissa havaittiin, että osa listan ominaisuuksista voitiin yksinkertaisesti jättää huomiotta, koska niillä ei ollut vaikuttavuutta tässä ohjelmistossa. Toisaalta joitakin ominaisuuksia rajattiin ohjelmistomäärittelyn ulkopuolelle myös muista syistä.

Muodostettu listaus erilaisista ominaisuuksista on melko pitkä ja vain osa listatuista ominaisuuksista on yhteisiä eri heitinlaitteistoille. Toisaalta ei voitu olettaa muodostetun ominaisuuslistauksen olevan täydellisen kattava kuvaus eri omasuojaheittimien ominaisuuksista. Ei voitu myöskään olettaa ominaisuuslistan sellaisenaan tukevan mahdollisia tulevaisuuden kehityspolkuja. Niinpä vaatimusmäärittelyyn päätettiin lisätä vaatimus, mahdollisten omasuojaheittimen uusien ominaisuuksien lisäämisestä ohjelmistoon, loppukäyttäjän toimesta. Eli loppukäyttäjän pitää pystyä lisäämään kulloisellekin omasuojaheittimelle ominaisuuksia, joita ei ole aiemmin huomioitu. Toisaalta monet listan ominaisuuksista koskevat vain yksittäisiä heitinjärjestelmiä, joten kaikkien listan ominaisuuksien tarjoaminen ohjelmiston käyttäjälle jatkuvasti, olisi todennäköisesti harhaanjohtavaa. Lisäksi se kuormittaisi käyttäjän kognitiivista kapasiteettia tarpeettomasti. Tämän välttämiseksi vaatimusmäärittelyyn päätettiin asettaa vaatimus, ettei ohjelmisto missään tilanteessa tarjoa tarpeettomia vaihtoehtoja käyttäjälle.

3.5 Tutkimustuloksista

Ronkainen ym. (2011, 117) toteavat kirjassaan seuraavaa; Tutkimusaineisto on riittävää kun uusi tutkimusaineisto ei tuo enää uutta tietoa tutkittavasta aiheesta, vaan teemat toistuvat tai silloin kun tutkija pystyy vastaamaan tutkimuskysymykseensä. Koska samat teemat ja tulokset toistuivat jo osittain sekä kyselyssä että haastattelussa, voitaisiin karkeasti olettaa tutkimusaineiston olevan riittävää. Toisaalta tutkimukset toteutettiin pienelle ryhmälle ja kyselytutkimukseen saatiin ainoastaan kaksi vastausta, joten otos on suppea. Tutkimusaiheesta johtuen vastaajaryhmän koko on kuitenkin jo luonnostaan hyvin rajallinen, koska käytettävissä olevia, alaa tuntevia, henkilöitä on vähän.

Tällaisessa tilanteessa kolmen erillisen tutkimuksen perusteella voidaan olettaa, että ainakin kaikkein olennaisimmat tutkimusvastaukset on saatu kerättyä. Myös todennäköisyys uuden tiedon saamiseen, tutkimalla edelleen samaa käyttäjäryhmää, lienee vähäinen. Edellisen lisäksi yleisiä ohjelmistoteknisiä ratkaisuja ja konventioita voidaan hyödyntää myös ilman uutta käyttäjätutkimusta, koska niistä on saatavilla aineistoa muutenkin. Vaikka eri tutkimuksissa käytetty ryhmä oli pieni, se edusti tässä työssä määritettävän ohjelman tulevia loppukäyttäjiä erityisen hyvin. Niinpä saatuja tuloksia pidetään kohtuullisen luotettavina ja riittävinä. Lisäksi kuten kappaleen 3.1 alussa kerrottiin, tutkimuksen tavoitteena ei ollut uusi tieteellinen tietämys, vaan ainoastaan kerätä ohjelmistoon liittyviä käyttäjien toiveita ja vaatimuksia. Toteutetuilla tutkimuksilla tämä tavoite toteutui, joten vaatimusmäärittely päätettiin toteuttaa tähän mennessä saatujen tulosten pohjalta.

3.6 Vaatimusmäärittelyn toteutus

Varsinainen vaatimusmäärittely tuotettiin kappaleiden 3.1 – 3.3 mukaisten tutkimusten ja työn tekijän oman harkinnan perusteella. Jotta vaatimusmäärittely saatiin mahdollisimman kattavaksi ja eheäksi kokonaisuudeksi, käytettiin siihen valmista IEEE:n standardin 830 vuoden 1998 päivitysversion (IEEE 830 1998 10-20) mukaista ohjelmiston vaatimusmäärittelyn runkoa. Standardin mukainen vaatimusmäärittelyn runko on vapaasti opinnäytetyön tekijän suomentamana kuvattu taulukkoon 6. Taulukossa olevia otsikoita on osittain lavennettu, varsinaisen standardin mukaisia otsikoita laajemmiksi asian selkiyttämiseksi. Laventamisessa on käytetty apuna standardin otsikkokohtaisia lisäohjeita.

Sisällysluettelo

1. Johdanto
    1.1. Määrittelyn tarkoitus
    1.2. Ohjelman nimi, tarkoitus ja toiminnot
    1.3. Määritelmät ja lyhenteet
    1.4. Viittaukset muihin asiakirjoihin
    1.5. Määrittelyn yleiskuvaus
2. Ohjelman yleiskuvaus
    2.1. Toimintaympäristö ja sen rajoitteet

2.2. Ohjelman toiminnot
2.3. Käyttäjien ominaisuudet
2.4. Ohjelman rajoitteet
2.5. Oletukset ja riippuvuudet
2.6. Mahdolliset myöhemmät vaatimukset

3. Yksityiskohtaiset vaatimukset
3.1. Ulkoiset liitynnät
3.2. Ohjelman suorittamat toiminnot
3.3. Suorituskykyvaatimukset
3.4. Tietokannan ominaisuusvaatimukset
3.5. Suunnitteluraajoitteet
3.6. Noudatettavat standardit
3.7. Erilliset koko ohjelmaa koskevat vaatimukset

4. Mahdolliset lisävaatimukset

Hakemisto

Liitteet

Taulukko 6. Ohjelmiston vaatimusmäärittelyn runko IEEE:n standardin 830 (IEEE 830 1998, 11-20) mukaan opinnäytetyön tekijän vapaasti suomentamana.

Taulukon 6. mukaisen vaatimusmäärittelyn osa kolme, on koko määrittelyn laajin ja tärkein kokonaisuus. Siinä tulee määritellä kaikki ohjelman toiminnot sillä tarkkuudella, että ohjelmistosuunnittelija pystyy niiden perusteella suunnittelemaan ohjelman, joka täyttää asetetut vaatimukset ja ohjelmistotestaaja pystyy testaamaan että ohjelma täyttää samat vaatimukset. Tämän osion pitää vähintäänkin kuvata kaikki ne toiminnot, jotka vaativat syötteen, tuottavat tuloksen tai tapahtuvat syötteen seurauksena tai tuloksen saavuttamiseksi. (IEEE 830 1998, 15.)

Standardin käytöstä huolimatta vaatimusmäärittelyn varsinainen ydin, eli kappaleen kolme yksityiskohtaiset vaatimukset, supistuivat vaatimusmäärittelyn ensimmäisessä valmiissa versiossa varsin vähäisiksi. Yksityiskohtaiset vaatimukset koostuivat lähinnä listasta käyttäjiltä kerättyjä vaatimuksia. Joitakin lisäyksiä kuitenkin tehtiin jo ennen toiminnallisen määrittelyn aloittamista, koska vaatimusmäärittelyssä oli sellaisenaan selkeitä puutteita. Koska työn varsinaisena tarkoituksena oli tuottaa toiminnallinen

määrittely, ei vaatimusmäärittelyä haluttu kuitenkaan jumiutua viilaamaan liikaa. Lisäksi kuten Haikalan kirjassa (2004, 78) todetaan, vaatimusmäärittelyn tärkeimpänä lopputuloksena syntyy toiminnallinen määrittely. Tässä vaiheessa havaittiin, että vaatimukset joita käyttäjiltä oli kerätty, olivat kaikki käytännössä myös ohjelmiston toiminnallisia vaatimuksia. Tämä helpotti vaatimusmäärittelyn tekemistä, koska varsinainen erillinen vaatimusmäärittely voitiin rajata hyvin korkean tason asiakirjaksi, jossa kuvataan ainoastaan ohjelmiston yleisiä vaatimuksia, sekä niitä tuloksia ja tavoitteita joita sillä halutaan saavuttaa. Niinpä tätä vaatimusmäärittelyä päätettiin käyttää ohjelmiston minimivaatimuksina, jotka sen ainakin tulee täyttää ja joihin tehdään myöhemmin lisäyksiä, sen perusteella mihin ongelmiin tai valintoihin tulevissa vaiheissa ajaudutaan.

Muodostettua vaatimusmäärittelyä täydennettiinkin vielä ohjelmiston toiminnallisen määrittelyn yhteydessä havaittujen tarpeiden pohjalta. Periaatteessa tämä on kappaleen kolme alussa kerrotun vastaista, aiheuttaen ongelmia ohjelmiston myöhempien vaiheiden toteuttamisessa. Toisaalta se on juurikin samassa yhteydessä mainittua vaatimustenhallintaa, joka on olennainen osa jokaista ohjelmistoprosessia, sen alusta loppuun saakka. Joka tapauksessa muutokset varsinaiseen vaatimusmäärittelyyn olivat välttämättömiä hyvän ja toimivan lopputuloksen saavuttamiseksi. Muodostetun vaatimusmäärittelyn laajuudesta johtuen, sitä ei liitetty kokonaisuudessaan osaksi tätä raporttia, vaan ainoastaan sen sisällysluettelo on nähtävissä liitteenä 5.

4 Ohjelmiston toiminnallinen määrittely

4.1 Tavoitteet toiminnalliselle määrittelylle

Asetettaessa tavoitteita tälle työlle, määriteltiin että toisen vaiheen tarpeellisuutta, arvioidaan ensimmäisessä vaiheessa saatujen tulosten perusteella. Sivulla 23, taulukossa kolme, läpikäydyistä tuloksista käy ilmi, että tällainen työkalu olisi käyttäjille hyödyllinen, mikäli se täyttää sille asetetut toiveet. Niinpä kun vaatimusmäärittelyn ensimmäinen, riittävän täydellinen, versio oli saatu tuotettua, työtä jatkettiin tavoitteiden mukaisesti, muodostamalla ohjelmistolle toiminnallinen määrittely. Todellisuudessa tämä työ oli aloitettu jo vaatimusmäärittelyn

kirjoittamisen ohessa, tutkimuksissa kerättyjen vastausten perusteella. Näin oli tehty, koska näillä kahdella asiakirjalla on varsin vahva yhteys toisiinsa. Lisäksi pääosa vaatimusmäärittelyyn kirjatuista vaatimuksista, voitiin siirtää melko suoraan osaksi toiminnallista määrittelyä. Niinpä työtä tehtiin jossain määrin rinnakkaisina prosesseina.

Toiminnallisessa määrittelyssä haluttiin painottaa, alkuperäisen suunnitelman mukaisesti, hyvää käyttäjäkokemusta ja käsitellä teknisiä yksityiskohtia mahdollisimman vähän. Tämä tarkoitti samalla monien asioiden jättämistä määrittelemättä, jättäen tilaa myöhemmälle suunnittelulle. Käytännössä tämä tarkoitti, ettei toiminnallisesta määrittelystä pyrittykään saamaan täydellistä vielä tässä vaiheessa. Syinä valintaan olivat tavoite, lähinnä käyttäjäkokemuksen määrittelemisestä ja toisaalta opinnäytetyöntekijän vähäinen kokemus ohjelmistoprojekteista ja ohjelmoinnista. Syvällisempi tekninen suunnittelu olisi vaatinut laajaa lisäopiskelua ohjelmistotekniikasta ja tietokannoista.

Edellä kerrottu toiminnallisen määrittelyn jättäminen väljäksi, on kuitenkin ohjelmistotuotannon perusperiaatteiden vastaista. Jos suunnittelu on vajaata tai kaikkia osia ei ole määritelty, se saattaa johtaa ongelmiin ohjelmistoprojektin myöhemmässä vaiheessa. Esimerkiksi tietokannan jättäminen suunnittelematta toiminnallisen määrittelyn tässä vaiheessa, saattaa johtaa myöhemmin tarpeeseen, muuttaa tässä vaiheessa määritettyä käyttäjäkokemusta. Niinpä toiminnallisen määrittelyn puutteelliset osat on vielä määriteltävä tarkasti, ennen varsinaisen ohjelmointityön aloittamista.

Koska opinnäytetyössä määritettävän ohjelmiston tuottaminen perustuu hyvään käytettävyyteen ja ohjelmistoergonomiaan, pyrittiin suunnittelussa luonnollisesti noudattamaan hyviä ohjelmistoergonomisia käytäntöjä. Varsinaisesti ohjelmistoergonomiasta on toistaiseksi saatavilla varsin niukalti materiaalia. Lisäksi lähes kaikki standardit ovat maksullisia, joten mahdollisuudet niiden hyödyntämiseen olivat rajallisia. Niinpä tässä työssä hyödynnettiin ISO 14915-1 –standardia (ISO 14915-1, 2002), joka määrittelee multimediaohjelmistojen käyttöliittymäergonomiaa. Tämä oli saatavilla olevista standardeista tähän työhön soveltuvin. Toiminalliseen määrittelyyn kirjattiin kuitenkin myös vaatimus ISO 9241-210 –standardin (ISO 9241-210) huomioimisesta, vaikka sitä ei voitukaan hyödyntää

tässä määrittelyssä. ISO 9241-210 -standardi (ISO 9241-210) käsittelee tietokoneperustaisten, interaktiivisten systeemien ja ihmisen välisen vuorovaikutuksen ergonomiaa, ja olisi siten ollut luonnollinen valinta käytettäväksi tässä opinnäytetyössä.

Varsinainen ohjelmistoergonomia käsitteenä, kokoaa yhteen useita erilaisia tekijöitä, samalla tavalla kuin muissakin ergonomian käsitteissä. Ohjelmistoergonomia on siis yleiskäsite, kaikille niille erillisille asioille, jotka yhdessä tekevät ohjelmistosta helpon ja miellyttävän käyttää, sekä sellaisen ettei sen käyttö rasita käyttäjää tarpeettomasti tai altista häntä vaaroille. Aiemmin ohjelmistoergonomia on käsitetty ainoastaan käyttöliittymiin liittyväksi asiaksi. Ohjelmistoergonomiassa käyttöliittymä on edelleenkin merkittävässä roolissa, mutta muut, erityisesti hyvään käytettävyyteen liittyvät, tekijät ovat yhä merkittävämpiä tekijöitä pelkän käyttöliittymän rinnalla. Pelkästään hyvä käyttöliittymä ei tee esimerkiksi epäluotettavasta ohjelmistosta hyvää. Ohjelmistoergonomiaan liittyy siis erillisiä osa –alueita, joita ovat ainakin käyttöliittymä, käytettävyys ja ihmisen kognitiiviset ja fyysiset ominaisuudet. Useimmiten ohjelmistoja suunniteltaessa unohdetaan juuri iIhmisten kognitiiviset ominaisuudet, jolloin niiden asettamia rajoja ei noudateta. Ihmisten muistia koetellaan tarpeettomasti, jättämällä asioita kertomatta, tai antamalla käyttäjälle samanaikaisesti niin paljon vaihtoehtoja, ettei hän kykene tekemään valintaa. Ohjelmistoergonomia on siis ainoastaan tapa niputtaa yhteen ja käsitellä kokonaisuutena monia erillisä asioita, joita ohjelmistotuotannossa on tai olisi pitänyt huomioida jo pitkään.

4.2 Toiminnallisen määrittelyn toteutus

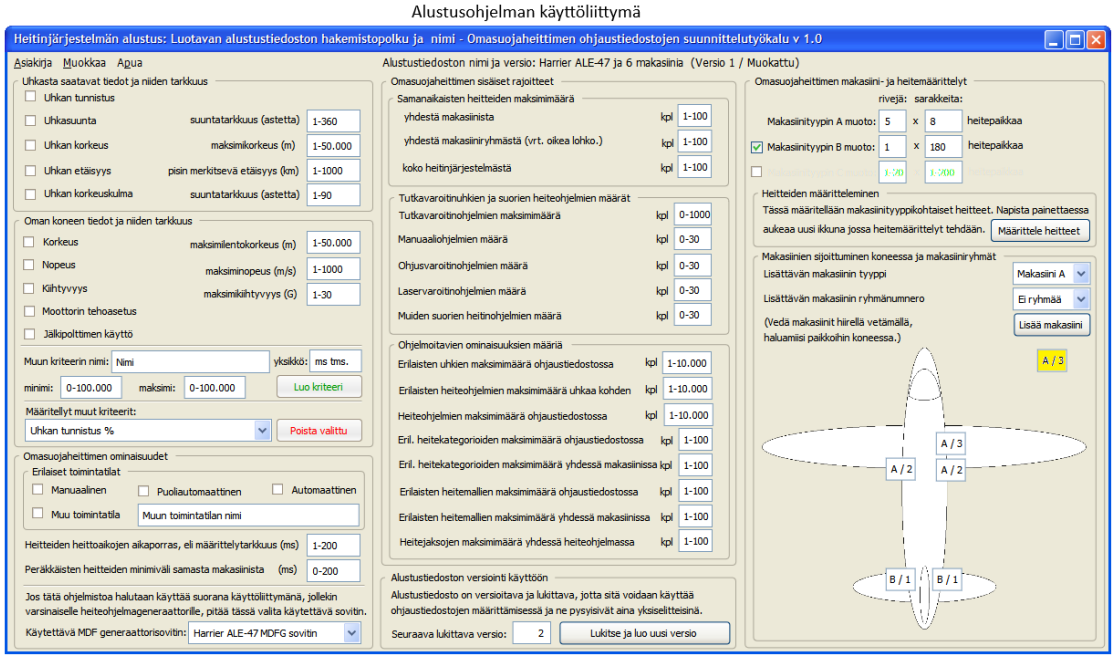
Ohjelmiston toiminnallista määrittelyä lähdettiin toteuttamaan hieman soveltaen. Koska IEEE –standardin 830 (IEEE 830 1998) mukainen rakenne, jota käytettiin aiemmin myös vaatimusmäärittelyn tekemisessä, vaikutti olevan hyvä pohja myös toiminnallisen määrittelyn tekemiseen, käytettiin aluksi sitä. Jo kappaleen 1 jälkeen huomattiin, että ohjelmiston hahmottaminen suoraan standardin mukaisessa järjestyksessä, on oikeastaan mahdotonta, jolloin siirryttiin kesken toisen kappaleen tuottamisen suunnittelemaan pääohjelman tietorakennetta. Tässä vaiheessa

havaittiin että ohjelmasta täytyy erottaa erillinen alustustila, sillä ohjelmassa ei haluttu joutua määrittelemään muita kuin ohjaustiedostokohtaisia asioita.

Käytettävä lentokone ja siihen liitetty omasuojajärjestelmä, muodostavat laitteistokokonaisuuden, joka pysyy suhteellisen vakiona pitkiäkin aikoja ja jopa vuosikymmeniä. Uusia ohjaustiedostoja kyseiselle laitteistokokonaisuudelle taas voidaan tuottaa hyvinkin usein. Niinpä on luontevaa, että laitteistokokoonpanosta riippuvat asiat ja ohjaustiedostokohtaisesti valittavat asiat, erotetaan toisistaan. Niinpä opinnäytetyössä määriteltävästä ohjelmasta muodostuikin ohjelmisto, jossa on erillinen alustustila ennen pääohjelmaan siirtymistä. Alustustilassa tavallaan luodaan käytettävä heitinlaitteisto, sekä sen integraatio lentolaitteeseen. Siinä määritellään käytettävän lentokoneen, ja siihen liitetyn omasuojaheitinlaitteiston ominaisuudet, sekä käytettävissä olevat heitemallit.

Alustusohjelman käyttöliittymä suunniteltiin suoraan Liitteen 4 perusteella. Tämä oli mahdollista, koska kappaleessa 3.4 kerätyt omasuojaheittimien toiminnallisuuteen vaikuttavat asiat olivat juuri samat, joita alustusohjelmaan pitää syöttää. Näin pääohjelman käyttöliittymä, voidaan muodostaa vastaamaan kulloistakin lentokoneen ja siihen liitettyjen omasuojalaitteiden kokonaisuutta. Tällöin vähennetään käyttäjän kuormitusta, tarjoamalla hänelle ainoastaan relevantteja vaihtoehtoja. Alustustilassa tehdyt määrittelyt, tallennetaan erilliseen alustustiedostoon, jota pääohjelma käyttää, tarjoamansa käyttöliittymän muokkaamiseen ja syötteiden raja-arvojen asettamiseen. Näin loppukäyttäjän ei tarvitse koskaan itse muistaa, mitä vaihtoehtoja hänellä on käytettävissään, koska ohjelmisto antaa hänelle käyttöön vain oikeat vaihtoehdot. Tätä alustustiedostoa siis käytetään, määriteltäessä kaikkia niitä ohjaustiedostoja, joita kyseiselle laitteistokokoonpanolle tuotetaan. Alustusohjelman pääkäyttöliittymä on nähtävissä seuraavassa kuvassa 9.

Alustusohjelman käyttöliittymä



Kuva 9. Alustusohjelman käyttöliittymä.

Tyypillisesti omasuojaheittimien ohjaustiedostojen määrittelyyn liittyy asioita, jotka määritellään vai yhden kerran koko ohjaustiedolle. Toisaalta se sisältää myös asioita joita määritellään, uhka tai uhkaluokka ja heitesekvenssikohtaisesti, monia kertoja uudelleen ohjaustiedoston sisällä. Tämän määriteltävän ohjelmiston tarkoituksena on helpottaa juuri tuota jatkuvasti toistuvaa määrittelytyötä. Toisaalta erilaisia heitintyyppejä ja niihin sisältyviä, toisista heitintyypeistä poikkeavia ominaisuuksia, on valtavasti. Myöskään laitteista julkisesti saatavilla oleva tieto, ei sisällä kaikkia laitekohtaisesti ohjelmoitavia ominaisuuksia. Näistä syistä, tämän ohjelmiston avulla ei pääasiallisesti määritellä mitään sellaisia asioita, jotka määritellään ainoastaan kerran kullekin ohjaustiedostolle, kuten tyypillisesti erilaisten laitteistomääritysten tekeminen.

Edellä kuvatut erikoiset laitteistomäärittelyt käyttäjä voi halutessaan kirjoittaa omasanaisesti koko ohjaustiedostoa koskevaan lisämääritykset -kohtaan, jonka käyttäjä voi avata pääkäyttöliittymästä. Tämä lisämääritykset –kohta on erillinen tekstitiedosto, jonka koko mukautuu automaattisesti sisällön mukaan ja se liitetään tarvittaessa, generoitavien raporttien alkuun tai loppuun, käyttäjän valintojen mukaisesti. Näiden lisämääritysten tekeminen ei ole tämän ohjelmiston lopputuloksen kannalta pakollista, mutta ne lisäävät ohjaustiedoston suunnittelun

täydellisyyttä. Lisämääritysten tekeminen saattaa olla myös pakollista, mikäli määrittelyn ja varsinaisen ohjelmointityön tekevät eri henkilöt tai määrittely hyväksytään formaalisti ennen varsinaisen ohjelmoinnin aloittamista. Nämä määrittelyt eivät siis ole osa edellisissä kappaleissa kerrottua alustustiedostoa, koska ne ovat ohjaustiedostokohtaisia lisämäärityksiä.

Kun varsinaisen pääohjelman rakennetta alettiin määritellä, huomattiin ettei IEEE – standardin 830 (IEEE 830 1998) mukainen rakenne ollutkaan paras valinta toiminnallisen määrittelyn tekemiseen ja sille päätettiin etsiä sopivampi vaihtoehto. Internetistä löytyikin helposti runsas määrä erilaisia, valmiita toiminnallisia määrittelyitä, sekä erilaisia pohjia niiden tekemiseen. Näistä valikoitiin tässä työssä käytettäväksi Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksen ylläpitämä pohjatiedosto (Ahtee 2012). Tämä pohjatiedosto oli helppokäyttöinen, tehty suomeksi ja sisälsi hyvän sisäisen ohjeistuksen. Pohjaa kuitenkin muutettiin, pohjassa olleen ohjeistuksen mukaisesti, vastaamaan paremmin juuri tämän ohjelmiston erityisvaatimuksia.

Osissa yksi ja kolme asetettujen tavoitteiden mukaisesti toiminnallisen määrittelyn tuottamisessa keskityttiin hyvän käyttäjäkokemuksen määrittelemiseen. Tätä pyrittiin toteuttamaan erityisesti suunnittelemalla koko ohjelmiston käyttö etenemään mahdollisimman loogisesti ja käyttäjälle luontaisella tavalla. Käyttöliittymäikkunoista pyrittiin tekemään mahdollisimman yksiselitteisiä sekä visuaalisia. Käyttöliittymissä pyrittiin hyödyntämään ainakin länsimaalaisille ihmisille tuttuja värikonventioita sekä etenemistapaa. Lisäksi erillisten ikkunoiden kokonaismäärä pyrittiin minimoimaan ja niiden käyttölogiikka sekä sisäinen rakenne pyrittiin pitämään mahdollisimman samankaltaisena. Tämä osoittautui kuitenkin melko vaikeaksi, koska osaan ikkunoista jouduttiin sijoittamaan paljon eri asioita tai laajoja kuvia. Toisiin ikkunoihin taas sisältyi vähemmän määriteltäviä asioita ja ne jäivät selvästi väljemmiksi. Kuitenkin pyrkimyksenä oli mahdollisimman hyvän ohjelmistoergonomian toteuttaminen, niin ettei käyttäjälle koskaan anneta liikaa vaihtoehtoja ja toisaalta hänelle näytetään kaikki mahdollisuudet selkeästi.

Toiminnallisessa määrittelyssä otettiin kantaa myös käyttäjän auttamiseen virhetilanteissa, sekä silloin kun hän miettii, kuinka jatkaa omaa työskentelyään. Aina jos käyttäjä antaa väärän syötteen, hänelle neuvotaan syötteen oikea muoto ja

annetaan jopa tarkentavaa tietoa syöteen tarkoituksesta. Samoin jos käyttäjä pitää hiiren osoitinta pitkään paikoillaan, hänelle annetaan ohjeistus kyseisen syötteen merkityksestä ja oikeasta muodosta. Ohjelmistosta pyrittiin tekemään niin selkeä ja yksinkertainen, että se olisi erittäin helposti opittava, eikä varsinaista muistettavuutta edes tarvittaisi.

Virhemahdollisuuksien poistamiseen kiinnitettiin myös erityishuomiota, koska se on yksi perussyy koko ohjelmiston tarpeellisuuteen. Ohjelmisto tarkastaa jokaisen annetun syötteen oikeellisuuden, eikä salli virheellisten syötteiden tallentamista. Ohjelmisto myös kertoo selkeällä värikoodauksella käyttäjälle, niin koko ohjaustiedoston, kuin jokaisen sen osa-alueenkin osalta, kulloisenkin suunnittelutilanteen. Näin minimoidaan käyttäjävirheen mahdollisuuksia ainakin suunnittelun kattavuuden osalta.

4.3 Toiminnallisen määrittelyn arviointi

Kun toiminnallisesta määrittelystä oli muodostettu sitä kuvaavat käyttöliittymäkuvat ja pääosa sanallisista kuvauksista, arvioitiin sitä käyttäen ns. Nielsenin listaa (Nielsen 1995). Nielsenin listassa on kuvattu kymmenen tärkeintä heuristiikkaa eli nyrkkisääntöä, joita kaikkien ohjelmistojen tulee noudattaa täyttääkseen käytettävyyden minimivaatimukset. Nielsenin lista on kuvattu vapaasti suomennettuna seuraavassa taulukossa 7.

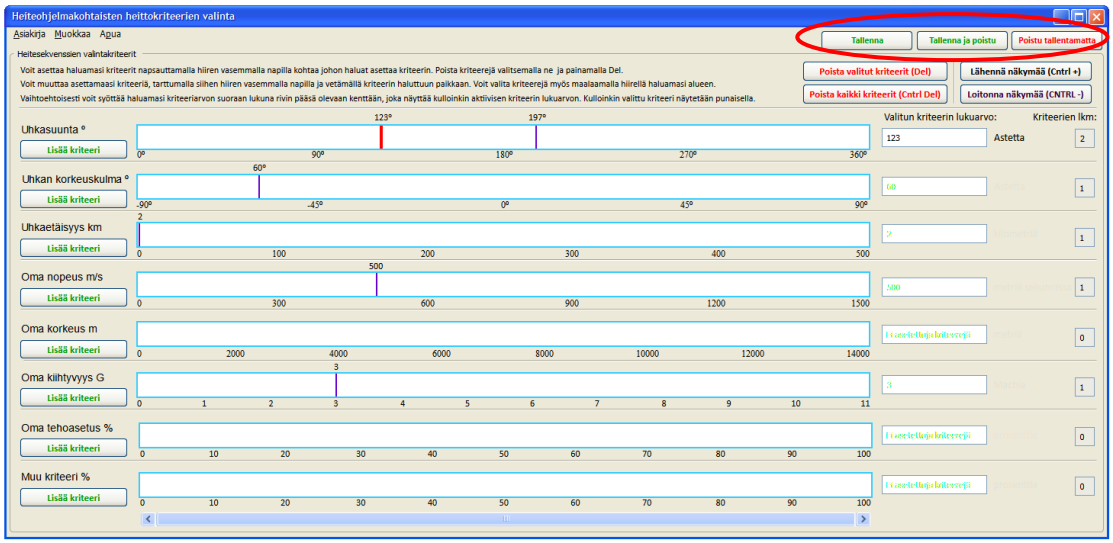
1. Käyttäjän pitää saada soveltuvaa palautetta ohjelmiston kulloisestakin toimintatilasta kohtuullisessa ajassa.
2. Järjestelmässä tulee käyttää käyttäjälle luontaista ja tuttua kieltä sekä noudattaa tavanomaisia käytäntöjä niin, että tieto näytetään käyttäjälle luonnollisesti ja loogisessa järjestyksessä.
3. Ohjelmiston kaikissa osissa tulee olla aina selkeä poistumistie ja edellinen toiminto pitää voida kumota ja toistaa.
4. Koko ohjelmiston pitää noudattaa johdonmukaisesti samoja käytäntöjä ja standardeja, eikä sanojen tai valintojen merkitys saa vaihtua eri osissa.
5. Ehkäise käyttäjävirheiden mahdollisuuksia ja kysy käyttäjältä varmistus virhemahdollisuuden sisältävissä tilanteissa.
6. Minimoi käyttäjän muistin rasitus näyttämälle tälle kaikki mahdollisuudet. Ohjeiden tulee olla näkyvissä tai helposti saatavilla.
7. Tue oikopolkuja, jotka nopeuttavat kehittyneempien käyttäjien toimintaa, mutta mahdollista myös peruskäyttäjän toiminta. Anna käyttäjän räätälöidä tyypillisimpiä toimintoja.
8. Dialogien pitää olla miellyttäviä ja ytimekkäitä, ilman tarpeetonta tietoa.
9. Auta käyttäjää selviytymään virhetilanteista, kertomalla selkeästi virheen syy ja keinot selvitä tilanteesta.

Taulukko 7. Nielsenin lista vapaasti opinnäytetyön tekijän suomentamana. (Nielsen 1995)

Ohjelmistosuunnitelman tarkistus Nielsenin listan (Nielsen 1995) avulla, toteutettiin melko pintapuolisesti. Arvioinnin perusteella suunnitelma oli kohtuullisen hyvällä tasolla ja pääasiallisesti täytti, subjektiivisesti arvioiden, Nielsenin listan mukaiset vaatimukset. Olennaisin havaittu puute, oli selkeän poistumistien puuttuminen useimmista ikkunoista. Sitä ei kuitenkaan korjattu vielä ennen käyttäjätestejä, vaikka joitakin muutoksia suunnitelmaan jouduttiinkin tekemään jo ennen testeihin siirtymistä. Nämä muutostarpeet johtuivat erityisesti suunnitelman epätäydellisyydestä ja tehdyistä valinnoista, jotka eivät kaikin osin olleet koko ohjelmistoa ajatellen täysin johdonmukaisia. Lisäksi, joidenkin asioiden vaikutusta ohjelmiston käytettävyyteen päätettiin havainnoida käyttäjätestissä, ennen mahdollisesti tarpeettomien muutosten tekemistä.

Esimerkki selkeän poistumistien puuttumisestä on kuvassa 10. jossa on esitetty käyttöliittymäkuva uhkakohtaisten heittokriteerien valintaikkunasta. Aiemmin ikkunasta puuttuivat kokonaan erilliset poistu –napit ja poistuminen piti suorittaa Asiakirja –valikon kautta. Kuvassa siihen on lisätty selkeät poistumisnapit, jotka on kuvassa ympyröity, mutta niiden sijoittelu on jälkikäteen toteutettuna hieman kömpelö.

Heittokriteerien valintaikkuna



Kuva 10. Uhkan tai laitekohtaisen heiteohjelman heittokriteerien valintaikkuna.

4.4 Toiminnallisen määrittelyn käyttäjätesti

Jotta muodostettu määrittely vastaisi toiminnallisuuksiltaan sekä käytettävyydeltään käyttäjien vaatimuksia, päätettiin sille suorittaa käyttäjätesti. Näin saavutettaisiin todennäköisesti merkittävästi parempi lopputulos, mikäli ohjelmisto tulevaisuudessa toteutettaisiin. Heuristisen arvioinnin perusteella tehtyjen, vähäisten muutosten, jälkeen toiminnallinen määrittely oli niin valmis, että käyttäjätestin suorittaminen oli ajankohtaista.

Mahdollisuuksia käyttäjätestien suorittamiseen oli hyvin rajoitetusti, niin testikäyttäjien saatavuuden kuin testaajan oman ajankäytön kannalta. Niinpä kun tilaisuus käyttäjätestin suorittamiseen ilmaantui, oli se hyödynnettävä vaikka ohjelmiston toiminnallinen määrittely ei ollutkaan vielä täysin valmis. Testiä varten tarvittavat käyttöliittymäkuvat sekä ohjelmiston toimintamalli, olivat kuitenkin jo

hyvin yksityiskohtaisesti suunniteltu. Niinpä testisuunnitelman ja itse testin tuottaminen ja järjestäminen olivat silti mahdollisia ja testit oli järkevää järjestää.

4.4.1 Käyttäjätesti menetelmänä

Käyttäjätestissä, mahdollisimman hyvin tuotteen loppukäyttäjää edustava koehenkilö, suorittaa tuotteella etukäteen suunniteltuja tehtäviä. Testin aikana testaajat tekevät havaintoja tuotteen käyttöliittymän toiminnasta sekä siihen liittyvistä puutteista ja käytettävyysongelmista. Käyttäjätesti voidaan suorittaa niin valmiille tuotteelle kuin varhaiselle prototyypillekin. Käyttäjätestit ja heuristiset menetelmät eivät ole toisiaan poissulkevia, vaan toisiaan tukevia menetelmiä ja tuottavat yleensä yhdessä paremman lopputuloksen, kuin mikään menetelmä yksinään. Käyttäjätesteissä ongelmia aiheuttaa testitilanteen luonnottomuus, koska käyttöympäristö ei ole luonnollinen ja testattava tietää osallistuvansa testiin. Tätä voidaan lieventää, lisäämällä tilanteen luonnollisuutta ja kertomalla käyttäjälle testitapahtumasta ja sen kulusta etukäteen. Toinen ongelma on koehenkilön valinta, koska koehenkilön tulisi edustaa loppukäyttäjää mahdollisimman hyvin. (Kuutti 2003, 68-69.)

Mitä aiemmassa vaiheessa ongelmat löydetään, sitä enemmän niiden löytämisestä on jatkossa hyötyä, joten käyttäjätestausta on hyvä tehdä mahdollisimman aikaisessa vaiheessa. Toimivien prototyyppien tuottaminen vie kuitenkin aikaa ja niinpä testaaminen on parasta aloittaa pikatesteillä, joissa käytetään pelkkiä kuvia suunnitellusta käyttöliittymästä. Varsinaista toimintaa ei päästä kokeilemaan, mutta hyvällä kysymyksenasettelulla voidaan saada hyödyllistä tietoa. (Wiio 2004, 218.)

Pikatestauksen aluksi, koekäyttäjälle kerrotaan testin kulusta ja korostetaan testauksen kohdistuvan tuotteeseen eikä käyttäjään. Tämän jälkeen hänelle kuvataan lyhyesti ohjelmiston käyttötilanne jossa hän on ja mitä hänen odotetaan seuraavaksi tekevän. Testaajan pitää varoa kertomasta koekäyttäjälle, miten hän on käyttötilanteeseen päässyt tai miten hän saa annetun tehtävän suoritettua. Koekäyttäjälle annetaan mahdollisuus rauhassa tutustua käyttöliittymään ja kertoa miten hän lähtisi suorittamaan annettua tehtävää. Jos käyttäjä valitsee oikean toimenpiteen, vaihdetaan hänen nähtäväkseen kuva tilanteesta, joka vastaa hänen tekemäänsä toimenpidettä. Jos käyttäjän valitsema toimenpide on väärä, kysytään

häneltä, miksi se tuntui hänestä oikealta ja tarvittaessa myös, harkitsiko hän jotain toista vaihtoehtoa. Käyttäjän annetaan kokeilla uudelleen, mutta jos käyttäjä ei kolmannellakaan yrityksellä osu oikeaan, hänelle osoitetaan oikea valinta, jotta päästään eteenpäin ja testikäyttäjä ei hermostu. Aina kun käyttäjä tekee oikean valinnan tai häntä on autettu eteenpäin, vaihdetaan käyttöliittymäkuva vastaamaan käyttäjän tekemiä valintoja. Jos käyttäjän tekemä muutos on kirjoitusta, radionapin painallus tms. käytettävässä näkymässä tapahtuva muutos, merkitään se koekäyttäjälle näkyviin, mikäli mahdollista. (Wiio 2004, 219-221.)

Pikatestauksessa on tarkoitus testata vain yhtä asiaa ja testin pituus rajoitetaan muutamaan toimintoon. Testi keskeytetään jonkin osatehtävän loppuun. Tällöin voidaan vielä kysyä, miten koekäyttäjä olisi jatkanut käyttöä tai mitä hän olettaisi seuraavien toimenpiteiden olevan. (Wiio 2004, 220.) Testaamisessa on tärkeää, että koekäyttäjän kohdatessa vaikeuksia, häntä ei auteta, vaan pyritään mahdollisimman pitkälti antamaan käyttäjän itse selvittää ratkaisu tilanteeseen (Kuutti. 2003, 75).

Käyttäjätesti on valmisteltava huolellisesi, jotta siitä saadaan luotettavia tuloksia ja eri koekäyttäjät suorittavat testit samalla tavalla. On suunniteltava vähintäänkin taustatarina, jossa kerrotaan koekäyttäjälle mikä on tilanne. Toiseksi on suunniteltava testitehtävä, joka määrittelee lähtötilanteen ja kuvaa sen, mitä käyttäjän on saatava suoritettua. Lisäksi on muodostettava lista kysymyksistä, joita käyttäjälle halutaan esittää testin jälkeen. Luonnollisesti myös kaikki tarvittavat kuvat käyttöliittymän esittämiseen testin suorittamiseksi, täytyy olla muodostettuna. Lisäksi täytyy suunnitella paikka ja esitystapa, jolla käyttöliittymäkuvat koekäyttäjälle esitetään. (Wiio 2004, 222.) Jotta voidaan olla melko varmoja siitä, että testi voidaan viedä suunnitellusti läpi ja että siitä saadaan tarvittavaa tietoa, kannattaa vielä järjestää pilottitesti, jonka perusteella testijärjestelyitä voidaan vielä parantaa (Kuutti 2003, 73). Sen jälkeen on vielä hankittava koekäyttäjät, joiden optimimäärä on 5-6. Lisäksi on sovittava varsinainen testitapahtuma koekäyttäjien ja testaajien kanssa. (Wiio. 2004, 227.)

Käyttäjätestien viimeinen vaihe on tulosten tulkinta. Tulosten tulkitsemiseksi saadut tulokset on koottava yhteen ja järjestettävä, niin että saatuja tietoja on helppo käsitellä. Tämän jälkeen tietoja voidaan käsitellä monin eri tavoin, mutta esimerkiksi samojen ongelmatilanteiden yleisyyttä eri käyttäjillä, voidaan arvioida. Jos ongelma

esiintyi vain yhdellä käyttäjällä, saattaa ongelma olla satunnainen, kun taas saman ongelman ilmeneminen useimmilla käyttäjillä on ilmeinen käytettävyysongelma. Kun ongelma on havaittu, sen alkuperä pitää pyrkiä selvittämään ja samalla arvioitava sen vakavuutta. Nämä kerrotaan testausraportissa, jossa voidaan myös arvioida mahdollisia korjausehdotuksia havaittuihin ongelmiin. (Kuutti 2003, 78-80.)

4.4.2 Käyttäjätestin suunnittelu

Koska käyttäjätestissä ei ollut käytettävissä valmista ohjelmaa, vaan ainoastaan toiminnalliseen määrittelyyn liittyviä sanallisia kuvauksia ja kuvia käyttöliittymästä, suoritettiin käyttäjätestaus niiden avulla. Tämä vastasi siis suoraan Wiion esittelemää pikatestausta, joskin siinä tarkoitus on testata suunnitelmaa jo varhaisilla konsepteilla, nopeasti, ilman erityisjärjestelyitä. Tämä testi taas järjestettiin varsin valmiin ohjelmistosuunnitelman pohjalta. Tällöin on riskinä, että käyttäjätestin perusteella joudutaan tekemään enemmän muutostyötä, varhaisiin konseptitason testeihin verrattuna. Testin toteuttaminen loppukäyttäjillä vaati merkittäviä järjestelyitä, niin testipaikan, kuin -ajankin suhteen. Niinpä oli luonnollista, että koko suunniteltua ohjelmistoa haluttiin testata mahdollisimman kattavasti samalla kertaa. Ohjelmisto kuitenkin jaettiin useaan pienempään testikokonaisuuteen, joilla mahdollistettiin rajattujen kokonaisuuksien testaaminen peräkkäin eri käyttäjillä. Myös tulosten kirjaaminen ja tulkinta helpottui, kun käsiteltiin pienempiä osakokonaisuuksia.

Käyttäjätestiin valmistautuminen aloitettiin, käymällä läpi muodostettu ohjelmistosuunnitelma sekä siihen liittyvät käyttöliittymäkuvat, joiden avulla testi oli tarkoitus toteuttaa. Käyttäjätestistä muodostettiin tutkimussuunnitelma, jonka avulla määritettiin testin toteutustapa ja testin jälkeen testikäyttäjille esitettävät kysymykset. Suunnitelman avulla testit pystyttiin toteuttamaan mahdollisimman hyvin, samanlaisina eri testihenkilöille. Muodostettu tutkimussuunnitelma on esitetty liitteenä 6. Tutkimussuunnitelman perusteella muodostettiin varsinainen testi, käyttäen Microsoft PowerPoint –diaesitystyökalua. Tämä diaesitys on liitteenä 7. mutta osa sen sisältämistä kalvoista on poistettu luottamuksellisuussyistä.

Diaesitystyökalun avulla, käyttöliittymäkuvat pystyttiin näyttämään käyttäjälle testisuunnitelman mukaisesti videotykillä. Samalla testi pystyttiin toteuttamaan

käyttäen ainoastaan yhtä tiedostoa, ilman tarvetta esitellä käyttäjälle erillisiä kuvatiedostoja. Itse diaesityksessä esiteltiin ensin taustatarina ja nykytilanne, sekä kunkin tehtävän alussa seuraavaksi tehtävät toimenpiteet. Tämän jälkeen testikäyttäjälle näytettiin peräkkäisinä käyttöliittymäkuvina, oletetun tehtävien toteuttamistavan mukainen sarja erilaisia käyttöliittymäkuvia. Kaikki käyttäjän antamat syötteet testaaja lisäsi tussilla tussitaululle, jolle diaesitys heijastettiin. Näin kaikki käyttäjän antamat syötteet, voitiin havainnollistaa suoraan käyttöliittymäkuvaan, jolloin päästiin arvioimaan karkeasti ohjelmiston oppimiseen ja käyttämiseen kuluvaa aikaa.

4.4.3 Käyttäjätestin toteutus

Ennen varsinaisten käyttäjätestien aloittamista suoritettiin ns. koetesti, jolla testattiin testijärjestelyn ja erityisesti diaesityksen toimivuutta ja soveltuvuutta varsinaisen käyttäjätestin suorittamiseen. Koekäyttäjäksi saatiin Ilmavoimista insinööri, jolla oli jonkin verran tuntemusta omasuojaheitinjärjestelmistä sekä heitteistä, mutta erittäin vähän kokemusta varsinaisesta omasuojaheittimien ohjelmoinnista ja sekin jo useiden vuosien takaa.

Koetestin aikana ehdittiin aikataulurajoituksista johtuen, käymään läpi ainoastaan testin kolme ensimmäistä vaihetta. Nämäkin riittivät antamaan kuvan testijärjestelyn toimivuudesta ja diaesityksen käyttökelpoisuudesta, käyttäjätestin suorittamiseen. Koetesti osoitti, että diaesitystä voidaan hyödyntää käyttäjätestin suorittamiseen, mutta se vaatii merkittävää panostusta itse diaesityksen muodostamiseen, jotta se toimisi loogisesti ja tuottaisi riittävän mielikuvan ohjelmiston toiminnallisuuksista. Myös tussitaulun hyödyntäminen käyttäjän syötteiden havainnollistamiseen oli toimivaa. Lisäksi diaesitys antoi koekäyttäjälle mahdollisuuden tarkastella ohjelman eri osia ja ikkunoita melko vapaasti, kun hän pystyi liikkumaan diaesityksessä eteen ja taaksepäin sekä tarvittaessa palaamaan tehtävänantoon.

Koetestissä havaittiin myös selviä puutteita, niin itse ohjelmistossa kuin testiä varten muodostetussa diaesityksessäkin. Ensimmäinen tehtävä ei onnistunut ilman tutkijan merkittävää puuttumista, koska käyttäjä ei hahmottanut alustusohjelman tarkoitusta. Siksi ensimmäistä tehtävänantoa ja taustatarinaa piti näiltä osin parantaa. Toisaalta testivaiheiden kaksi ja kolme suorittaminen onnistui nopeasti,

eikä niissä ilmennyt merkittäviä vaikeuksia. Testivaiheessa kolme havaittiin, että heitteiden lisääminen heiteohjelmiin kannattaisi mahdollistaa, yksinkertaisesti hiiren vasenta nappia painamalla. Koetestin perusteella tehtiin pieniä muutoksia itse ohjelmistomäärittelyyn sekä merkittäviä muutoksia testeissä käytettävään diaesitykseen. Ilman koetestiä ja sen perusteella tehtyjä muutoksia, varsinaisen testin toteutus olisi ollut melko vaikeaa ja saadut tulokset olisivat olleet todennäköisesti huonompia. Niinpä koetestillä oli merkittävä vaikutus itse käyttäjätestin onnistumiselle.

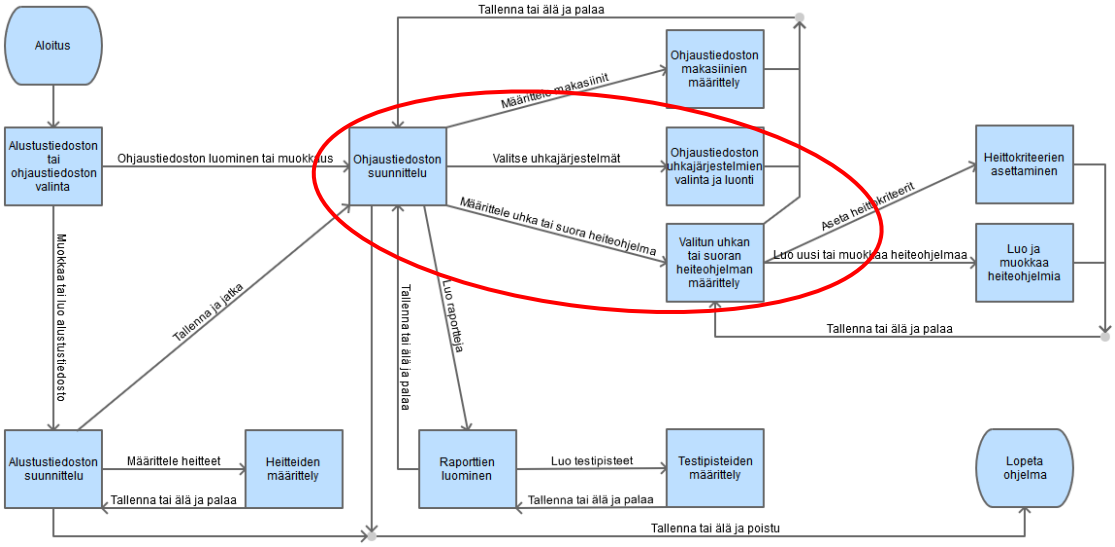
Varsinaiset käyttäjätestit aloitettiin pian sen jälkeen, kun itse diaesitys oli saatu muokattua koetestistä saatujen tulosten perusteella. Varsinaiseen käyttäjätestiin ei valitettavasti saatu samalla kertaa mukaan kuin yksi testikäyttäjä. Kyseinen testikäyttäjä oli Ilmavoimien diplomi-insinööri, jolla oli yli kymmenen vuoden omakohtainen kokemus omasuojaheittimien ohjelmoinnista. Testin aikana käytiin lyhyin tauoin läpi kaikki testisuunnitelman mukaiset testivaiheet. Tuloksia saatiin runsaasti, samoin kuin testikäyttäjän spontaanisti antamia kehitysehdotuksia. Testin kaikki tulokset on kerrottu liitteessä 8.

4.4.4 Käyttäjätestin tulokset

Kun ensimmäisen varsinaisen käyttäjätestin tuloksia käytiin läpi, havaittiin että jo niiden perustella, on syytä tehdä joitakin muutoksia itse ohjelmistosuunnitelmaan. Muutoksia tehtäessä, havaittiin ohjelmistosuunnitelmassa hyvin perustavaa laatua oleva puute. Koko ohjelmiston tavoitteena oli helpottaa yksittäisten heiteohjelmien suunnittelua, sekä tehdä ohjaustiedoston suunnittelemisesta havainnollisempaa, varmistaen samalla koko ohjaustiedoston suunnittelun kattavuus. Kuitenkin käyttäjälle annettiin hyvää, visuaalista tietoa ainoastaan yksittäisten uhkajärjestelmien määrittelyn täydellisyydestä. Koko ohjaustiedostoa ja sen muita osia kuin uhkajärjestelmiä ei huomioitu lainkaan, vaikka se oli yksi koko ohjelmiston tärkeimmistä tavoitteista.

Niinpä ohjelmiston pääkäyttöliittymä jouduttiin suunnittelemaan tässä vaiheessa kokonaan uudelleen, jotta mahdollistettiin koko ohjaustiedoston suunnittelutilanteen havainnointi suoraan pääkäyttöliittymästä. Tämä pakotti siirtämään itse uhkiin ja suoriin heiteohjelmiin liittyvät toiminnot erilliseen ikkunaan,

kun ne olivat aiemmin olleet osana ohjelmiston päänäkymää. Tällöin yksittäisten uhkien ja suorien heiteohjelmien suunnittelutilanne, voitiin esittää pääkäyttöliittymässä samalla tavalla visuaalisesti, kuin aiemmin yksittäisten uhkienkin osalta. Tämä näkyy havainnollisesti jäljempänä olevasta kuvasta 11. missä on koko ohjelmiston uusi näyttökartta. Kuvaan on punaisella ympyröity ne kolme erillistä ikkunaa, joiden toiminnallisuudet olivat vielä testin aikana kaikki samassa ikkunassa. Noiden kolmen eri ikkunan toiminnot ja informaatio oli aiemmin yritetty mahduttaa yhteen ainoaan näkymään. Vielä muutosten jälkeenkin, ohjaustiedoston suunnitteluikkunassa ja uhkakohtaisessa heitemäärittelyikkunassa, käytetään koko näytön ala hyödyksi. On siis ymmärrettävää, että aiemmin käytetyssä ikkunassa ei voitu esittää luontevasti kaikkea tarpeellista informaatiota ja käyttäjän oli vaikea hahmottaa käyttöliitymän kaikkia toiminnallisuuksia.



Kuva 11. Ohjelmiston näyttökartta.

Samalla kun pääkäyttöliittymässä, päätettiin näyttää ohjaustiedoston valmiusastetta suorien heiteohjelmien ja uhkien tasolla, päätettiin myös valikkopuu rajata samalle tasolle ja jättää yksittäiset heiteohjelmat siitä pois. Vastaavasti yksittäisten uhkien tai suorien heiteohjelmien määrittelyikkunassa, päätettiin listata kaikki ohjaustiedoston sisältämät ja siten valittavissa olevat yksittäiset heiteohjelmat, joita ei nyt siis enää näytetty päävalikon hakemistopuussa. Samalla yksittäisten heiteohjelmien kopiointi toisista ohjaustiedostoista, päätettiin toteuttaa ainoastaan tässä samassa näkymässä, koska muualla se ei olisi ollut luontevaa. Koska yksittäisten heiteohjelmien

kopioiminen mahdollistettiin uhkakohtaisessa määrittelyikkunassa, päätettiin niiden kopioiminen samalla poistaa heiteohjelmien määrittelyikkunasta. Tällöin heiteohjelmien määrittelyikkunan toiminnallisuus, voitiin rajata ainoastaan yksittäisten heiteohjelmien luomiseen ja muokkaamiseen.

Koska ohjelmisto ei siis testivaiheessa vastannut alkuperäistä tavoitettaan ja sille asetettuja vaatimuksia, olivat edellä kuvatut muutokset välttämättömiä. Kokonaisuutena muutokset vaikuttivat hyvin merkittävästi koko ohjelmiston käyttötapaan ja toimintaan. Muutokset selkeyttivät ohjelmiston rakennetta ja tekivät pääkäyttöliittymästä huomattavasti intuitiivisemman, sekä mahdollistivat ohjaustiedoston suunnittelun kokonaistilanteen havainnoinnin. Tällä oli merkittävä positiivinen vaikutus koko ohjelmiston ohjelmistoergonomiaan.

Ohjelmistoon tehtiin siis jo ensimmäisen käyttäjätestin jälkeen merkittäviä muutoksia, jotka vaikuttavat ohjelmiston käyttöliittymään ja käyttöön huomattavasti. Siksi käyttäjätestien suorittamista, alkuperäisen suunnitelman ja diaesityksen avulla, ei pidetty enää järkevänä. Niinpä seuraavaksi päätettiin keskittyä muokkaamaan varsinainen ohjelmistosuunnitelma vastaamaan mahdollisimman hyvin lopullista tavoitetilaa. Tässä työssä pyrittiin varmistamaan, ettei vastaavia merkittäviä muutoksia enää tarvittaisi. Tällöin mahdollisissa tulevissa käyttäjätesteissä voitaisiin keskittyä pienempien käytettävyyttä haittaavien ja käyttöä hidastavien tekijöiden havaitsemiseen. Toisaalta haluttiin myös varmistaa, että tulevat käyttäjätestit voitaisiin suorittaa kokonaisuudessaan laajemmalle testaajaryhmälle, ilman tarvetta muutosten tekemiseen testien välillä. Myös käytettävissä olevien koekäyttäjien määrä, rajoittaa osaltaan mahdollisuuksia useiden testien suorittamiseen.

Tässä muutosprosessissa, havaittiin hyvin konkreettisesti sekä ohjelmistoprosessien iteratiivinen luonne, sekä jo aivan suunnittelun alussa tehtyjen virheiden, kostautuminen myöhemmässä vaiheessa. Toisaalta nyt kun ohjelmisto oli vasta suunnitteluasteella, tehtyjen virheiden korjaaminen oli vielä kohtuullisen helppoa ja nopeaa, kun itse koodia ei tarvinnut muuttaa. Tämän kaltainen iteratiivinen, eli samojen asioiden määritteleminen tai tekeminen uudelleen useita kertoja, on hyvin tyypillistä useimmille ohjelmistoprojekteille. Usein ohjelmasta tuotetaan useita peräkkäisiä versioita ohjelmistoprojektin aikana ja jokaisessa otetaan mukaan joitakin uusia ominaisuuksia. Tällöin iteratiivisuus on suunnitelmallista ja se auttaa

varmistamaan aiemmin toteutettujen ominaisuuksien oikean toiminnan. Toisaalta iteratiivisuus saattaa aiheuttaa monia ongelmia, ja esimerkiksi lisätä tarvittavan testauksen määrää. (Haikala 2011. 40-42.)

Usein iteratiivisuus on myös tahatonta ja johtuu vaatimusten hallinnassa, ohjelmiston määrittelyssä tai varsinaisessa ohjelmoinnissa tapahtuneista virheistä tai laiminlyönneistä, jotka tulevat esiin vasta ohjelmistoprosessin myöhemmässä vaiheessa. Myös asiakasvaatimukset saattavat muuttua, ohjelmistoprojektin aikana. Tällöin usein ainoa tapa korjata tilanne, on palata takaisin, muuttamaan tehtyä suunnitelmaa tai ohjelmointia. Tällaisen tapahtumista voi kuitenkin ehkäistä ennalta, varmistamalla asiakasvaatimukset hyvissä ajoin ja määrittelemällä asiat erittäin tarkasti jo mahdollisimman aikaisessa vaiheessa. Mitä väljemmin suunnittelu alussa tehdään, sitä enemmän ongelmia se jatkossa aiheuttaa ja riski muutostarpeiden syntymiselle kasvaa. (Haikala, 2011. 41-42 ja 61.) Juuri näin kävi myös tässä työssä, kun tehty toiminnallinen määrittely ei ollut riittävän yksityiskohtainen ja kattava, jotta sen sisältämä karkea virhe olisi havaittu ajoissa.

Huonoa tässä ensimmäisessä käyttäjätestissä oli se, ettei siinä päästy vielä pureutumaan ohjelmiston varsinaisiin ohjelmistoergonomisiin ominaisuuksiin. Niinpä seuraavassa käyttäjätestissä mielenkiinto on syytä kohdistaa, entistä voimakkaammin, juuri näihin tekijöihin. Kuitenkin tässä vaiheessa, kaikki tämän opinnäytetyön keskeiset tavoitteet oli saavutettu. Toisaalta seuraavakaan käyttäjätesti, ei todennäköisesti tuottaisi vielä kaikkia tuloksia ohjelmistomäärittelyn lopullisen version muodostamiseksi. Niinpä uuden käyttäjätestin määrittäminen ja suorittaminen päätettiin jättää toteutettavaksi myöhemmin, tämän opinnäytetyön ulkopuolella, yhdessä mahdollisen asiakkaan kanssa.

5 Tulokset

Tämän opinnäytetyön keskeinen tavoite oli selvittää vaatimukset ja toteuttamismahdollisuudet ohjelmistolle, jolla pystyttäisiin erityisen hyvän ohjelmistoergonomian avulla parantamaan omasuojaheittimien ohjelmointia. Kuten jo ensimmäisen kyselytutkimuksen tuloksista kohdassa 3.1.3 havaittiin, tämän kaltaisen ohjelmiston uskottiin parantavan omasuojaheittimen ohjelmoinnin loppu-

tulosta ja ohjaustiedostojen laatua sekä lisäävän käyttäjän luottamusta ohjelmoinnin onnistumiseen. Samalla kyselytutkimuksessa sekä kaikissa seuraavissa tutkimuksissa kerättiin käyttäjiltä vaatimuksia joita he ohjelmistolle asettavat. Vaikka kaikissa tutkimuksissa osallistujat edustivat samaa hyvin suppeaa ryhmää, he vastasivat erittäin hyvin myös ohjelmiston tulevaa käyttäjäjoukkoa. Niinpä saatujen tulosten voidaan todeta olevan relevantteja ja kuvastavan hyvin ohjelmiston todellisia vaatimuksia.

Koska samat vastaukset alkoivat toistua eri tutkimuksissa, voitiin todeta että tekemällä lisää tutkimuksia samoilla osallistujilla, ei voitu enää odottaa uusia tuloksia. Toisaalta ilman ulkomaisten osallistujien hankkimista, vastaajajoukkoa ei olisi voitu enää laajentaa niin, että vastaajat olisivat edelleen edustaneet ohjelmiston tulevia käyttäjiä. Niinpä näillä perusteilla todettiin, että vaatimusmäärittelyyn tarvittavat käyttäjävaatimukset oli saatu kerättyä, siinä määrin kuin se oli järkevästi mahdollista. Näiden käyttäjävaatimusten perusteella kyettiin muodostamaan ohjelmistolle vaatimusmäärittely. Työn tavoitteeksi oli asetettu yleiskäyttöisen heitinohjelmistojen suunnittelutyökalun määrittäminen. Koska vaatimuksia kerättiin ainoastaan suppealta käyttäjäryhmältä, yhdestä maasta, ei kerättyjä vaatimuksia voida pitää yleisesti tosina. Niinpä alkuperäinen tavoite tuotetun määrittelyn yleisestä kattavuudesta ei välttämättä toteudu. Kun taas arvioidaan määrittelyn kattavuutta teknisessä mielessä, voidaan todeta että tuotettu määrittely tukee erilaisten omasuojaheittimien ominaisuuksia varsin kattavasti. Näin ollen alkuperäisen tavoitteen mukainen yleinen kattavuus voidaan olettaa ainakin osittain saavutetun.

Tehtyjen tutkimusten perusteella pystyttiin myös vastaamaan asetettuun tutkimuskysymykseen. Tutkimusten perusteella ohjelmisto tuottaa käyttäjilleen hyötyä ja heidän työlleen lisäarvoa, mikäli se täyttää käyttäjien asettamat vaatimukset. Erityisen kriittistä tavoitteen toteutumiselle on erityisen hyvä käytettävyys ja visuaalisuus sekä mahdollisuus tuottaa automaattisesti havainnollistavia kuvia ohjaustiedoston toiminnasta. Lisäksi mahdollisuus tuottaa automaattisesti testipistemäärittelyitä suunnitellulle ohjaustiedostolle, pidettiin tutkimusten perusteella erityisen tärkeänä ominaisuutena.

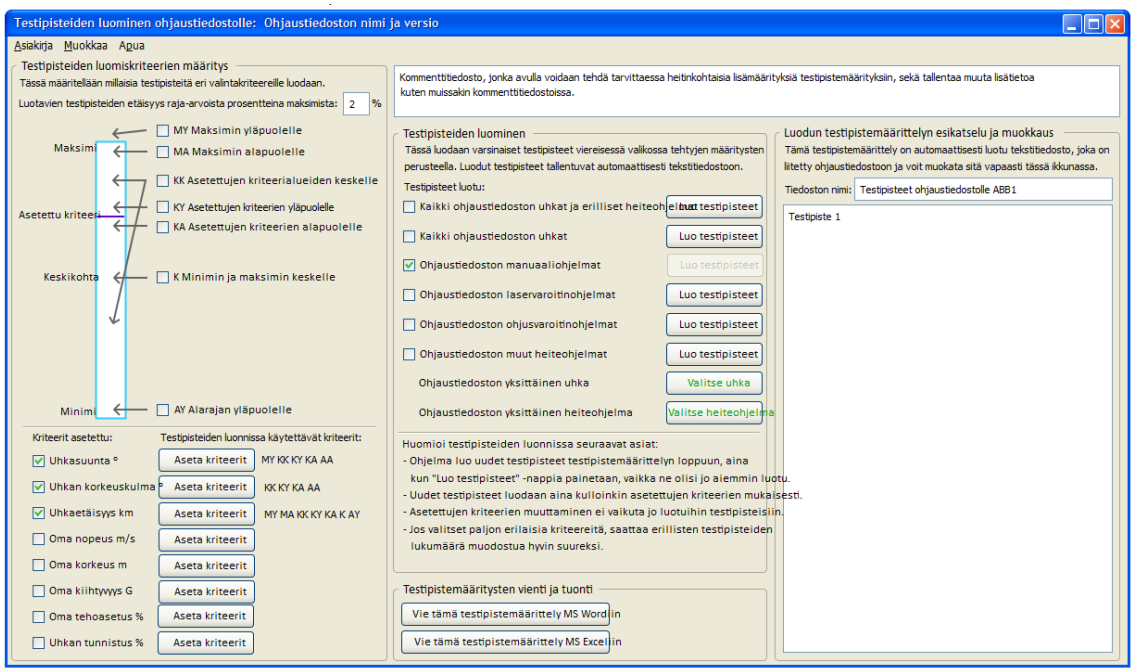
Työn toisessa vaiheessa toteutettiin ohjelmistolle vaatimusmäärittelyyn perustuva toiminnallinen määrittely, joka täyttää keskeiset käyttäjävaatimukset. Koska työn tavoitteena oli vähentää nykyisten ohjelmointityökalujen huonosta

ohjelmistoergonomiasta aiheutuvia haittoja, keskityttiin toiminnallisessa määrittelyssä erityisesti hyvien ohjelmistoergonomisten ratkaisujen määrittelemiseen. Samalla jätettiin suoraan käyttäjälle näkymättömät tekniset yksityiskohdat määrittelemättä. Niinpä toiminnallinen määrittely ei sellaisenaan ole vielä täydellinen, eikä sen perusteella voida vielä suoraan siirtyä ohjelmoimaan määritettyä ohjelmistoa.

Tuotetun toiminnallisen määrittelyn perusteella oli kuitenkin mahdollista toteuttaa käyttäjätutkimus määritellylle ohjelmistolle. Käyttäjätutkimuksen tulosten mukaan ohjelmisto täyttää keskeiset käyttäjävaatimuksissa esiintyvät toiminnot. Ohjelmiston parhaina ominaisuuksina pidettiin visuaalisuutta, erityisesti yksittäisten heiteohjelmien suunnittelussa ja niiden havainnollistamisessa. Lisäksi ohjelmistoa pidettiin keskivertoa helppokäyttöisempänä. Ohjelmisto ei kuitenkaan ohjelmistoergonomisten ominaisuuksiensa osalta ollut tavoitellulla tasolla ja siinä havaittiin karkea ohjelmistoergonominen puute. Niinpä käyttäjätestit keskeytettiin ja ohjelmiston toiminnalliseen suunnitelmaan tehtiin merkittäviä muutoksia. Tehtyjen muutosten jälkeen, uusia käyttäjätestejä ei enää suoritettu.

Koska toiminnalliseen määrittelyyn tehtiin merkittäviä muutoksia käyttäjätestin jälkeen, sen tulosten perusteella ei voida antaa luotettavaa lausuntoa toiminnallisen määrittelyn onnistuneisuudesta. Toiminnallisessa määrittelyssä pystyttiin täyttämään käyttäjien keskeisimmät erilliset vaatimukset, mutta se ei vastannut ohjelmistoergonomisilta ominaisuuksiltaan tavoiteltua tasoa. Niinpä voitiin todeta myös toisen tutkimuskysymyksen tulleen ainakin pääosin vastatuksi. Toisaalta toiminnalliseen määrittelyyn tehtyjen muutosten voidaan olettaa vaikuttavan positiivisesti määritellyn ohjelmiston ohjelmistoergonomiaan. Kuitenkin uuden tutkimustiedon puuttuessa voitiin todeta, että ohjelmistossa ei ainakaan vielä tavoitettu erityisen hyvää ohjelmistoergonomiaa. Työn jatkaminen ilman asiakasta ja heidän kanssaan jaettua ymmärrystä ohjelmiston tavoitteista, ei ole kuitenkaan kannattavaa. Opinnäytetyössä tuotetun toiminnallisen määrittelyn taso mahdollistaa työn jatkamisen yhdessä mahdollisen asiakkaan kanssa, jolloin yhteisen ymmärryksen muodostuessa myös asiakkaan piilevät tarpeet tulevat esiin. Tämä varmistaa että kaikki asiakkaan vaatimukset ovat mukana, kun toiminnalllisen määrittelyn ohjelmistoergonomisia ratkaisuja viimeistellään.

Esimerkiksi testipistemääritysten tekemisen osalta, tarvitaan vielä yhteistyössä tehtyjä tarkennuksia, jotta se vastaisi juuri käyttäjän tarpeisiin. Näiden testipistemääritysten tekemisen tavasta, ei saatu lainkaan tietoa tehdyissä tutkimuksissa. Siksi nyt tehty suunnitelma testipisteiden määrittämisestä, voi toimia ainoastaan lähtökohtana jatkotyöskentelylle, yhdessä asiakkaan kanssa. Nykyinen suunnitelma testipistemääritysten tekemiseen tarkoitetusta ikkunasta, on esitetty seuraavassa kuvassa 12.



Kuva 12. Testipisteiden määrittelyikkuna.

Opinnäytetyössä onnistuttiin vastamaan asetettuihin tutkimuskysymyksiin sekä tuottamaan tavoitteen mukaiset tuotteet. Opinnäytetyöprosessin tuloksena syntyivät vaatimusmäärittely sekä toiminnallinen määrittely ohjelmistosta, joka voidaan tarvittaessa toteuttaa. Toiminnallisessa määrittelyssä ei vielä tässä vaiheessa saavutettu riittävää ohjelmistoergonomian tasoa. Muodostamalla jaettu ymmärrys ohjelmiston tavoitteista ja kehittämällä toiminnallista määrittelyä edelleen, yhteistyössä asiakkaan kanssa, voidaan vaadittava ohjelmistoergonomin taso kuitenkin saavuttaa. Opinnäytetyössä saatujen tulosten luotettavuutta heikentää tehtyjen tutkimusten pieni osallistujamäärä, mutta saatuja tuloksia voidaan pitää hyvin relevantteina, koska vastaajajoukko on hyvin edustava.

Yleinen heitinohjelmistojen suunnittelutyökalu pystytään toteuttamaan, opinnäytetyön tuloksena tuotetun toiminnallisen määrittelyn perusteella. Sen todellisia vaiku-

tuksia omasuojaheittimien ohjelmoinnin lopputulokseen, ei voida kuitenkaan luotettavasti arvioida, ennen kuin se on olemassa. Vasta silloin voidaan käytännössä testata rinnakkain, suunnittelutyökalun avulla toteutettua ja nykyistä ohjelmointiprosessia, sekä verrata niiden nopeutta ja tuloksia toisiinsa. Suunnittelutyökalun vaikutuksia sotilaslentokoneiden todelliseen selviytymiseen taistelussa, ei toivottavasti ajauduta koskaan todentamaan.

6 Pohdinta

Työn tavoitteena oli kerätä käyttäjien vaatimukset ja tuottaa toiminnallinen suunnitelma ohjelmistolle, jolla voitaisiin suunnitella omasuojaheittimen ohjaustiedostoja. Tavoitteen asettamisvaiheessa tavoitteet vaikuttivat hyvin selkeiltä ja helpohkoilta toteuttaa. Hyvin pian kävi kuitenkin selväksi että muodostettava ohjelmisto onkin huomattavasti laajempi ja monimutkaisempi kuin miltä se ensin vaikutti. Niinpä määrittelyyn jouduttiin tekemään rajauksia. Esimerkiksi tietokantamäärittelyitä ei tehty osana tätä työtä. Myös julkisesti saatavilla olevat tekniset tiedot erilaisista omasuojaheittimistä olivat osittain pintapuolisia ja kotimainen asiantuntemus alalta kovin niukkaa. Niinpä tarvittavan pohjatiedon hankkimiseksi tarvittiin useita eri tutkimuksia ja opinnäytetyön työmäärä kasvoi suureksi.

6.1 Opinnäytetyöprosessiin liityviä haasteita

Vaikka opinnäytetyön sisältöä rajattiinkin merkittävästi jo aikaisessa vaiheessa, sen kokonaisuus muodostui lopulta suorastaan valtavaksi. Siksi työssä ei ollut mahdollista paneutua riittävän syvällisesti kaikkiin käsiteltäviin teemoihin, vaikka työn tekemiseen käytettiinkin aikaa lähes kaksi vuotta. Syvällisempi tekninen suunnittelu olisi vaatinut vielä merkittävästi laajempaa tutkimusta ja lisännyt työmäärää niin, ettei tämä työ olisi voinut valmistua. Samoista syistä ohjelmiston toiminnallisessa suunnitelmassa ei kyetty soveltamaan kaikkia ohjelmistoergonomiaan, käytettävyyteen ja käyttöliittymäsuunnitteluun liittyviä standardeja ja periaatteita, eikä saavuttamaan kaikkia opinnäytetyölle asetettuja tavoitteita. Prosessi opetti että

tutkimuksen rajaamiseen pitääkin käyttää enemmän aikaa ja sen tekemiseen pitää pyytää tukea.

Haasteita työn tekemiseen aiheutti myös se, että ohjelmistoergonomiasta oli käytettävissä vain joitakin hyviä lähdemateriaaleja, vaikka se ei käsitteenä olekaan aivan uusi. Myöskään ohjelmistojen käytettävyydestä tai käyttäjäystävällisyydestä ei muodostunut kovin laajaa lähdeaineistoa. Lisäksi erilaisten standardien saatavuus käyttöön ilman rahallista korvausta oli yllättävän huono. Opinnäyetyön laajuuden ja erillisten tutkimusten suuren määrän takia, teoreettisen tietämyksen käsittely opinnäytetyön raportissa haluttiin jättää kohtuullisen niukaksi. Teoriatiedosta haluttiin tuoda esiin vain olennaisin, juuri kyseisen asian ymmärtämiseksi tarvittava minimitieto.

Teoriatiedon ja itse työn etenemisen kuvauksen lomittaminen, synkronisen raportointitavan mukaisesti, osoittautui luettavuuden kannalta hyvin toimivaksi ratkaisuksi. Toisaalta joissain tapauksissa uuden teoriatiedon kerääminen vasta edellisen vaiheen jälkeen oli huono vaihtoehto. Tällöin uusi tieto saattoi koskea myös aiheita, joita oli käsitelty jo aiemmissa vaiheissa, eikä tätä uutta tietoa kyetty aina hyödyntämään aiemmassa vaiheessa tehtyyn työhön.

6.2 Opinnäytetyöprosessin lopuksi

Opinnäytetyön aihe valikoitui työelämässä tehdyn käytännön havainnon ja siihen liittyvän mielenkiinnon perusteella. Työ tehtiin omalla ajalla ja ilman tilaajan tukea. Nämä tekijät rajoitttivat merkittävästi työhön käytettävissä olevaa aikaa ja omasuojaheittimiin perehtyneiden henkilöiden käytettävyyttä prosessissa. Myös kirjoittajan osaaminen ohjelmistotuotannon alalta osoittautui liian suppeaksi. Haasteista huolimatta osa opinnäytetyön tavoitteista saavutettiin ja tuotetut tuotteet ovat hyödynnettävissä, mikäli niille löytyy asiakas.

Kirjoittaja on opintojensa aikana mm. tullut kaksi kertaa isäksi, remontoinut yhden talon ja asunnon sekä muuttanut uuteen työpaikkaan, toiseen maakuntaan. Näiden prosessien keskellä ajan ja motivaation löytäminen opiskelulle ja opinnäytetyön tekemiselle on ollut ajoittain haastavaa. Opinnäytetyön valmistuminen onkin siis suuri helpotus ja henkilökohtainen saavutus.

Haluan kiittää läheisiäni heidän osoittamastaan tuesta työtäni kohtaan, samoin kuin opinnäytetyöni ohjaajia uhraamastaan ajasta ja neuvoista.

Lähteet

Advanced Countermeasures Dispenser System. n.d. Esite Terma A/S:n verkkosivustolla. Viitattu 19.10.2015. http://www.terma.com/media/291474/advanced_countermeasures_dispenser_system.pdf.

Ahtee, T. muut. 2012. Toiminnallinen määrittely. versio 2.7. Verkkosivu. Viitattu 11.1.2016. http://www.cs.tut.fi/ohj/dokumenttipohjat/

ALE-47 Airborne Countermeasures Dispenser System. 2008. n.d. Esite BAE Systems:n verkkosivustolla. Viitattu 19.10.2015. http://www.baesystems.com/en/product/ale47-airborne-countermeasures-dispenser-system

BOL FOR F/A-18 ADVANCED COUNTERMEASURE DISPENSER. n.d. Esite SAAB AB:n verkkosivustolla. Viitattu 8.5.2016. http://saab.com/air/electronic-warfare/countermeasure-dispenser-systems/bol-fa-18/

Chaff (Countermeasure). 2015. n.d. Artikkeli Wikipedia:n verkkosivustolla. https://en.wikipedia.org/wiki/Chaff_(countermeasure). Viitattu 13.10.2015.

Cooper, A., Reimann, R. & Cronin, D. 2007. About Face 3, Indianapolis: Wiley Publishing Inc

ECDS Enhanced Countermeasures Dispensing System. 2009. n.d. Esite MES S.p.A Roma:n verkkosivustolla. Viitattu 21.10.2015. http://www.mesroma.it/ae/ae_iss2.htm

Haikala, I & Märijärvi, J. 2004. Ohjelmistotuotanto, 10. uudistettu painos, Hämeenlinna: Talentum Media Oy

Haikala, I & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt, 12. uudistettu painos, aiemmin nimellä Ohjelmistotuotanto, Hämeenlinna, Talentum Media Oy

IEEE Std 830. 1998. IEEE Recommended Practise for Software Requirements Specifications. Software Engineering Standards Committee of the IEEE Computer Society. New York, The Institute of Electrical and Electronics Engineers, Inc.

INTERNATIONAL STANDARD ISO 14915-1. First edition 2002-11-01. n.d. Software ergonomics for multimedia user interfaces — Part 1: Design principles and framework

INTERNATIONAL STANDARD ISO 9241-210:2010(en). First edition. n.d. Ergonomics of human-system interaction — Part 210: Human-centered design for interactive systems

Jackson, D. 2004. Missile Countermeasures. Verkkosivu. Viitattu 5.5.2016. http://www.aerospaceweb.org/question/electronics/q0191.shtml

Kuutti, W. 2003. Käytettävyys, suunnittelu ja arviointi. Helsinki, Talentum Media Oy

Nielsen, J. 1995. 10 Usability Heuristics for User Interface Design. Viitattu 11.1.2016. https://www.nngroup.com/articles/ten-usability-heuristics/

Pollock, D. 1993, The Infrared & Electro-Optical Systems Handbook volume 7 Countermeasure Systems. Washington, Infrared Information Analysis Center & SPIE Optical Engineering Press

Ronkainen, S., Pehkonen, L., Lindblom-Ylänne, S. & Paavilainen, E. 2011. Tutkimuksen voimasanat, Helsinki: WSOYpro oy

Saariluoma, P. 2004. Käyttäjäpsykologia, Vantaa, WSOY

Sinkkonen, I., Kuoppala, H., Parkkinen, J. & Vastamäki, R. 2006. Käytettävyyden psykologia, 3. uudistettu painos, Helsinki, Edita Publishing Oy

Stimson, G., Griffiths, H., Baker, C. & Adamy, D. 2014. Introduction to Airborne Radar, Third Edition, SciTech Publishing

Ulrich, K. & Eppinger, S. 2000, Product Design and Development, 2. uudistettu painos, Boston: McGraw-Hill

US Navy Naval Research Laboratory. n.d. Verkkosivu. Viitattu 13.10.2015. https://www.denix.osd.mil/denix/Public/Library/Rfchaff/Images/images.html

VICON 78 Series 455 Countermeasures Dispensing Systems. n.d. Esite Thales Group:n verkkosivustolla. Viitattu 21.10.2015. https://www.thalesgroup.com/sites/default/files/asset/document/1-%20cmds-vicon-78_va.pdf

Vilkka, H. 2006. Tutki ja havainnoi, Helsinki, Kustannusosakeyhtiö Tammi

Wiio, A. 2004. Käyttäjäystävällisen sovelluksen suunnittelu, Helsinki: Edita Publishing Oy

Liitteet

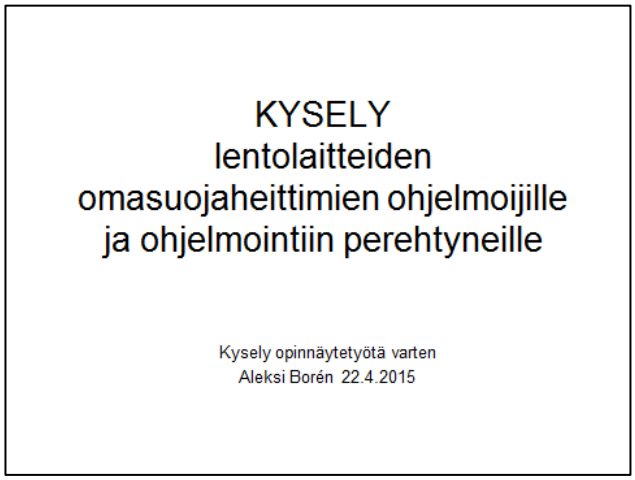
Liite 1. Käyttäjäkyselyn saateviesti ja taustadiat

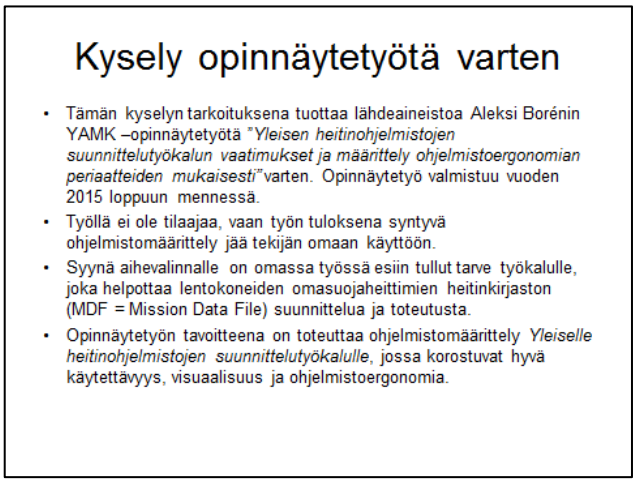
Saateviesti:

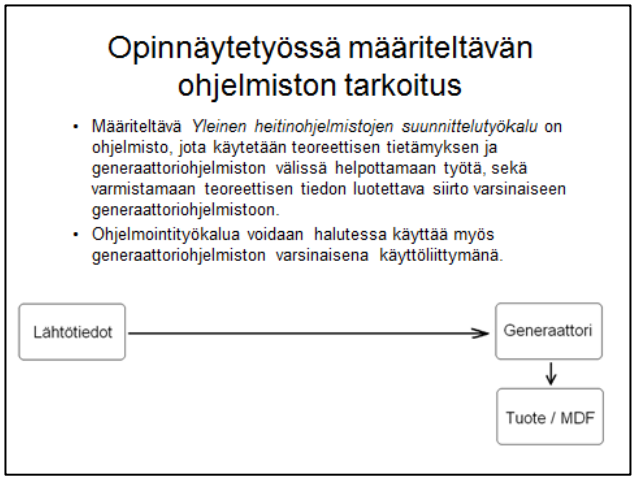
Tämän viestini liitteenä on, osana Aleksi Borénin YAMK -opinnäytetyötä toteutettava, omasuojaheittimien ohjelmointityökaluihin liittyvä kysely, johon toivoisin sinun vastaavan. Olisin todella kiitollinen, jos voisit käyttää hieman aikaasi ja vaivaa vastaamiseen, koska koko opinnäytetyöni pohja perustetaan tämän kyselyn vastauksiin.

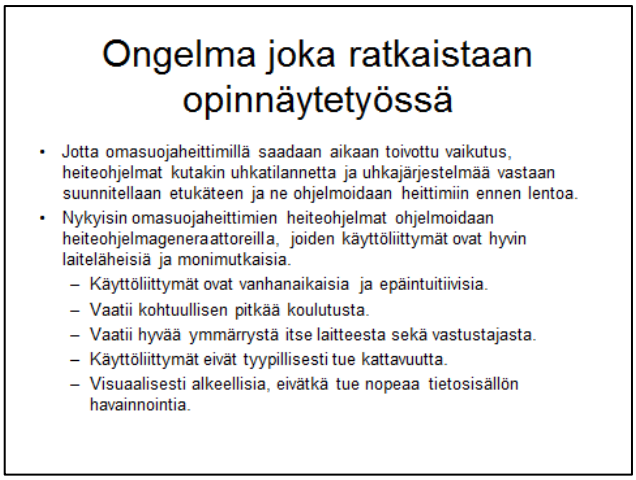
Kyselyn varsinainen alustus sekä kysymykset luettelona löytyvät liitteenä olevasta diaesityksestä ja valmis vastauspohja on liitteenä Word -asiakirjana. Kyselyyn vastaaminen kestää minimissään noin 10 minuuttia ja toivon että palautat vastauksesi 30.5.2015 mennessä.

Taustadiaesitys:









Opinnäytetyön lopputuote, ohjelmistomäärittely

- *Yleisen heitinohjelmistojen suunnittelutyökalun* tarkoituksena on helpottaa varsinaisen heiteohjelmiston muodostamista nykyisiä generointiohjelmia käyttäen.
- Määriteltävä työkalu tekee työstä:
  - Kattavaa, Helppoa, Visuaalista, Nopeaa ja Luotettavaa
- Ohjelmistomäärittely sisältää yksittäisten näyttöjen ja toiminnallisuuksien kuvaukset niiltä osin kuin ne ovat lopputuloksen kannalta olennaisia.
- Pääpainona työssä on määritellä työkalun tuottama käyttäjäkokemus
  - Olennaista on kaikki käyttäjälle näkyvä ja käyttäjän kokema toiminnallisuus.
  - Työssä ei oteta kantaa käyttäjälle näkymättömiin teknisiin yksityiskohtiin.
- Opinnäytetyössä tuotetaan ensin vaatimusmäärittely, jonka pohjalta muodostetaan varsinainen ohjelmistomäärittely

Yleistä kyselystä

- Tämä kysely on kohdistettu henkilöille, joilla on teoreettista tietämystä tai omakohtaista kokemusta omasuojaheittimien ohjelmoinnista ja siihen liittyvistä haasteista.
- Kyselyllä selvitetään *Yleisen heitinohjelmistojen suunnittelutyökalun* hyödyllisyyttä lentokoneiden omasuojaheittimien ohjelmoinnissa sekä erityisesti siihen kohdistuvia käyttäjävaatimuksia.
  - Vastauksia käytetään pohjana ryhmähaastattelulle, jossa saatuja vastauksia varmistetaan ja syvennetään, tämän kyselyn vastaanottajien kanssa.
  - Kerättyä aineistoa käytetään ohjelmiston käyttäjävaatimusten määrittämiseen.
- Kyselyn toteuttamiseen ei ole pyydetty lupaa, vaan sen toteuttaa Aleksi Borén yksityishenkilönä yksityishenkilöille ja saadut vastaukset ovat vapaasti kyselyn toteuttajan käytettävissä.
- Jokaisen kyselyn saaneen vastaaminen on erittäin tärkeää jotta:
  - Käyttäjävaatimukset ja nykyisten ohjelmointivälineiden ongelmakohdat tulisivat mahdollisimman kattavasti esiin.
  - Ohjelmistomäärittelyssä voidaan keskittyä ongelmien poistamiseen ja keskeisten käyttäjävaatimusten toteuttamiseen.

Ohjeet vastaamiseen

- Vastaa tähän kyselyyn antamalla vapaamuotoisia vastauksia kuhunkin kyselyn kohtaan pitäen mielessä edellä kerrotut tavoitteet.
- Erityisesti käyttäjävaatimuksia ja nykyisten ohjelmien ongelmia koskeviin kysymyksiin odotetaan mahdollisimman monia eri vastauksia.
- Helpoin tapa vastata, on käyttää käyttää liitteenä olevaa valmista Word -vastauspohjaa ja lisätä omat vastaukset suoraan kysymysten alle. Voit halutessasi lisätä palautuskaavakkeeseen nimimerkin käsittelyn helpottamiseksi.
- Erottele toisistaan erilliset asiat esimerkiksi erillisillä ranskalaisilla viivoilla tai vastaavalla itsellesi luontevalla tavalla.
- Toimita omat vastauksesi sähköpostin liitteenä samaan osoitteeseen, josta tämä kysely on lähetetty, eli osoitteeseen aleksi.boren@gmail.com

Kysymykset

Jos käytössäsi olisi paras mahdollinen heitinohjelmistojen suunnittelutyökalu…

1. Mikä voisi olla tämän työkalun tuottama tuote?
2. Mitä nykyisten ohjelmien ongelmia tällainen työkalu voisi korjata?
3. Mitä muita asioita työkalulla pitäisi voida tehdä?
4. Mitä vaatimuksia asettaisit työkalulle?
5. Mikä on mielestäsi kriittisin hyöty mikä kyseisestä työkalusta olisi saatavissa?
6. Lisäisikö vai vähentäisikö työkalu työmääräsi uusien ohjaustiedostojen tuottamisessa?
7. Parantaisiko työkalu luottamustasi ohjelmoinnin onnistumiseen?
8. Mitä työkalulla pitäisi vähintäänkin pystyä tekemään, jotta siitä olisi sinulle hyötyä?
9. Millaisia raportteja työkalun pitäisi tuottaa?
10. Miten uskoisit työkalun käyttöönoton vaikuttavan heiteohjelmien tuottamiseen?
11. Miten voisit hyödyntää työkalua työssäsi?
12. Parantaisiko työkalu työsi lopputulosta?
13. Olisiko tällaiselle työkalulle mielestäsi tarvetta?

Kiitos vastauksistasi!

- Myös muu palaute sekä vinkit opinnäytetyöhön liittyen otetaan kiitollisuudella vastaan.

Liite 2. Koonnos käyttäjäkyselyn vastauksista

1. Mikä voisi olla tämän työkalun tuottama tuote?

- Tuotteita olisi kolme:
    - MDF generaattorille kelvollinen määrittely, jonka pohjalta MDF-generaattori voisi luoda valmiin MDF:n
    - Testipistematriisi MDF:n oikeellisuuden varmentamiseksi.
    - Graafinen käyttöliittymä, jossa näkyisi mahdollisimman havainnollisesti MDF:ään määriteltävät atribuutit, heitesekvenssit, asetetut ML/MA ja prioriteettiarvot.
- varsinaisen generointiohjelmiston input-parametrit ainakin osittain
- raportit versiokuvaukseen
- historiatieto suunnittelun perusteista

2. Mitä nykyisten ohjelmien ongelmia tällainen työkalu voisi korjata?

- Graafinen käyttöliittymä auttaisi tekemään helpommin, nopeammin ja täsmällisemmin MDF-tiedostoja.
- visuaalisuus
- apu versiokuvausta varten (esim. raportit)
- jäisi tieto siitä miksi ko. heiteohjelmiin on päädytty

3. Mitä muita asioita työkalulla pitäisi voida tehdä?

- Tuottaa havainnollistavia kuvia MDF:n toiminnasta versiokuvaukseen.
- apu testausta varten (testipisteet)
- Yhteenvedot ohjaustiedostoista

4. Mitä vaatimuksia asettaisit työkalulle?

- Toiminnallisesti sen pitäisi olla parempi GUI suoraan MDFG:lle.
- Laadullisesti sen uudelleen käytettävissä (eli kullekin uhkajärjestelmälle ohjelmoituvaste tulisi olla noudettavissa tietokannasta tai vastaavasta) ja sen pitäisi olla helposti ylläpidettävissä jopa käyttäjäryhmän toimesta.
- visuaalisesti selkeä ja miellyttävä
- helppokäyttöisyys (intuitiivisyys, ohjeet helposti saatavilla)

- korkea luotettavuus
- yhteensopivuus asiayhteyteen liittyvien ohjelmien kanssa
- muistaa käyttäjän tekemät valinnat esim. hakemistojen suhteen

5. Mikä on mielestäsi kriittisin hyöty mikä kyseisestä työkalusta olisi saatavissa?

- Ohjelmoinnin helpottuminen yksityiskohtien näpertelystä kohti operatiivisen kokonaistilanteen hallintaa. Tämä tapahtuu pitkälti graafisen käyttöliittymän myötä.
- yhdenmukaistaa eri ohjaustiedostojen suunnittelun

6. Lisäisikö vai vähentäisikö työkalu työmäärääsi uusien ohjaustiedostojen tuottamisessa?

- Ajan mittaa vähentäisi, koska tunnettujen uhkien vastatoimet voisi ottaa käyttöön suoraan.
- luultavasti lisäisi

7. Parantaisiko työkalu luottamustasi ohjelmoinnin onnistumiseen?

- Kyllä parantaisi.
- kyllä

8. Mitä työkalulla pitäisi vähintäänkin pystyä tekemään, jotta siitä olisi sinulle hyötyä?

- Havaintokuvia versiokuvaukseen sekä testipistematriisi.
- visualisoida ja raportoida heiteohjelmat versiokuvausta varten

9. Millaisia raportteja työkalun pitäisi tuottaa?

- Raportti halutun heitesekvenssin määrittelyn siirtymisestä
- visualisoida ja raportoida heiteohjelmat versiokuvausta varten
- testipisteet
- eri ohjaustiedostojen sisältämät kohdejärjestelmät
- jatkokehitysehdotukset

10. Miten uskoisit työkalun käyttöönoton vaikuttavan heiteohjelmien tuottamiseen?

- Se luultavasti parantaa ohjaustiedostojen laatua.

- yhdenmukaistaa eri ohjaustiedostojen suunnittelun
- mahdollistaa jälkikäteen suunnittelun perusteiden tarkastamisen ja sitä kautta helpottaa jatkokehitystä

11. Miten voisit hyödyntää työkalua työssäsi?

- Nykyisessä työssäni sillä ei ehkä olisi mitään hyötyä. Ohjelmoinnista vastaavassa yksikössä vastaavasti kyllä.
- Ohjaustiedostojen suunnittelussa, toteutuksessa ja dokumentoinnissa

12. Parantaisiko työkalu työsi lopputulosta?

- Nykyisessä työssäni ei. Ohjelmoinnista vastaavassa yksikössä vastaavasti kyllä.
- voisi parantaa jos sen käyttö on miellyttävää eikä tunnu siltä että samoja asioita pitää naputella useisiin ohjelmiin

13. Olisiko tällaiselle työkalulle mielestäsi tarvetta?

- Ohjelmoinnista vastaavan yksikön kannalta kyllä.
- mahdollisesti

Liite 3. Haastattelututkimuksessa tehdyt huomiot

- Ohjelmistolta odotetaan korkeaa luotettavuutta. Tämän ymmärretään tarkoittavan ohjelmiston toiminnallista luotettavuutta, eli ohjelmisto ei kaatuile ts. sen reagointi käyttäjän toimenpiteisiin ei yllättäen lakkaa. Lisäksi luotettavuutena nähtiin myös se, että ohjelmisto aina tosiasiallisesti tallentaa omaan tietokantaansa kaikki käyttäjän tekemät valinnat ja määritykset, niin ettei käyttäjän tarvitse koskaan erikseen varmistaa tallennuksen toteutumista ja että ohjelmiston toiminta on aina toistettavaa, eli samoilla käyttäjän toimenpiteillä ohjelmisto tuottaa aina saman lopputuloksen.
- Kun tietoa kopioidaan aiemmin luodusta ohjaustiedostomäärittelystä, käyttäjän halutaan saavan siitä raportin, jossa kerrotaan kopioimisen onnistumisesta sekä siihen mahdollisesti liittyvistä puutteista tai rajoitteista.
- Kun jostakin aiemmin luodusta kokonaisuudesta muodostetaan uusi versio, pitää ohjelmiston pyydettäessä pystyä tuottamaan raportti eri versioiden välisistä eroista.
- Kopioiminen ohjaustiedostomäärittelyn sisällä pitää voida suorittaa käyttäen standardeja Windows näppäinyhdistelmiä, kuten Ctrl + c ja Ctrl + v. Myös palaaminen aiempaan tilaan ohjaustiedostomäärittelyssä, pitää voida suorittaa käyttäen standardeja Windows näppäinyhdistelmiä, kuten Ctrl + z.
- Uusia versioita muodostetaan ainoastaan käyttäjän omin toimenpitein, hänen itse muodostamansa versiointimääritelmänsä mukaisesti, käyttäen ohjelmistomäärittelyn nimeä ja mahdollisesti erillistä versionumerokenttää. Ohjelmiston ei siis haluta muodostavan uusia versioita automaattisesti.
- Ohjelmiston halutaan liittävän automaattisesti kokonaiseen ohjaustiedostomäärittelyyn sekä yksittäisiin heitesekvensseihin tiedon sen tekijästä käyttöoikeustiedon mukaisesti. Jos ohjaustiedostomäärittelyä tai heitesekvenssiä muutetaan, myös tekijän nimi vaihtuu, mutta kyseisen osakokonaisuuden aiempien muokkaajien nimet ovat kuitenkin näkyvissä vapaasti

muokattavassa kommenttikentässä, joka on linkitetty kyseiseen osakokonaisuuteen.

- Ohjelmiston kaikissa kentissä, joihin käyttäjä voi vapaasti syöttää haluamansa lukuarvon tai muun tiedon, pitää olla oikeellisuustarkistus, jossa tarkastetaan annetun tiedon oikea formaatti sekä pysyminen annetuissa raja-arvoissa. Kulloinkin käytettävissä oleva arvotyyppi sekä sen rajat, tulee näkyä jokaisen syöttökentän yhteydessä joko jatkuvasti tai siirryttäessä kyseiseen syöttöalueeseen.
- Kun käyttäjä on poistumassa jonkin osakokonaisuuden syöttötilasta tai aloittaa raporttien generoinnin, ohjelmisto raportoi käyttäjälle kaikista sellaisista syöttökentistä, joita käyttäjä ei ole joko täyttänyt tai muuttanut automaattisen täytön jälkeen.
- Edellinen kohta sisältää siis myös vaatimuksen mahdollisuudesta täyttää kenttiä automaattisesti. Eli esimerkiksi asetettaessa uusia uhkakohtaisia kriteerejä, tulee kaikki uudet tilannevaihtoehdot täyttää automaattisesti valitulla heiteohjelmalla tai ns. sisarohjelmalla, eli samalla heiteohjelmalla jota käytettiin siinä tilanteessa, josta uusi heittotilanne on kopio muuttunutta kriteeriä lukuun ottamatta.
- Kun käytetään automaattista heiteohjelmien täyttöä, sillä täytetyt heiteohjelmat merkitään omalla taustavärillä. (esim. sininen) Erikseen muodostetut heiteohjelmat merkitään jollain toisella taustavärillä (esim. keltainen), määrittelemättömät heiteohjelmat punaisella taustavärillä länsimaisen väristandardin mukaisesti jne. Pitäisikö ns. varmistettu tai hyväksytty heiteohjelma värittää vihreäksi vai pitäisikö olla jokin erillinen checkbox jokaisen heiteohjelman kohdalla varmistuksena että ko. heiteohjelma on ns. valmis? Tämän ns. hyväksynnän puute ei kuitenkaan voi olla ohjaustiedoston generoimisen este. Ratkaisuna voisi olla ns. positiivinen lukitus, joka on heiteohjelman edessä oleva checkbox jolla lukitaan kyseinen heiteohjelma niin, että sen sisältöä ei voi muuttaa niin kauan kuin checkbox on merkittynä. Samalla merkityn heiteohjelman väri muuttuu vihreäksi merkkinä siitä että se on valmis eikä sen sisältöä haluta muuttaa. Muuttaminen on kuitenkin heti jälleen mahdollista, jos merkintä poistetaan. Merkintä ei automaattisesti kopioidu toiseen kohtaan, vaikka kyseinen

heiteohjelma kopioidaan siihen. Kun merkintä poistetaan, muuttuu kyseinen heiteohjelma keltaiseksi, riippumatta siitä mikä sen väri oli ennen merkintää, mikäli se kuitenkin sisältää kriteerit täyttävän heiteohjelman. Aina jos jonkin heiteohjelman määrittely on vajaa tai muuten kriteerien vastainen, sen väri on punainen ja kyseistä ohjaustiedostoa ei voi generoida. Keltainen ja vihreä väri tulee valita niin, etteivät ne sekoitu toisiinsa puna-viher-värisokean henkilön käyttäessä ohjelmaa, esimerkiksi niin että vihreä väri on sävyltään selkeästi keltaista tummempi ja siten selkeästi erottuva.

- Ohjelmistossa tulee olla mahdollisuus liittää kuhunkin ohjaustiedostomäärittelyyn henkilökohtainen vapaatekstiosa, jonka voi avata ja sulkea ohjelmiston valikosta. Tämä vapaatekstiosa ei näy muille käyttäjille ja siinä on tarkoitus viestiä käyttäjälle itselleen, esimerkiksi siitä mihin vaiheeseen ohjaustiedostomäärittely on jäänyt ja mitä pitäisi tehdä seuraavaksi tai kopioida siihen alkuperäinen tehtävä, kuten mitä tämän ohjaustiedostomäärittelyyn tulee sisältää. Käyttäjän pitää voida määrittää tämä vapaatekstiosa halutessaan myös muiden käyttäjien nähtäväksi.
- Ohjelmiston pitää kyetä tuottamaan raportit vähintäänkin PDF, Word ja Excell -muodoissa. Käyttäjän pitää voida sisällyttää raportteihin haluamansa tiedot käyttäen erilaisia suodatusominaisuuksia, joiden tulee tukea tavallisia loogisia valintoja, kuten AND, NOT jne.
- Yksittäiset heiteohjelmat pitää voida kopioida normaalin Windows -toiminnallisuuden mukaisesti. Eli heiteohjelmia pitää voida kopioida yksittäisen uhkan tai uhkaluokan sisällä sekä eri ohjaustiedostosta toiseen.
- Yksittäisestä uhkasta tai uhkaluokasta pitää voida tehdä kopio sellaisenaan sisältäen heitesekvenssit ja muut uhka tai uhkaluokkakohtaiset määrittelyt.
- Käyttäjien pitää voida arvioida yksittäisiä heitesekvenssejä sekä uhkia tai uhkaluokkia asteikolla 1-5. Tämä helpottaa samojen heiteohjelmien tai uhkien jatkokäyttöä, koska niiden hyvyyttä on arvioitu ja se on nähtävissä. Tämä hyvyysluku pitää olla nähtävissä päänäkymässä kyseisen heiteohjelman tai uhkan perässä. Tämä luku voi myös puuttua, mikäli hyvyyslukua ei ole asetettu.

- Käyttäjien tekemien heiteohjelmien tai uhkien arvostelujen perusteella, pitää voida automaattisesti etsiä paras uhka tai kullekin uhkalle paras heiteohjelma.
- Uhkia ja heitesekvenssejä ei tule versioida automaattisesti. Jokaisella uhkalla ja heitesekvenssillä tulee olla oma kommenttikenttä johon voi kirjoittaa vapaasti mitä tahansa. Näiden kommenttikenttien tulee olla erillisiä tekstitiedostoja tai vastaavia, jotka mukautuvat automaattisesti tekstimäärän kasvaessa. Näitä kommenttikenttiä ei automaattisesti näytetä käyttäjälle, vaan ne avautuvat vain kyseisen heiteohjelman, uhkan tai vastaavan yhteydessä olevasta napista painamalla, jolloin kommenttikenttä avautuu käyttäjälle erilliseen ikkunaan. Tässä avautuvassa ikkunassa käyttäjä pystyy suoraan muokkaamaan kommenttikenttää, vaikka kyseinen heiteohjelma, uhka tai vastaava olisikin lukittu/hyväksytty. Kopioitaessa uhka tai heitesekvenssi tämä kommenttikenttä ei kopioidu. Kuitenkin uuden kopion kommenttikenttään kirjoitetaan tieto mistä heiteohjelmasta, mistä uhkasta ja mistä heitesekvenssistä tai vastaavasta se on kopioitu ja milloin sekä minkä käyttäjän toimesta. Lisäksi kopioitavan tietueen kommenttikenttään lisätään automaattisesti tieto siitä, mihin kyseinen tietue on kopioitu ja milloin. Nämä mahdollistavat pitkienkin kopiointiketjujen seuraamisen aina alkuperäiseen tietueeseen saakka, mikäli näitä automaattisia tietoja ei ole käyttäjän toimesta poistettu kommenttikentistä.
- Valitusta uhkasta tai uhkaluokasta pitää voida generoida raportti, josta käy ilmi mitä heitesekvenssejä kyseiselle uhkalle on käytetty kaikissa aiemmissa ohjaustiedostoissa. Tätä raporttia pitää pystyä rajaamaan ajallisesti sekä tekijän mukaan.
- Edellinen vaatimus johtaa automaattisesti siihen, että jokaiseen ohjaustiedostoon, uhkaa tai uhkaluokkaan sekä heiteohjelmaan on liitettävä automaattisesti aikaleima. Aikaleiman pitää sisältää päiväys ja kellonaika, jotka pyydetään käytettävältä työasemalta.
- Ohjaustiedostomäärittelyitä pitää voida tallentaa eri nimillä uusien versioiden tai kokonaan uuden ohjaustiedoston pohjaksi. Lisäksi seuraavia yksittäisiä kokonaisuuksia pitää voida erikseen kopioida uuteen ohjaustiedostomäärittelyyn jo aiemmin tuotetuista ohjaustiedostomäärittelyistä.

- Yksittäiset heiteohjelmat
- Yksittäiset uhkat tai uhkaluokat
- Kutakin uhkaa tai uhkaluokkaa vastaavat kaikki heiteohjelmat pitää saada näkymään kerrallaan yhdessä näkymässä.
- Heiteohjelmien ja niiden sisältämien uhkien tai uhkaluokkien sekä niiden sisältämien erillisten parametrien pitää muodostaa hakemistopuu, joka alkaa korkeimmalla tasolla yksittäisistä ohjaustiedostoista ja päättyy yksittäisten uhkien tai uhkaluokkien yksittäisiin heiteohjelmiin.
- Shift ja Cntrl –näppäimiä käyttämällä pitää voida valita useampia heiteohjelmia samanaikaisesti. Tällöin voidaan valita muokata ja kopioida useampia heiteohjelmia uhkan sisällä samalla kertaa. Ensin valitaan, mikä heiteohjelma halutaan kopioida ja sitten minne se halutaan sijoittaa, kuten normaalisti Windowsissa.
- Useampien eri heiteohjelmien kopiointi kerrallaan aiheuttaisi epäselvyyttä, eikä ole siten oikein järkevää vaan parempi toimintatapa olisi pystyä valitsemaan ainoastaan yksi kopioitava heiteohjelma kerrallaan.
- Heiteohjelmaa muokattaessa käyttäjälle näytetään itse heitesekvenssin määrittelyn lisäksi visuaalinen representaatio kyseisestä heiteohjelmasta aikajanana sekä sen nimi ja sen käyttämiseen kuluva kokonaisheitemäärä heitetyypeittäin.
- Päänäkymässä näytetään jatkuvasti ainoastaan kyseisen heiteohjelmien ensimmäiset kaksi heitejaksoa, jotka ovat ensimmäinen silppujakso sekä ensimmäinen soihtujakso. Jos heiteohjelmaan sisältyy muita heitejaksoja, näytetään siitä selvä merkki käyttäjälle sekä aktivoidaan nappi josta käyttäjä voi avata koko heiteohjelman tarkasteltavaksi ja muokattavaksi. Napin painalluksella loput segmentit näytetään allekkaisina riveinä suoraan päänäkymään lisättyinä riveinä, jolloin päänäkymän rivimäärä vain kasvaa avautuvien segmenttien määrällä tai erillisessä ikkunassa joka avautuu päänäkymän päälle. Vaihtoehtoisesti päänäkymässä näytettävien heitejaksojen lukumäärän voi itse määritellä asetustiedostossa.
- Eri heiteohjelmien suuntaavuus voidaan määritellä alustustiedostossa, niin että siinä määritellään kaikki koneessa käytettävissä olevat makasiinit. Makasiineina voidaan käsittää myöhemmässä vaiheessa joko yksittäiset

makasiinit tai heittosuunnat. Tämä valinta ei siis rajoita varsinaista makasiinisuunnittelua vaan liittyy ainoastaan siihen, mistä makasiinista tai minkä suunnan makasiineista halutaan heittää ja tämä yhdistetään yksittäiseen heiteohjelmaan valitsemalla tietty makasiini tai makasiinit (heitesuunnan tapauksessa). Eli alustustiedostossa määritellään ja nimetään jokainen koneen maksiini erikseen ja sen jälkeen niistä voidaan muodostaa vapaasti valittavia ryhmiä, joille annetaan ryhmänimi. Varsinaisesti ohjaustiedoston heitesekvensseissä voidaan sitten valita vapaasti joko yksittäinen makasiini, makasiiniryhmä tai valita ANY, jolloin omasuojaheitin voi muodostaa heitesekvenssin vapaasti, käyttäen mitä tahansa koneen makasiineja, joissa on haluttua heitetyyppiä.

Liite 4. Koonnos omasuojaheittimien ohjelmoimiseen vaikuttavista ominaisuuksista.

Lentokoneeseen asennettavissa olevia omasuojalaitteita sekä muita laitteita, jotka voivat antaa tietoa omasuojaheittimelle tai suoraan ohjata sen toimintaa:

- Tutkavaroitin
- Ohjuslaukaisuvaroitin
- Laservaroitin
- Omasuojahäirintälähetin
- Suunnattava infrapunavastatoimilähetin
- Oman koneen avioniikalta saatavat tiedot oman koneen tilasta.
- Heiteohjelmien hyväksyntäpainikkeet tai vastaavat
- Manuaaliohjelmien laukaisupainikkeet tai vastaavat
- Omasuojaheittimen toimintatilan valintapainikkeet tai vastaavat
- Mahdolliset muut laitteet

Edellä kerrottujen laitteiden antamia tietoja, joita voidaan hyödyntää käytettävän heiteohjelman valinnassa:

- Uhkan tunnistus
- Uhkan toimintatila
- Uhkasuunta suhteessa omaan lentokoneeseen
- Uhkan korkeus
- Uhkan etäisyys
- Uhkan korkeuskulma suhteessa omaan koneeseen
- Heiteohjelman laukaisupyyntö manuaaliohjelmalle 1-n
- Heiteohjelman laukaisupyyntö ohjuslaukaisuohjelmalle 1-n
- Heiteohjelman laukaisupyyntö automaattiohjelmalle 1-n
- Omasuojaheittimen toimintatilan valinta
- Jokin muu kriteeri
- Joissakin tapauksissa muut laitteet voivat antaa myös suoria heittokäskyjä omasuojaheittimelle. Tällöin heittimelle kerrotaan

ainoastaan ajanhetket, jolloin heitteitä halutaan heitettävän, heitetyyppi ja samanaikaisten heitteiden lukumäärä.

Lentokoneen omaan lentotilaan liittyviä ominaisuuksia, joita voidaan käyttää heite-ohjelmien valintakriteereinä:

- Oman koneen korkeus
- Oman koneen nopeus
- Oman koneen kiihtyvyys
- Oman koneen nousukulma
- Oman koneen kiertonopeus
- Oman koneen kääntönopeus
- Oman koneen nousunopeus
- Oman koneen jälkipolttimen käyttö
- Oman koneen moottorien tehoasetus
- Jokin muu ominaisuus

Omasuojaheittimien ominaisuuksia, jotka vaikuttavat niiden mahdollisuuksiin tuottaa erilaisia heiteohjelmia.

- Heittimen makasiinikokoonpano
- Mahdollisuus määritellä mistä makasiinista tai makasiineista heitteitä heitetään
- Makasiinin muoto n x m heitettä.
- Samanaikaisten heitteiden maksimimäärä yhdestä tai kahdesta makasiinista tai koko järjestelmästä.
- Erilaisten makasiinikuvausten maksimimäärä
- Erilaisten uhkaluokkien maksimimäärä
- Korvaavien heiteohjelmien määrä, mikäli ensisijaista heiteohjelmaa ei pystytä heittämään
- Erilaisten heiteohjelmien maksimimäärä uhkaluokkaa kohden
- Erilaisten heiteohjelmien kokonaismaksimimäärä
- Tutkavaroitinuhkien maksimimäärä
- Ohjusvaroitinuhkien maksimimäärä

- Laservaroitinuhkien maksimimäärä
- Manuaaliohjelmien maksimimäärä
- Erilaisten heitetyyppien maksimimäärä (ECDS Enhanced Countermeasures Dispensing System 2009)
- Erilaisten heitemallien maksimimäärä (VICON 78 Series 455 Countermeasures Dispensing Systems, 2)
- Heitejaksojen lukumäärä heiteohjelmassa (VICON 78 Series 455 Countermeasures Dispensing Systems, 2)
- Erilaisten heitemallien maksimimäärä per makasiini (ALE-47 Airborne Countermeasures Dispenser System 2008, 2)
- Heitteen ikään perustuva pudotus (Advanced Countermeasures Dispenser System, 2)
- Mahdollisuus käyttää kahdesti laukaistavia heitteitä. (Advanced Countermeasures Dispenser System, 2)
- Mahdollisuus käyttää kaksinkertaisia heitteitä, eli kahden normaaliheitteen kokoisia heitteitä. (ECDS Enhanced Countermeasures Dispensing System 2009)
- Toimintatilat, manuaali, puoliautomaatti tai automaatti (VICON 78 Series 455 Countermeasures Dispensing Systems, 2)
- Pakkopudotusmahdollisuus / Jettison
- Heittimen aikaporras, eli millä tarkkuudella heittimen heittoajankohdat voidaan määritellä
- Peräkkäisten heitteiden välinen minimiaika samasta makasiinista
- Muut ominaisuudet?

Liite 5. Sisällysluettelo asiakirjasta Yleinen omasuojaheittimen ohjaustiedostojen suunnittelutyökalu Vaatimusmäärittely

Sisällys

Liite 6. Käyttäjätestin tutkimussuunnitelma

Suunnitelma yleisen heitinohjelmistojen suunnittelutyökalun käyttäjätestistä

Aleksi Borén maaliskuu 2016

Tämä suunnitelma kuvaa yleisen heitinohjelmistojen suunnittelutyökalun ohjelmistosuunnitelman mukaisen ohjelmistotuotteen käytettävyyden testaamisen järjestelyt. Käytettävyyttä testataan ohjelmistosta muodostettujen käyttöliittymäkuvien avulla.

1. Testijärjestelyt

   Koe järjestetään neuvotteluhuoneessa, Ilmavoimien esikunnassa Jyväskylässä erikseen sovittavana ajankohtana. Testissä käytetään valkoista tussitaulua, jolle käyttöliittymäkuvat heijastetaan. Sitä mukaa kun käyttäjä tekee valintoja tai kirjoittaa tekstiä, testaaja lisää kyseiset valinnat ja tekstit suoraan tussilla piirtotaululle. Mikäli käyttäjän tekemä valinta aiheuttaisi ohjelmassa uuden ikkunan avaamisen tai käyttäjä avaa alasvetovalikon tms. vaihdetaan käyttäjän nähtäväksi kyseinen uusi ikkuna tai alasvetovalikko jatkotoimenpiteiden suorittamiseksi.

   Testaaja tarkkailee käyttäjän toimintaa ja kirjaa käsin paperille tekemänsä havainnot. Erityisesti havainnoitavia asioita ovat seuraavat:

   a. Kohdat jotka estävät ohjelmiston käyttämisen jatkamisen.
   b. Tilanteet joissa käyttäjä ei pysty suorittamaan haluamiaan toimintoja.
   c. Tilanteet joissa käyttäjä on epävarma siitä, mitä hänen pitäisi seuraavaksi tehdä.
   d. Ohjelmiston ominaisuudet, jotka eivät tue käyttäjän tehokasta työskentelyä huonon ohjelmistoergonomian tai muun syyn vuoksi.
   e. Ohjelmiston tai käyttäjän aiheuttamat virhetilanteet.

2. Testihenkilöt

   Tutkimukseen pyritään saamaa osallistujiksi omasuojaheittimien ohjelmointiin syvällisesti perehtyneitä henkilöitä sekä omasuojaheittimien ja heitteiden toimintaperiaatteita tuntevia henkilöitä. Tutkimukseen pyritään saamaan mukaan viisi varsinaista osallistujaa sekä lisäksi pilottitestaaja, jolla ei ole var-

sinaista aiempaa kokemusta omasuojaheittimen ohjelmoinnista. Tällöin käyttöliittymän intuitiivisuus sekä itse koejärjestelyn toimivuus joutuu erityisen kovalle koetukselle jo pilottitestivaiheessa ja testiä voidaan tarvittaessa vielä säätää ennen varsinaista koekäyttäjätestien aloittamista. Koehenkilöiden määrä ja valinta perustuu lähinnä käytettävissä olevien, soveltuvien henkilöiden vähäiseen määrään sekä 5-6 testihenkilön optimaaliseen määrään.

3. Testin kulku

Testivaihe 1.

Ensimmäisessä testivaiheessa käyttäjälle näytetään kuva alustusohjelman käyttöliittymästä ja häntä pyydetään luomaan kokonainen alustustiedosto alusta loppuun asti ja tallentamaan se.

Testivaihe 2.

Toisessa testivaiheessa käyttäjää pyydetään luomaan uusi ohjaustiedosto alusta alkaen siihen vaiheeseen, jossa seuraavat asiat on toteutettu.

- Ohjaustiedosto on luotu ja sille on annettu nimi.
- Ohjaustiedostoon on luotu ainakin yksi uhka.
- Uhkalle on määritetty vähintään yksi heittokriteeri.

Testivaihe 3.

Kolmannessa testivaiheessa jatketaan vaiheen 2 ohjaustiedostoa siihen asti kunnes seuraavat asiat on toteutettu.

- Ohjaustiedostoon on luotu vähintään yksi heiteohjelma, jossa on vähintään kaksi erilaista heitejaksoa.
- Ainakin kahteen erilaiseen heittokriteeriin on liitetty kriteerejä vastaava heiteohjelma.
- Ohjaustiedostoon on luotu vähintään yksi makasiinimäärittely.

Testivaihe 4.

Neljännessä testivaiheessa muokataan vaiheen 3 ohjaustiedostoa seuraavasti.

- Vaihda vaiheessa 3. heittokriteereihin asetettuja heiteohjelmia.
- Lukitse heittokriteerit.
- Poista nollaohjelmat näkymästä.
- Salli automaattinen toimintatila.
- Lisää tekstiä kommenttitiedostoon.
- Arvioi ainakin yksi heiteohjelma asteikolla 1-5.
- Lukitse ohjaustiedostoversio.
- Tallenna ohjaustiedosto.

Testivaihe 5.

Viidennessä testivaiheessa avataan aiemmin vaiheessa 4 tallennettu ohjaustiedosto ja tallennetaan siitä kopio uudella nimellä sekä siirrytään luomaan siitä raportteja. Käyttäjä luo ja tallentaa Excel –tiedostoon raportin, joka sisältää ainoastaan seuraavat asiat.

- Ohjaustiedoston sisältämät uhkat.
- Ohjaustiedoston uhkat, joiden arvosana on vähintään 3.

Testivaihe 6.

Kuudennessa testivaiheessa tuotetaan aiemmin luodulle ohjaustiedostomäärittelylle testipisteet. Käyttäjä luo ainakin seuraavat testipisteet ja tallentaa testipistemäärittelyn Excel –tiedostoksi.

- Uhkasuunta asetettujen kriteerialueiden keskeltä.
- Oma korkeus asetettujen kriteerialueiden ylä- ja alapuolilta, sekä maksimin yläpuolelta.
- Kaikille ohjaustiedoston uhkille sekä manuaaliohjelmille.

4. Taustatarina

Olet saanut tehtäväksesi tuottaa ensimmäisen kerran uuden ohjaustiedostosuunnitelman uudella juuri tähän tarkoitukseen hankitulla ohjelmistolla. Tar-

koitus on opetella ohjelmiston käyttöä ja luoda ensimmäiset mallitiedostot, joiden pohjalta voit opastaa muille käyttäjille kyseisen ohjelmiston käyttöä. Tarkoitus on kokeilla ohjelmiston ominaisuuksia mahdollisimman kattavasti, jotta saavutat ymmärryksen ohjelmiston eri toiminnoista.

5. Testivaiheiden jälkeen esitettävät kysymykset

    1. Kuinka helpolta juuri käyttämässäsi ohjelmiston osa sinusta vaikutti?
    2. Jos vertaat ohjelmistoa muihin käyttämiisi ohjelmiin, oliko ohjelmisto mielestäsi visuaalisesti huonompi, samalla tasolla tai parempi kuin muut vastaavat ohjelmat?
    3. Mikä olisi tärkein muutettava asia juuri käyttämässäsi ohjelmiston osassa?
    4. Mitä ominaisuuksia tai valintoja juuri käyttämästäsi ohjelmiston osasta mielestäsi puuttuu?
    5. Jos tarvitsisit ohjelmistoa työssäsi, oliko siinä asioita jotka pitäisi ehdottomasti muuttaa, ennen kuin voisit ottaa sen käyttöön?
    6. Mikä oli parasta juuri käyttämässäsi ohjelmiston osassa?
    7. Poikkesiko juuri käyttämäsi ohjelmiston osa logiikaltaan tai käyttöliittymältään aiemmin kokeilemistasi tämän saman ohjelmiston osista?
    8. Oliko ohjelmiston antama palaute hyödyllistä ja riittävää?

Liite 7. Käyttäjätestin diaesitys

Alla esitetystä käyttäjätestin diaesityksestä on luottamuksellisuussyistä poistettu muutamia dioja. Alkuperäinen testi käsitti kaikkiaan 24 diaa.

Yleisen heitinohjelmistojen suunnittelutyökalun testaus

Aleksi Borén maaliskuu 2016

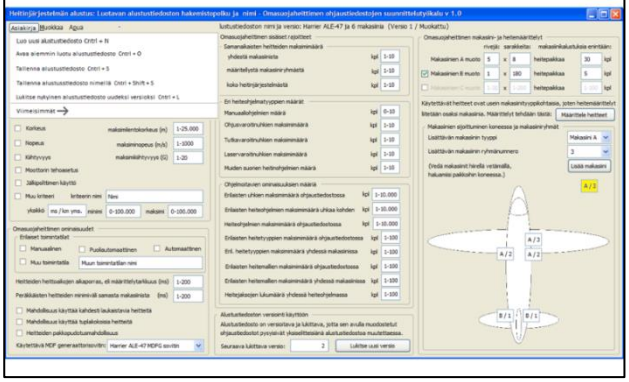
Tausta

- Olet saanut tehtäväksesi tuottaa ensimmäisen kerran uuden ohjaustiedostosuunnitelman, uudella juuri tähän tarkoitukseen hankitulla ohjelmistolla. Tarkoitus on opetella ohjelmiston käyttöä ja luoda ensimmäiset mallitiedostot, joiden pohjalta voit opastaa muille käyttäjille kyseisen ohjelmiston käyttöä. Tarkoitus on kokeilla ohjelmiston ominaisuuksia mahdollisimman kattavasti, jotta saavutat ymmärryksen ohjelmiston eri toiminnoista.

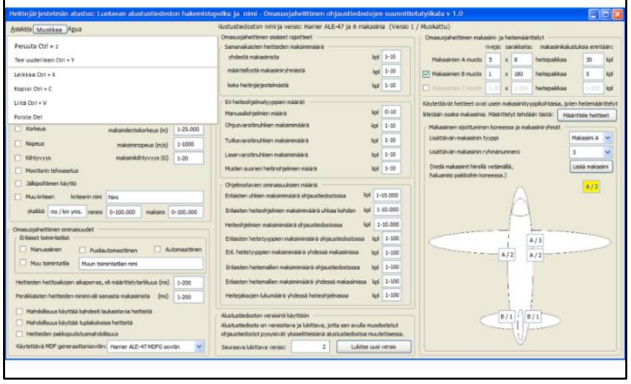
Tehtävä 1

- Taustaa:
  - Ohjelmisto jakautuu kolmeen pääosaan, joista ensimmäistä käytetään periaatteessa ainoastaan kerran yhtä heittimen ja lavetin muodostamaa kokonaisuutta kohti.
  - Ensimmäisessä osassa muodostetaan ns. alustustiedosto, jonka avulla määritellään heittimen ja lavetin muodostaman kokonaisuuden ominaisuudet ja rajoitukset sekä käytettävissä olevat heitteet.
  - Alustustiedostoon tallennetut määritykset rajaavat pääohjelmassa tarjottavat ominaisuudet vain sellaisiin, joita lavetti ja heitinjärjestelmä oikeasti tukevat.
- Tehtävä:
  - Luo uusi ohjaustiedostojen alustustiedosto täytä se haluamallasi tavalla.
  - Tallenna ja versioi tekemäsi alustustiedosto.

Alustustiedoston suunnittelu: Asiakirja



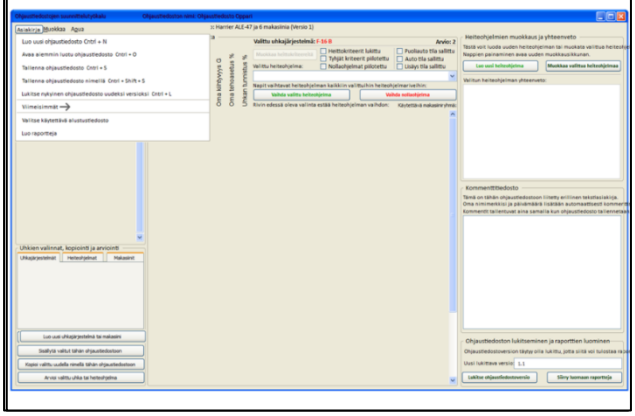
Alustustiedoston suunnittelu: Muokkaa/Apua



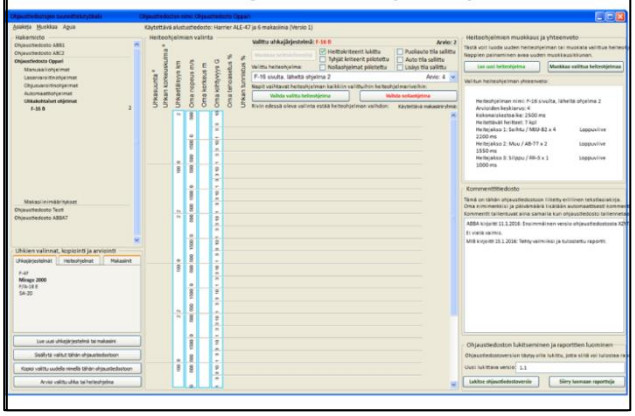
Tehtävä 2

- Taustaa:
  - Ohjelmiston toinen osa on ns. pääohjelma, joka käyttää aiemmin muodostettua alustustiedostoa määrittelemään näytettäviä vaihtoehtoja sekä varmistamaan annettujen syötteiden pysyminen annetuissa raja-arvoissa.
  - Pääohjelma on jatkuvasti taustalla käynnissä, vaikka sen päälle avataan uusia ikkunoita tarvittavien toimenpiteiden toteuttamiseksi.
- Tehtävä:
  - Avaa ohjaustiedostojen suunnittelutyökalu ja luo ja nimeä uusi ohjaustiedosto. Valitse samalla aiemmin luomasi alustustiedosto käyttöön.
  - Lisää ohjaustiedostoon ainakin yksi uhka.
  - Määritä luomallesi uhkalle vähintään yksi heittokriteeri.

Pääohjelma: Tyhjä



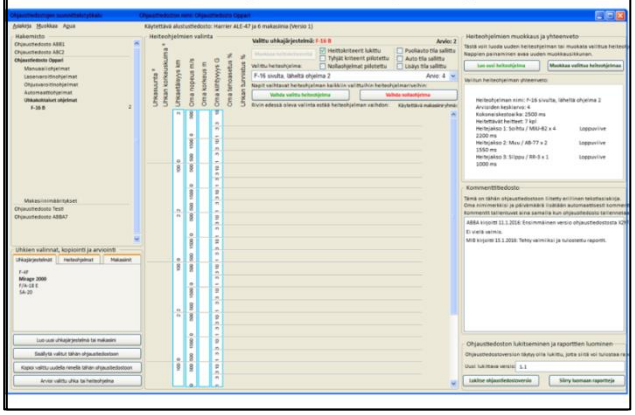
Pääohjelma: Täytetty



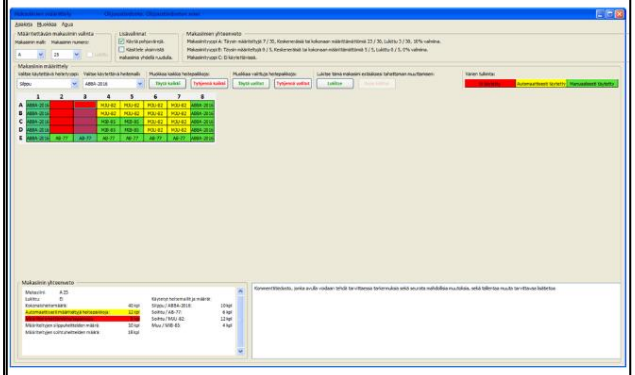
Tehtävä 3

- Taustaa:
  - Pääohjelmassa tehtävässä makasiinimäärittelyssä, alustusohjelmassa määritellyille makasiinimalleille luodaan makasiinimäärittelyt, joihin sisältyvät kunkin makasiinin kuhunkin heitepaikkaan asetettavat heitteet.
- Tehtävä:
  - Luo aiempaan ohjaustiedostoon vähintään yksi uusi heiteohjelma, jossa on vähintään kaksi erilaista heitejaksoa.
  - Liitä ainakin kahteen erilaiseen heittokriteeriin heiteohjelma.
  - Luo ohjaustiedostoon vähintään yksi makasiinimäärittely.

Pääohjelma: Täytetty



Makasiinien määrittely:



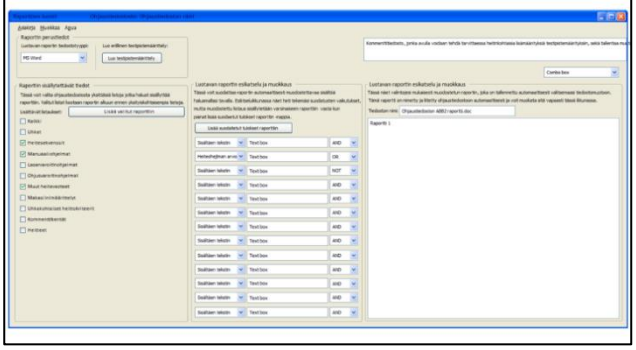
Tehtävä 4

- Tehtävä:
  - Vaihda vaiheessa 3. heittokriteereihin asetettuja heiteohjelmia, käyttäen myös nollaohjelmia
  - Lukitse heittokriteerit.
  - Poista nollaohjelmat näkymästä.
  - Salli automaattinen toimintatila.
  - Lisää tekstiä kommenttitiedostoon.
  - Arvioi ainakin yksi heiteohjelma asteikolla 1-5.
  - Lukitse ohjaustiedostoversio.
  - Tallenna ohjaustiedosto ja sulje ohjelma.

Tehtävä 5

- Taustaa:
  - Ohjelmiston tärkein tehtävä on tuottaa erilaisia raportteja varsinaisen ohjaustiedoston ohjelmoimiseksi sekä tuottaa automaattisesti testipisteitä ohjelmoidun ohjaustiedoston testaamista varten.
- Tehtävä:
  - Avaa aiemmin luotu ohjaustiedosto ja tee siitä kopio uudella nimellä.
  - Avaa Raporttien luomisikkuna.
  - Luo raportti joka sisältää ainoastaan seuraavat tiedot:
    - Ohjaustiedoston sisältämät uhkat.
    - Ohjaustiedoston uhkat, joiden arvosana on vähintään 3.
  - Tallenna muodostettu raportti Excell –tiedostoksi.

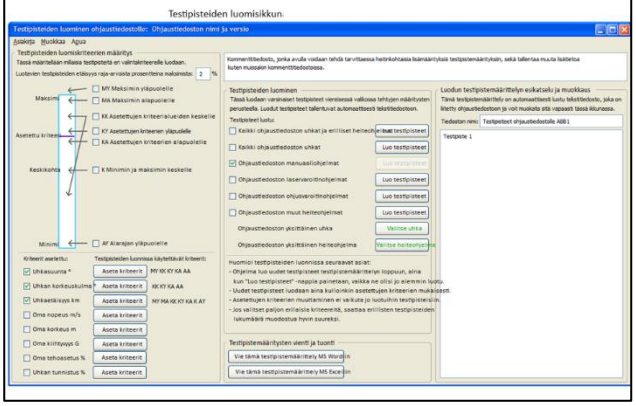
Raporttien luominen:



Tehtävä 6

- Tehtävä:
  - Luo aiemmin määritellylle ohjaustiedostolle automaattisesti ainakin seuraavat testipisteet:
    - Uhkasuunta asetettujen kriteerialueiden keskeltä.
    - Oma korkeus asetettujen kriteerialueiden ylä ja alapuolilta, sekä maksimin yläpuolelta.
    - Kaikille ohjaustiedoston uhkille sekä manuaaliohjelmille.
  - Vie testipistemäärittelyt Exceliin.

Testipisteiden luominen:



Kiitos osallistumisesta testiin!

Vastaa vielä seuraaviin kysymyksiin.

1. Kuinka helpolta juuri käyttämässäsi ohjelmiston osa sinusta vaikutti?
2. Jos vertaat ohjelmistoa muihin käyttämiisi ohjelmiin, oliko ohjelmisto mielestäsi visuaalisesti huonompi, samalla tasolla tai parempi kuin muut vastaavat ohjelmat?
3. Mikä olisi tärkein muutettava asia juuri käyttämässäsi ohjelmiston osassa?
4. Mitä ominaisuuksia tai valintoja juuri käyttämästäsi ohjelmiston osasta mielestäsi puuttuu?
5. Jos tarvitsisit ohjelmistoa työssäsi, oliko siinä asioita jotka pitäisi ehdottomasti muuttaa, ennen kuin voisit ottaa sen käyttöön?
6. Mikä oli parasta juuri käyttämässäsi ohjelmiston osassa?
7. Poikkesiko juuri käyttämäsi ohjelmiston osa logiikaltaan tai käyttöliittymältään aiemmin kokeilemistasi tämän saman ohjelmiston osista?
8. Oliko ohjelmiston antama palaute hyödyllistä ja riittävää?

Liite 8. Käyttäjätestin tulokset

Testi suoritettiin 18.3.2016 Ilmavoimien esikunnassa Tikkakoskella. Testitilana toimi kahvihuone, jossa videotykillä heijastettiin kalvoesitys piirtotaululle, niin että käyttöliittymäkuviin pystyi tussilla syöttämään haluamansa tiedot.

Tehtävä 1 aloitettiin klo 14.25.

Käyttäjä toivoi mahdollisuutta asettaa enemmän lisäkriteereitä kohdassa ”Oman koneen tiedot ja niiden tarkkuus”.

Peräkkäisten heitteiden minimivälin valinta on nyt 1-200 ms, mutta 0-200 ms mahdollistaisi täysin samanaikaisten heitteiden heittämisen. Muutettava tämä valinta vastaamaan todellista tarvetta ollen jatkossa 0-200 ms.

Käyttäjä toivoi että heitemäärä makasiiniryhmästä valinta liitettäisiin osaksi makasiinimäärittelyitä, koska sen merkitys jäi tässä kohtaa epäselväksi.

Samanaikaisten heitteiden maksimimäärä koko heittimestä on nyt enintään 10, mutta se pitäisi voida asettaa myös suuremmaksi.

Kohdassa ”Eri heiteohjelmatyyppien määrät” pitää kaikkien valintojen otsikoissa lukea ohjelmien eikä uhkien, kuten nyt osassa lukee.

Edelleen kohdassa ”Eri heiteohjelmatyyppien määrät” pitää mahdollistaa useampien muiden suorien heiteohjelmien lisääminen tarvittaessa.

Edelleen kohdassa ”Eri heiteohjelmatyyppien määrät” pitää tutkavaroitinuhkien kokonaismäärää nostaa vastaamaan paremmin todellisia heitinjärjestelmiä. Asetetaan uudeksi maksimimääräksi 1000.

Erilaiset heitteiden kokoon ja muihin ominaisuuksiin liittyvät valinnat pitää sisällyttää makasiinikohtaisiin valintoihin eikä kohtaan ”Omasuojaheittimen ominaisuudet”, koska mahdollisuudet ovat usein myös makasiinikohtaisia.

Heitteiden määrittelyikkunassa käytetty nimeämistapa heitetyyppi ja heitemalli tai heitenimi ei ole käyttäjälle ymmärrettävä, joten se pitää muuttaa. Käyttäjä ehdotti käytettäväksi nimityksiä heitekategoria ja heitemalli.

Heitteiden määrittelyikkunassa käytetty heitteiden poistamislogiikka ei selvinnyt käyttäjälle, eikä edes käyttäjän avustaminen helpottanut tilannetta. Poistamiseen on käytettävä muista ikkunoista tuttua valintaruutua radio-napin sijaan ja otsikoitava valinnan tarkoitus tai mietittävä jokin muu toimintatapa heitemallien poistamiseen.

Tehtävä 1 päätettiin klo 15.05, jolloin siirryttiin suoraan tehtävään 2.

Versionumerointi ei vastaa tarkoitusta, eli 1.1 pitää muuttaa muotoon 1. Eli versiot eivät muotoa ole 1.1, 1.2 jne. vaan 1, 2, 3 jne. koska käyttäjä ei voi itse päättää seuraavaa versionumeroa.

Käyttäjä ei ymmärtänyt Asiakirja –valikon nimeämistä ja se esti käyttäjää luomasta uuden ohjaustiedoston ilman avustamista. Valikon nimeämistä uudelleen muotoon Aloitus tai Tiedosto on harkittava.

Erillinen Poistu –valinta saattaisi olla hyvä ikkunoissa, joista palataan pääkäyttönäkymään, erityisesti silloin kun käyttäjällä ei ole erillistä tallenna -nappia.

Tehtävä 2 päättyi klo 15.25, jonka jälkeen pidettiin 5 minuutin tauko ja siirryttiin tehtävään 3 klo 15.30.

Käyttäjä toivoi Heiteohjelmien suunnitteluikkunassa olevaan visualisointiin liitettäväksi tieto kunkin heitteen kategoriasta ja mallista esimerkiksi kunkin heitteen kohdalle. Lisäksi eri kategorioiden heitteiden haluttiin näyttävän erilaiselta.

Palattaessa heiteohjelmien suunnitteluikkunasta takasin, ei päädytty oikeaan ikkunaan. Tämä on ainoastaan virhe testiä varten muodostetussa kalvoesityksessä, eikä itse ohjelmistossa.

Tehtävä 3 päättyi klo 15.55, jonka jälkeen jatkettiin suoraan tehtävään 4.

Heiteohjelmien lukitus jäi käyttäjälle epäselväksi, mutta lyhyen selvityksen jälkeen hän piti sitä käyttökelpoisena ominaisuutena.

Ohjaustiedostojen versioinnin ja lukituksen vaikutus on epäselvää ja sen toimintalogiikkaa pitää tarkentaa.

Tehtävä 4 päättyi klo 16.07, jonka jälkeen pidettiin lyhyt tauko. Tehtävä 5 alkoi klo 16.10.

Raporttien luomisikkunassa pitää nappi ”Lisää suodatetut tulokset raporttiin” muuttaa muotoon ”Hyväksy…”, koska nykyisellään toimintalogiikka jäi käyttäjälle epäselväksi. Myös selkeämmän logiikan muodostamista toivottiin.

Käyttäjä ei ymmärtänyt, että kaikki valitut suodattimet ovat automaattisesti samanaikaisesti käytössä, joten asiasta pitää laittaa huomautus ohjeistukseen tai lisätä aktivointivalinta jokaiseen eri suodattimeen.

Tehtävään 5 liittyvää tallennusta ei voitu suorittaa, koska ohjelma tallentaa raportit automaattisesti, ilman erillistä tallentamista. Tämä on ominaisuus jota käyttäjän on vaikea havaita tai käsittää ja saattaa siksi aiheuttaa epäluottamusta ohjelmistoa kohtaan.

Tehtävän 5 päättymisajankohtaa ei merkitty muistiin ja sen jälkeen jatkettiin suoraan tehtävään 6.

Käyttäjä ei ymmärtänyt kriteerien asettelun toimintalogiikkaa. ”Kriteeri asetettu pitäisi poistaa käyttäjän käytettävistä, niin että ne ovat ainoastaan tuplavarmistus sille että kyseiselle heittokriteerille on jo asetettu testipisteiden luontikriteerit.

Käyttäjä toivoi että yhden % -osuuden valitsemisen sijaan jokaiselle heittokriteerille voitaisiin valita käytettävä raja-arvojen läheisyysarvo erikseen.

Testipisteiden tallennus pitäisi tapahtua samalla tavalla kuin Raporttien luontiikkunassakin. Varsinainen logiikka on vielä suunniteltava erikseen uudelleen.

Käyttäjän toiveen mukaan testipistemääritysten tekeminen Raportit ikkunan kautta on poistettava.

Varsinainen testi päättyi noin klo 16.45, jonka jälkeen käyttäjälle esitettiin vielä aiemmin valmistellut kysymykset, joiden vastaukset ovat alla.

1. Keskivertoa helpompi.
2. Yhdellä ruudulla on paljon asiaa, joten hahmottaminen vie paljon aikaa ja hankaloitti käyttöä ensimmäisellä käyttökerralla. Saattaisi jatkossa myös helpottaa käyttöä, kun ikkunoita on vähemmän. Testipisteet pois raporttien alta voisi olla hyvä.
3. Asioiden jakaminen eri sivuille voisi helpottaa käyttöä, kun sivut olisivat vähemmän täynnä asiaa ja valintoja.
4. Ei mitään.
5. Ei.
6. Visualisoinnit heitteistä heitesekvenssejä suunniteltaessa, sekä heiteohjelman visualisointi.
7. Ei.
8. Palautetta ei saatu eikä sitä kaivattu, joten vastaaminen on mahdotonta.

Testi lopetettiin klo 16.58, jolloin kaikki vastaukset esitettyihin kysymyksiin oli saatu.