VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Mikko Palomaa

SOVELLUKSEN TESTAUS

Merlot Palotarkastus

Tekniikka ja liikenne
2011

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Mikko Palomaa
Opinnäytetyön nimi	Sovelluksen testaus
Vuosi	2011
Kieli	suomi
Sivumäärä	29 + 2 liitettä
Ohjaaja	Ghodrat Moghadampour

Pelastuslain uudistaminen, on johtanut siihen, että palolaitosten käyttämää Merlot Palotarkastus -sovellusta on täytynyt uudistaa vastaamaan lakimuutoksia. Tutkimuksen aiheena on sovelluksen uudistusten testaaminen. Tutkimuksessa käydään läpi testauksen eri vaiheita teoriassa ja käytännössä.

Tutkimuksen teoreettinen viitekehys sisältää testaussuunnitelman, testitapausten sekä virheraportin laatimisen. Opinnäytetyö käy läpi, mitä kannattaa ottaa huomioon, kun näitä dokumentteja luodaan, sekä mitä varten dokumentit ovat luotu.

Testauksessa ei löytynyt paljon virheitä. Pieni virhemäärä johtuu ohjelman iästä sekä siitä, ettei sovelluksesta ole olemassa monta eri versiota.

Avainsanat	Testaksen suunnittelu, testaussuunnitelma, testitapaus, virherapotti

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Mikko Palomaa
Title	Testing Software
Year	2011
Language	Finnish
Pages	29 + 2 Appendices
Name of Supervisor	Ghodrat Moghadampour

The laws handling safety are being renewed. This means that the fire inspection software “Merlot Palotarkastus”, used by the fire brigades, has to be updated to comply with the new laws. The purpose of the thesis is to test the updated software. Different steps of the testing path are gone through in this thesis, both in theory and in practice.

Creation of the testplan, testcases and error report are being handled in theory. The thesis goes through what one needs to keep in mind creating these documents. What these documents are used, for is also explained.

There were not many faults found in the testing of the new features and modifications. Most of the small, errors occurred because the definition of the renewal was not accepted by the clients in time. The small amount of errors, is because of the age of the software and because there are not many different versions of the software.

Keywords Planning of tests, testplan, testcase, error report

SISÄLLYS

Avainsanat Testaksen suunnittelu, testaussuunnitelma, testitapaus, virheraportti

KUVIO- JA TAULUKKOLUETTELO

LIITELUETTELO

1 JOHDANTO

Vuonna 2003 viimeksi uusittua pelastuslakia ollaan uusimassa jälleen. Lain uudistaminen vaikuttaa palolaitosten käyttämään palotarkastussovellukseen Merlot Palotarkastus. Logica Suomi Oy on Merlot Palotarkastuksen toimittaja.

Opinnäytetyön aiheena on Merlot Palotarkastuksen, palolaista johtuvan, päivityksen testaaminen. Päivityksen testaaminen on tärkeää, koska testaamisella Logica haluaa varmistaa, että sekä ohjelman vanhat toiminnot edelleen toimivat, että uudet toiminnot toimivat oikein.

Logica on it-alan yritys, joka on perustettu vuonna 1969 Iso-Britanniassa. Yrityksellä on n. 39 000 työntekijää Euroopassa sekä muutamassa Aasian maassa. Suomessa Logicalla on n. 3200 työntekijää 18:ssa eri kaupungin toimipisteissä. Logica on kasvanut Suomessa mm. WM-Datan ostoksen myötä ja tekee paljon yhteistyötä isojen yritysten sekä valtion kanssa.

Opinnäytetyö käsittelee sovelluksen testaamista, sekä testaamisen soveltamista käytännössä Merlot Palotarkastus -sovellukseen. Työ alkaa testaussuunnitelmalla, jonka jälkeen siirrytään testitapauksiin ja lopuksi virheraporttiin. Dokumentit toimivat sekä testauksen tuotoksina, että esimerkkeinä sovelluksen tulevaa testausta varten. Tämä johtuu siitä, että palotarkastussovellusta ei olla aiemmin testattu testitapauksilla.

2 OHJELMISTOTESTAUS

2.1 Yleistä testauksesta

Ohjelmistotestauksella pyritään löytämään ohjelman virheitä, sekä varmistamaan ohjelman laadun. Testauksen eri vaiheita ovat testauksen suunnittelu, testiympäristön luonti, testien suoritus sekä tulosten tarkastelu. Testaukseen, sekä siinä löytyvien virheiden jäljittäminen sekä korjaus vievät tyypillisesti puolet ohjelmointiprojektin resursseista, joten testaukseen kannattaa kiinnittää huomiota. (Haikala & Märijärvi, 283)

Virheetöntä ohjelmistoa ei koskaan tule olemaan. Ohjelmissa on virheitä, koska ne ovat usein niin monimutkaisia ja laajoja, että ohjelma joudutaan hajauttamaan useammalle ohjelmoijalle sekä projektiin osallistuville muille henkilöille. Tämän lisäksi on kommunikaatio-ongelmia sekä vaihtuvia vaatimuksia. (Sorvo 2010)

Ohjelmistotestauksen yhteydessä testaus määritellään perinteisesti suunnitelmalliseksi virheiden etsimiseksi. Tässä määritelmässä testaus suoritetaan usein umpimähkäisillä syötteillä ja tavoitteena on usein ohjelman toimivuuden osoittaminen, eikä virheiden löytäminen. Testaamista ei kuitenkaan kannata tehdä umpimähkään. Tarkoin suunniteltu muutaman tunnin testaus voi johtaa parempaan tulokseen kuin monen päivän umpimähkäinen kokeilu. (Haikala & Märijärvi 2004, 284)

Nykyään testaus määritellään siten, että sen katsotaan käsittävän kaikki menetelmät, joilla pyritään mittaamaan ja parantamaan ohjelman laatua. Tässä määritelmässä korostuu testauksen tuottamat mittarit kuten vielä korjaamattomien virheiden määrä. Myös esimerkiksi tarkastuksia ja ohjelmakoodin staattista analysointia pidetään testauksena tämän uudemman määrittelyn mukaan. (Haikala & Märijärvi 2004, 284-285)

Yksinkertaisessa testitilanteessa ohjelmaan annetaan syöte X, jonka jälkeen ohjelma antaa tulosteen Y. Syöte voi esimerkiksi olla hissin napin painallus ja tuloste että hissi saapuu kerrokseen. Testauksen tulos riippuu paitsi syötteestä myös

ohjelman sisäisestä tilasta. Sisäinen tila voi tarkoittaa esimerkiksi ohjelman muuttujien arvoja tai levylle tallennettuja tietoja. Testin oikeellisuus edellyttää että tuloste Y on oikein sekä että sisäinen tila on muuttunut oikein. (Haikala & Märijärvi 2004, 285)

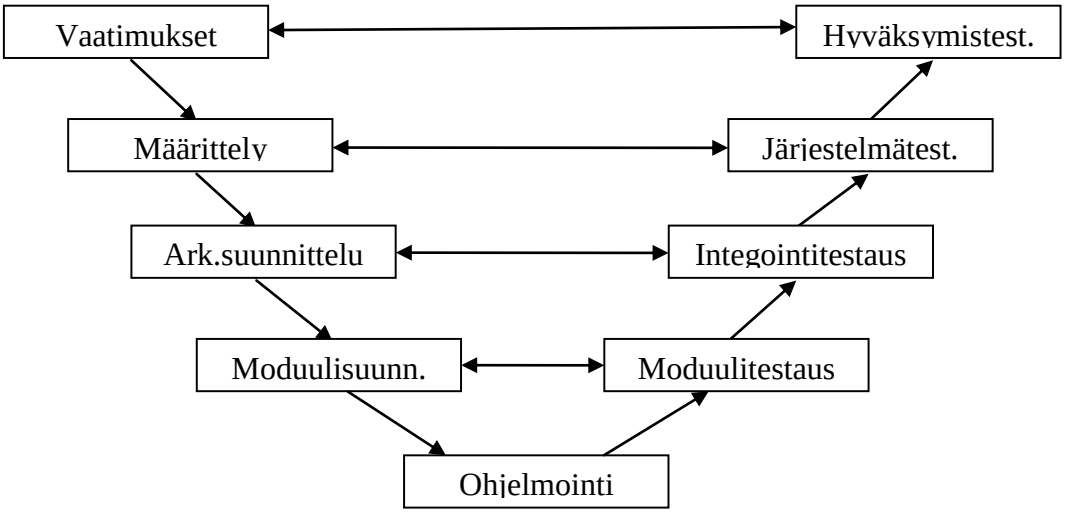
100 % kattavaa testausta kaikilla mahdollisilla syötteillä on yleensä käytännössä mahdotonta suorittaa. Yksinkertaisessa summausohjelmassakin on miljardeja syötekombinaatioita, joita kaikkia ei ehdi testata. Testauksen avulla voidaan siis osoittaa, että ohjelmassa on virheitä, mutta ohjelman virheettömyyttä ei voi taata edes yksinkertaisissa tapauksissa. Testausta kannattaa kuitenkin tehdä, ei vain kannata liiaksi luottaa ohjelman toimivuuteen, hyvistä testaustuloksista huolimatta. (Haikala & Märijärvi 2004, 285-286)

Yleensä arvioidaan valmiissa ohjelmassa olevan yksi virhe muutamaa kymmentä ohjelmariviä kohden. Pidempään käytössä olleet ohjelmat saattavat sisältää yhden virheen tuhatta riviä kohden. Arvioiden mukaan n. 5 % ohjelmavirheistä ei koskaan löydy. Tämä johtuu muun muassa siitä, että syöteavaruus on iso. Ohjelmassa voi myös olla virheitä, jotka korjaantuvat ohjelman toisen osan toiminnan tai virheen myötä. (Haikala & Märijärvi 2004, 287)

2.2 Testaustasot

Eri testaustasot V-mallin (Kuvio 1) mukaan ovat moduulitestaus, integrointitestaus, järjestelmätestaus sekä hyväksymistestaus. Hyväksymistestausta ei aina suoriteta. Se voidaan myös suorittaa järjestelmätestauksen kanssa. (Haikala & Märijärvi 2004, 288)

Vasen puoli V-mallista osoittaa ohjelmistoprojektin normaalia etenemistapaa, kun taas oikea puoli kuviosta osoittaa projektin testauksen eri tasot ja mitä projektin osaa ne testaavat. Jos testauksen suunnittelu halutaan tehdä tehokkaasti, kannattaa jokaisen tason testaus suunnitella samalla, kun kyseistä tasoa tehdään projektissa. Esimerkiksi järjestelmätestaus kannattaa suunnitella samalla, kun tehdään projektin määrittely. Tällöin määrittely on vielä tekijöiden tuoreessa muistissa ja järjestelmätestauksen suunnittelu on huomattavasti nopeampaa ja helpompaa.



Kuvio 1. Testauksen V-malli (Haikala & Märijärvi, 289).

Moduulitestauksessa testataan yksittäisiä moduuleja, joita suunniteltiin moduulisuunnitteluvaiheessa. Moduuli on yleensä n. 100–1000 ohjelmariviä. Moduulitestauksessa tarvitaan yleensä testipetejä, jotka voivat simuloida ohjelman muita osia, jos niitä ei vielä ole olemassa. Testiajureilla voidaan kutsua moduulin toteuttamia palveluita sekä tarkastella niiden tuloksia. (Haikala & Märijärvi 2004, 289)

Integrointitestauksessa testataan useamman moduulin tai moduuliryhmän välisiä rajapintoja. Integrointitestaus etenee usein rinnan moduulitestauksen kanssa. Integraatiotestaus etenee yleensä kokoavasti ylöspäin alimman tason moduuleista. (Haikala & Märijärvi 2004, 290)

Järjestelmätestauksessa testataan koko järjestelmää. Järjestelmätestaukseen voi myös liittää hyväksymistestauksen ja kenttätestauksen. Järjestelmätestauksessa testataan järjestelmän lisäksi ei-toiminnalliset ominaisuudet: kuormitustestit, luotettavuustestit, asennustestit, käytettävyystestit jne. (Haikala & Märijärvi 2004, 290)

Hyväksymistestaus suoritetaan käyttöympäristössä. Ohjelman asiakkaan edustajat toimivat yleensä testaajina, varmistaen että ohjelma täyttää asiakkaan vaatimukset. Hyväksymistestaus tehdään, kun kaikki muut testivaiheet ovat valmiita. (Partanen 2009)

Mitä aikaisemmin virhe löydetään, sitä halvemmaksi virheen korjaaminen tulee. Tämä johtuu siitä, että mitä myöhemmässä vaiheessa virhe löytyy, sitä enemmän tulee regressiotestaukseen. Uudelleen testaamalla varmistetaan ettei virheen korjaaminen rikkonut mitään, jonka todettiin toimivan aiemmalla testikierroksella

2.3 Testauksen lähestymistavat

Mustalaatikko- sekä lasilaatikkotestaus ovat kaksi peruslähestymistapaa testitapausten valintaan. Mustalaatikkotestauksessa ei välitetä ohjelman toteutuksesta, vaan valitaan testitapaukset ohjelman spesifikaatioiden perusteella. Lasilaatikkotestauksessa taas käytetään hyväksi tietoa ohjelman toteutuksesta. Kun siirrytään testauksen V-mallissa alhaalta ylöspäin, muuttuu testauksen luonne lasilaatikkotestauksesta yhä enemmän mustalaatikkotestaukseksi. (Haikala & Märijärvi 2004, 291)

Mitä matalammalla tasolla ollaan V-mallissa, sitä enemmän käytetään lasilaatikkomenetelmää. Kun siirrytään ylöspäin, siirtyy menetelmä enemmän mustalaatikkomenetelmän puolelle. Musta- ja lasilaatikkotestauksen yhdistelmää kutsutaan harmaalaatikkotestaukseksi. (Saarinen 2008).

Riittävän testauksen määrän arviointi on erittäin vaikeaa. Järjestelmätestausta, varsinkin, voi jatkaa kunnes resurssit loppuvat kesken. Usein joudutaan tekemään kompromissi olevien vikojen aiheuttamien kustannusten sekä tuotteen myöhästymisen aiheuttamien menetettyjen tuottojen välillä. Testauksen lopettamiselle tulisi asettaa hyväksymiskriteerit, jotka määritellään testaussuunnitelmassa. Järjestelmätestauksen hyväksymiskriteeri voi liittyä esimerkiksi järjestelmän virheiden määrään. Moduulitestauksen hyväksymiskriteerinä voi toimia määritelty testauksissa suoritettavan ohjelmakoodin kattavuusprosentti. (Haikala & Märijärvi 2004, 293-294)

Testaussuunnitelmassa kerrotaan, mitä testataan, milloin testataan, miten testataan ja millaisia lopputuloksia odotetaan. Testauksen lopettamiskriteerit määritellään myös testaussuunnitelmassa. Lopettamiskriteerinä voi esimerkiksi olla ajankohta,

määritelty prosentuaalinen osuus testitapauksista suoritettu tai, että kaikki löytyneet viat on korjattu. (Haikala & Märijärvi 2004, 299-300)

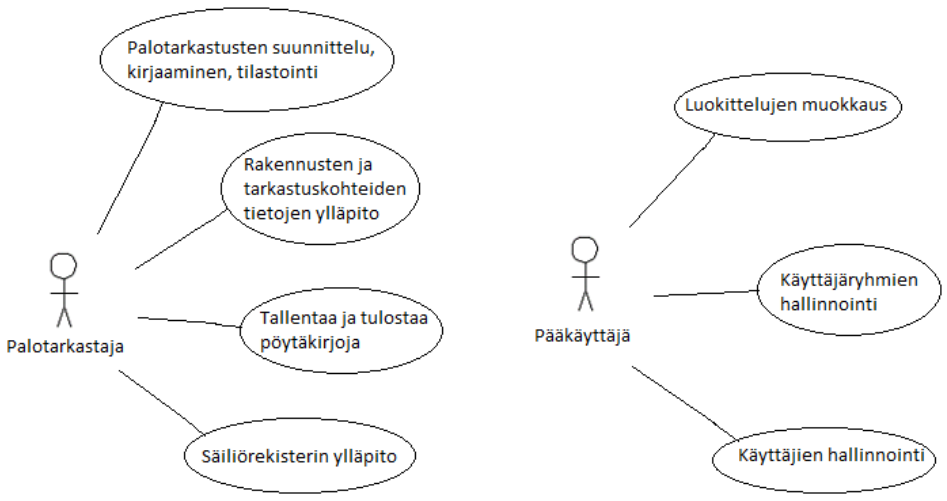
Löydetyt viat tulisi raportoida ja analysoida. Raportissa tulee mainita virheen kuvaus, vakavuus, löydön ajankohta, miten virhe olisi voinut löytää aikaisemmin sekä miten virheen olisi voinut estää. Virheitä voi raportoida virheraporttiin tai virhepäiväkirjaan. (Haikala & Märijärvi 2004, 300).

3 TESTAUKSEN TAUSTA JA TARKOITUS

3.1 Merlot Palotarkastus

Merlot Palotarkastus on sovellus, jonka tarkoitus on auttaa pelastuslaitoksia palotarkastustoiminnassa. Sovellus koostuu useasta pienemmästä osasta, jolla on yhteinen tekninen alusta. Itse Palotarkastusohjelma ja sen rakennusrekisteri toimivat sovelluksen runkona. Ohjelmassa hallinnoidaan rakennusrekisteriä, joka myös toimii keskeisenä rekisterinä ohjelman karttaosiossa. Rakennusrekisteri saadaan Väestörekisterikeskuksesta (VRK), ja koostuu kuntien VRK:lle lähettämistä rakennusrekisteritiedoista. Integroituja osasovelluksia ovat Karttaliittymä (MapTP), Väestönsuojelun suunnittelu (VSS) sekä Operatiivinen kohdekortti. Ohjelman peruskartta-aineisto toimii Maanmittauslaitoksen maastotietokanta ja osoitteisto.

Uusia osasysteemejä ja toimintoja kehitetään yhteistyössä asiakaskäyttäjien kanssa. Kehitystyö tehdään tiiviissä yhteistyössä pelastuslaitosten sisäisen kehitysryhmän kanssa. Kehitysasioiden priorisointi tehdään yhteisillä suunnittelu- ja kehityspäivillä.



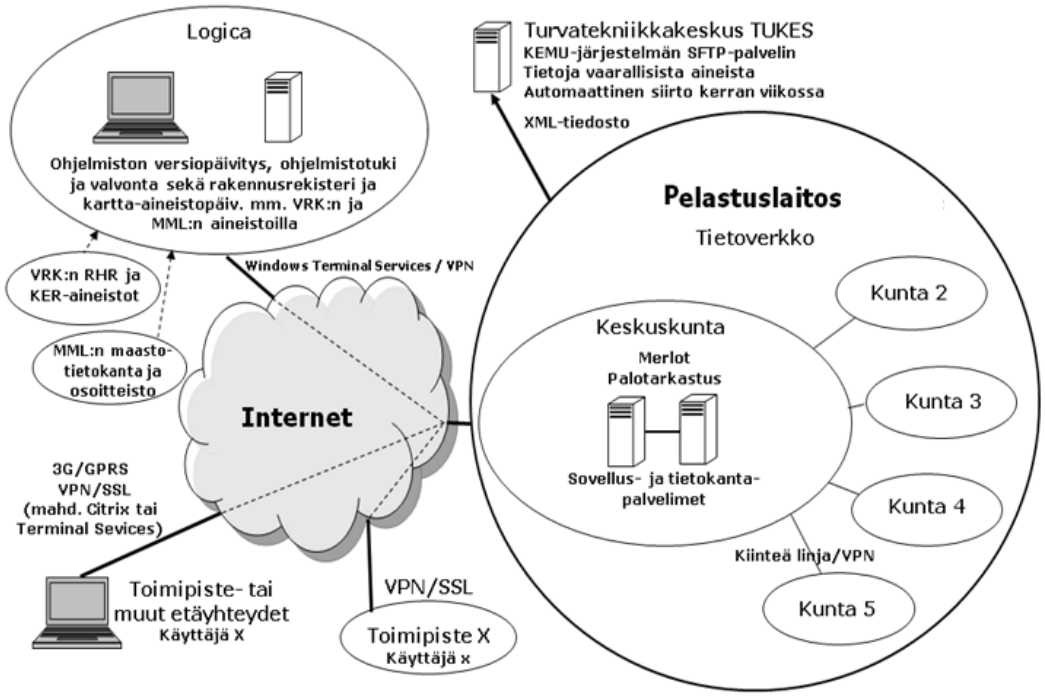
Kuvio 2. Merlot Palotarkastus –ohjelman käyttötapakaavio

Sovellus sisältää monta toimintoa ja ominaisuutta palotarkastuskohteiden käsittelyyn sekä tarkastusten rekisteröintiin. Merlot Palotarkastuksessa on toimintoja, jotka ovat tärkeitä suunnitteluun, rekisteröintiin ja statistiikan kirjaamiseen.

Käyttötapakaaviosta (Kuvio 2) näkee ohjelman päätoiminnot palotarkastajan ja pääkäyttäjän näkökulmasta. Merlot Palotarkastusohjelman tärkeimmät ominaisuudet yksityiskohtaisemmin ovat:

- rakennusrekisterin ylläpito
- palotarkastuskohdetietojen ylläpito
- palotarkastusten suunnittelu
- sammutus- ja pelastusvälineiden tietojen ylläpito
- erikoiskohdetietojen ylläpito
- säiliörekisteri
- oman valmiuden kohdetietojen ylläpito
- palotarkastuspöytäkirjojen rekisteröinti
- tarkastuspäivämäärän nopea rekisteröinti rakennusluetteloon
- integroitu karttaliittymä
- palotarkastushistoriikki
- tarkastusten seuranta
- statistiikka suoritetuista palotarkastuksista
- tulosteiden luonti (pöytäkirjat, peruskohdekortti, statistiikka jne.)
- tulosteet voi tulostaa tarkastajan mukaan sekä kunta-, toiminta-alue- tai pelastuslaitostasolla

- keskitetty käyttäjäryhmien ja -profiilien administraatio
- automaattinen käyttöliittymä Turvallisuus- ja kemikaaliviraston (TUKES) valvontakohderekisterin (KEMU) kemikaalivalvontaa varten.



Kuvio 3. Merlot Palotarkastus –ohjelmiston rakennekuvaus

Kuvasta (Kuvio 3) näkee millainen sovelluksen rakenne on. Logica päivittää ohjelmiston versiota, suorittaa ohjelmistotukea ja tekee esimerkiksi Maanmittauslaitoksen rakennusrekisterin päivitykset etäyhteydellä pelastuslaitoksen sovelluspalvelimille. Käyttäjät tarvitsevat internetiin yhteyden, jotta voivat käyttää sovellusta, sillä yhteys toimipisteestä palvelimeen vaatii sitä. Valvontakohderekisteri päivitetään kerran viikossa suoraan pelastuslaitoksen palvelimille SFTP-yhteydellä.

3.2 Pelastuslaki

Pelastuslakia sovelletaan tulipalojen ehkäisyyn, pelastustoimintaan sekä väestönsuojeluun. (Pelastuslaki)

Pelastuslaki (468/2003) korvataan vuoden 2011 alussa voimaan tulevalla uudella pelastuslailla, uudistamisen tavoitteina ovat:

- ”Pelastuslain onnettomuuksien ehkäisyä, väestönsuojelujärjestelmää, väestönsuojien rakentamista, pelastustoimintaa sekä pelastustoimien hallintojärjestelmää koskevien säännösten tarkistaminen.”

- ”Pelastustoimesta annettuun valtioneuvoston asetuksen tarkistaminen ottaen huomioon pelastuslakiin esitettävät muutokset.”

- ”Pelastuslain nojalla annettujen sisäasiainministeriön asetusten tarkistaminen ottaen huomioon pelastuslakiin esitettävät muutokset.”

Nämä uudistukset vaikuttavat Merlot Palotarkastus -sovellukseen, jota joudutaan päivittämään, jotta ohjelma olisi pelastuslain mukainen. Yksi esimerkki muutoksesta, on rakennusten tarkastusvälin lisääminen tulosteisiin. Lisäksi muutama avaintermi on vaihtunut. Esimerkiksi palotarkastusluokka on muuttunut kohdetyypiksi.

4 TESTAUS

Testauksen suunnittelua aloittaessa, tutkittiin määrittelydokumentit tarkasti läpi, jotta olisi tiedossa mitä testataan.

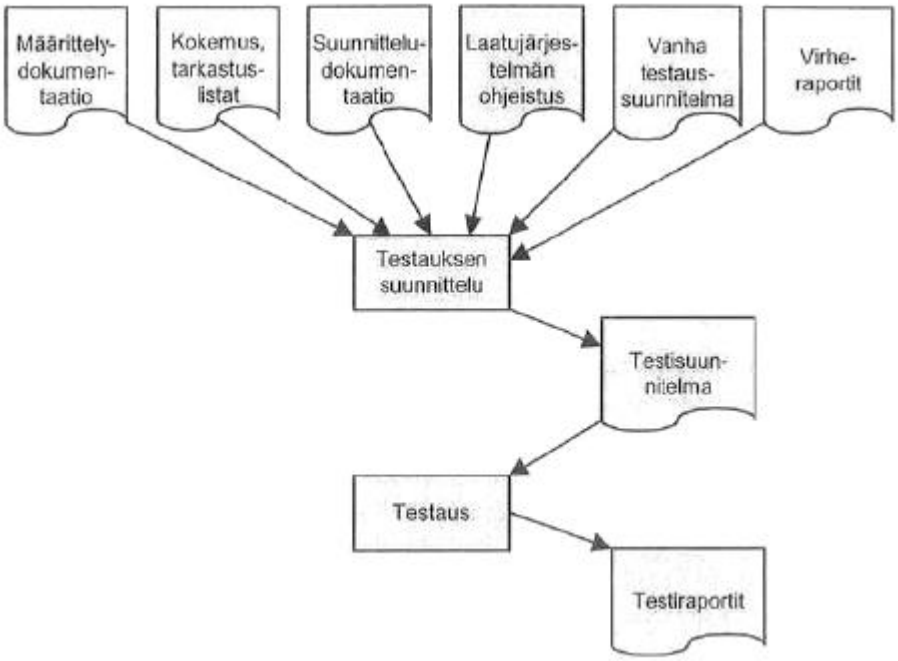
4.1 Testauksen tavoitteet

Testauksen tavoitteet laskevassa tärkeysjärjestyksessä ovat:

- Todeta että uudet ja muokatut toiminnot on tehty ja toimivat määritelmien mukaisesti.
- Suorittaa testaus 1.5.2011 mennessä. Vapun jälkeen sovellus menee asiakkaalle pilottitestaukseen, joten testaukset pitää olla suoritettuna ennen kuin tuote menee asiakkaalle.
- Todeta että ohjelman perustoiminnot edelleen toimivat.
- Luoda hyviä testausdokumentteja, joista voi ottaa mallia sovelluksen myöhempään testaukseen.

4.2 Testaussuunnitelma

Ennen kuin testaus voidaan aloittaa tarvitaan testaussuunnitelma. Kuten kuvio (Kuvio 2) kertoo, testauksen suunnittelussa pitää ottaa huomioon monta eri asiaa.



Kuvio 4. Testauksen suunnittelu (Haikala & Märijärvi, 298).

4.2.1 Testaussuunnitelman laatiminen

Määrittelydokumentaatio määrittelee pitkälti, mitä aiotaan testata. Kokemuksen avulla testien suunnittelija tietää, mihin kannattaa panostaa. Apuna testien painotuksessa toimivat myös tarkastuslistat, jos tarkastuksia on tehty pitkin projektia. Tarkastuksia ei olla suoritettu tässä projektissa. Suunnitteludokumentaatio kertoo määrittelydokumentaatiota yksityiskohtaisemmin, mitä projekti sisältää ja auttaa täten testauksen suunnittelua. Jos projektin pitää seurata laatujärjestelmää, pitää se ottaa huomioon myös testauksen suunnittelussa. Merlot Palotarkastuksen kehityksessä ei noudateta mitään laatujärjestelmää, joten testauksen suunnittelussa ei tarvitse ottaa laatujärjestelmää huomioon. Jos projektissa on vanha testaussuunnitelma, voi siitä ottaa mallia uudempiin. Virheraporteista näkee, missä virheitä on aikaisemmin ollut. Kun tietää missä on historiallisesti ollut paljon virheitä, tietää mihin kiinnittää huomiota tulevissa testeissä. Vanhoja testaussuunnitelmia, eikä virheraportteja, ei ollut olemassa Merlot Palotarkastus -sovellukseen.

IEEE 829 -standardia käytettiin testaussuunnitelman pohjana ja sitä yksinkertaistettiin sopimaan projektin tarpeisiin. Aikaisempaa testausdokumentaatiota ei ollut

saatavilla. Sovelluksen testausta on aikaisemmin tehty kehittäjien puolesta ilman kunnon dokumentaatiota. Työn testaussuunnitelma löytyy liitteistä (LIITE 1).

4.2.2 Testaussuunnitelma -dokumentti

Testaussuunnitelman versio

Versio	Tekijä	Päivämäärä	Kommentti
0.1	Mikko Palomaa	4.3.2011	Alustava runko otsikoineen sekä vähän sisältöä.
0.2	Mikko Palomaa	8.3.2011	Lisätty mitä testataan ja mitä ei testata.
0.3	Mikko Palomaa	9.3.2011	Määrittelydokumentit
0.4	Mikko Palomaa	23.3.2011	Lisää testattavia ominaisuuksia
1.0	Mikko Palomaa	25.4.2011	Virheiden korjaus

Viitteet

Testitapadokumentti:

- O:\Aluepelastuslaitos\Testaus\

Virheraporttidokumentti:

- O:\Aluepelastuslaitos\Testaus\

Määrittelydokumentit:

- O:\Aluepelastuslaitos\Testaus\Merlot uusia ominaisuuksia v.2.8.0.doc
- O:\Aluepelastuslaitos\Testaus\Merlot Valvontasuunnitelma-2010-12.pdf

- O:\Aluepelastuslaitos\Testaus\Valvontasuunnitelma Merlotiin määrittely-201012.doc
- O:\Aluepelastuslaitos\Toimintojen suunnittelu\Kuvia ja ohjeita\ohjelmointiohjeita\kehityslista\Valvontasuunnitelma\Valvontasuunnitelma Merlotiin määrittely.doc

Johdanto

Tämän testaussuunnitelman tarkoituksena on määritellä Merlot Palotarkastusohjelman (PT) testausta. PT:tä on päivitetty runsaasti, johtuen Palolain muutoksista. Nämä palolain vaatimat muutokset on saatava testattua ja täten varmistaa että PT:n muutokset toimivat oikein, sekä myös että ohjelman muut osat edelleen toimivat.

Testattavat ominaisuudet

Tehdään järjestelmätestaus, joka suoritetaan testitapauksia noudattaen käyttöliittymän kautta.

Määrittelydokumentissa ”Merlot uusia ominaisuuksia v.2.8.0.doc” on lista Merlot Palotarkastus –ohjelman muutoksista, joita testataan. Kyseisessä dokumentissa lukee tarkemmin muutoksista.

1. Uusi toiminto: Kohteiden luokittelu
2. Uusi tilasto: Prontotilasto 2
3. Uusi tilasto: Valvontasuunnitelma1
4. Uusi tilasto: Valvontasuunnitelma2
5. Uusi tuloste: Tarkastussuunnitelma kohteittain
6. Uusi toiminto: Tarkastusvälien ylläpito
7. Uusi toiminto: Käyttötarkoitusryhmien ylläpito
8. Muutoksia: Käyttötarkoitusten ylläpito
9. Muutoksia: Tarkastusten ylläpito
10. Muutoksia: Tarkastusten suunnittelu
11. Muutoksia: Tarkastuspvm toiminto
12. Muutoksia: Tarkastuskohdelista

13. Muutoksia: Tarkastusluettelo
14. Muutoksia: Lausuntoluettelo
15. Muutoksia: Tarkastuskohteiden tiedot
16. Muutoksia: Palotarkastusilmoitukset
17. Muutoksia: Tarkastusten auditoinnit
18. Muutoksia: Palotarkastusilmoitukset
19. Muutoksia: Alueiden päivitys
20. Muutoksia: Tarkastuskohteen tason korjaus
21. Muutoksia: Tarkastukset lajeittain ja pt-luokittain
22. Muutoksia: Uuden rakennuksen lisäys

Ei-testattavat ominaisuudet

Järjestelmätestauksessa, joka suoritetaan, ei testata ei-toiminnallisia ominaisuuksia: kuormitustesti, luotettavuustesti, asennustesti, käytettävyystesti jne.

Lähestymistapa

Muutoksien määrittelyjen perusteella luodaan testitapauksia testausta varten. Testitapaukset jaetaan ohjelman eri osien mukaan eri dokumentteihin. Muutosten lisäksi testataan ohjelman perustoimintoja, jos resurssit sallivat.

Kun testitapauksia testataan läpi, kirjataan löydetyt virheet virheraporttiin. Virheraportissa kerrotaan kuka löysi virheen sekä yksityiskohtaisesti miten virhetilaan päästiin.

Testien keskeyttämis- ja jälleenkäynnistyskriteerit

Jos testeissä löytyy virhe joka vaikuttaa jäljelle jääneiden testien läpivientiin, voidaan testaus pysäyttää kunnes kyseinen virhe on korjattu.

Testauksen lopputuotteet

Tämän testausprojektin tuotteita tulevat olemaan:

- Testaussuunnitelma
- Testitapauksia
- Virheraportti

Testausympäristö

Testauksessa käytetään APL2-testaustietokantaa, joka myös on kehittäjien käytössä kehitysvaiheessa. Testaus suoritetaan testaajien omilla työkoneilla.

Henkilö- ja koulutustarpeet

Testaukset suorittavat ohjelman kehittäjät sekä määrittelyjen laatija. Koulutuksen tarvetta ei siis ole.

Vastuut

Testisuunnitelman sekä testitapaukset laatii Mikko Palomaa. Testaaja toimii Mikko Palomaa.

Aikataulu

Testit ja testeissä löydettyjen virheiden korjaaminen tulisi olla suoritettuna 1.5.2011 mennessä. Toukokuun alussa ohjelma menee jo muutamalle asiakkaalle pilottivaiheeseen, joten siihen mennessä pitää kaikki olla valmiina.

Hyväksyntä

Testaus on suoritettu, kun sen on hyväksynyt sovelluksen tuotepäällikkö, Selim Backman.

4.3 Testauksen toteutus

Ensimmäiset viikot kuluivat lähteiden, testausopusten ja muiden vastaavien opinnäytetyöraporttien etsinnässä ja lukemisessa. Lisäksi käytiin läpi Merlot Palotarkastus -ohjelman muutosten määrittelydokumentteja. Löydetyn tiedon perusteella laadittiin testaussuunnitelma. Testaussuunnitelmasta lisää luvussa 5.1 Testaussunnitelma.

Seuraavaksi laadittiin testitapauksia, määrittelydokumenttien mukaan, jotka käyvät läpi ohjelman muutokset. Ohjelman muutokset olivat suurimmalta osin tehtynä ennen kuin testitapausten laatiminen alkoi. Tästä johtuen, testaaminen tapahtui samaan aikaan kun testitapaukset luotiin. Testitapauksista lisää luvussa 5.2 Testitapaukset ja virheraportista lisää luvussa 5.3 Virheraportti.

Testattaessa, löytyi 11 virhettä. Mikään virheistä ei ollut kovin vakavaa laatua. Suurin osa virheistä koski termimuutoksia, eli ohjelmassa esiintyi vielä vanhoja termejä. Lisäksi ohjelman opastusta ei oltu vielä ehditty päivittämään, eikä uusiin toimintoihin lisätty opastusta ollenkaan. Vakavin virhe, mikä löytyi, oli että sovelluksen käyttöliittymä piti käynnistää uudestaan, ennen kuin käyttötarkoitusryhmän muutokset esiintyivät toisessa osassa ohjelmaa. Vähemmän vakava virhe oli, että ohjelma antoi kaksi virheilmoitusta, kun vain yhden pitäisi esiintyä eräässä virhetilanteessa. Lisäksi ohjelmassa oli hieman eroja siinä, miten ohjelma käyttäytyy samankaltaisissa tilanteissa eri puolilla ohjelmaa.

4.3.1 Testitapaukset

Testitapaukset ohjeistavat askel askeleelta, miten ohjelman ominaisuuksia testataan. Jos testi tarvitsee jotain esiasetelmaa, siitä mainitaan ennen testitapausta. Testitapaukset myös kertovat, miten ohjelma käyttäytyy tai mitä tulisi tapahtua, kun testiaskeleita suoritetaan. Testitapauksissa on monesti myös tyhjä tila, mihin voi täyttää, mitä testissä tapahtui. Tämä tyhjä tila on kuitenkin usein vain testaajan omille muistiinpanoille, ja jos testissä sattui virhe, se raportoidaan erikseen virheraporttiin.

Lajittelin testitapaukset määrittelydokumenttien muutosten otsikoiden mukaan, kolmeentoista eri testitapausdokumenttiin:

- 001 – Tarkastusväli muutetaan kohdekohtaiseksi: Tarkastusvälien hallinnointi testataan.
- 002 – Käyttötarkoitukset: Käyttötarkoitusten sekä käyttötarkoitusryhmien hallinnoinnin testaus.
- 003 – Tarkastusten ylläpito: Testaa tarkastusvälin muuttamista sekä uusia perusarvoja valintalistoihin.
- 004 – Termimuutoksia: Testaa termimuutoksia muutamassa eri paikassa ohjelmaa.
- 005 – Kohteiden luokittelu: Uuden toiminnon testaaminen. Toiminnolla voi muuttaa kohdetyyppiä ja tarkastusväliä monelle kohteelle samaan aikaan.
- 006 – Valvontasuunnitelma: Testaa tilastoon tehtyä hakukriteerin muutosta.
- 007 – Valvontasuunnitelma 10v: a: Testaa tilastoon tehtyä hakukriteerin muutosta.
- 008 – Tarkastusten suunnittelu: Testaa uusia hakukriteerejä sekä suunnitellun tarkastuspäivän laskemista.
- 009 – Tarkastuspvm: Testaa suunniteltujen tarkastusten historiataulua.
- 010 – Tarkastuskohdelista: Testaa uusia hakukriteerejä.
- 011 – Tulosteet: Testataan että tarkastusväli on lisätty kaikkiin tulosteisiin, joissa on valintana kohderyhmä.
- 012 – Prontotilasto2: Testataan uutta tilastoa.
- 013 – Valvontasuunnitelma kohteittain: Testataan uutta tulostetta.

Testitapauksiin laitettiin täytettäväksi dokumentin alkuun testitapauksen nimen, testitapauksen laatijan sekä testitapauksen laadinnan päivämäärän. Itse testitapaukset täytetään taulukkoon, jossa on neljä saraketta ja niin monta riviä kuin on tarvetta. Sarakkeiden otsikot ovat: ID, Syöte, Odotettu tulos ja Tulos. ID-kenttä auttaa tunnistamaan testitapauksen ja ID voidaan mainita esimerkiksi toisessa testitapauksessa tai virheraportissa. Jos tiettyä ominaisuutta testataan monella eri osasyötteellä, tai jos jokin testi vaatii monta osa-askelta, voidaan testitapauksen ID jakaa pienempiin osiin. Esimerkiksi 001.1 testitapaus jaetaan osiin 001.1.1, 001.1.2 ja niin edelleen. Syöte-kentässä kerrotaan, mitä testitapauksessa tehdään. Odotettu tulos-kenttä kertoo, miten ohjelman odotetaan reagoivan syötteeseen. Tulos-kenttä on testaajan omia muistiinpanoja varten. Esimerkkinä testitapauksesta, testitapaus ”001 - Tarkastusväli muutetaan kohdekohtaiseksi”, löytyy liitteistä (LIITE 2).

4.3.2 Virheraportti

Virheraporttia täytetään, kun testaamisessa löydetään virheitä. Virheraportti näyttää erittäin samannäköiseltä kuin testitapaukset. Testitapausten neljän sarakkeen lisäksi, virheraportista löytyvät otsikot ”Pvm” sekä ”Testaaja”. Virheraporttiin laitoin täytettäväksi dokumentin alkuun virheraportin laatijan sekä virheraportin laadinnan päivämäärän.

Virheraportin taulukkoon täytetään ID-kenttään, mitä testitapausta noudattaessa virhe esiintyi. Syöte-kenttään ja Odotettu tulos-kenttään kopioidaan tekstit testitapauksesta. Jos syöte on riippuvainen aiemmista testitapauksista, niin siitä pitää lisätä informaatiota Syöte-kenttään. Tulos-kenttään kirjoitetaan, miten virhe esiintyy, niin yksityiskohtaisesti kuin pystyy. Pvm-kenttään merkitään, koska virhe löydettiin, ja Testaaja-kenttään, kuka löysi virheen. Liitteistä löytyy esimerkki yhdestä löydetystä virheestä virheraportissa (LIITE 2).

On erittäin tärkeää että tulos, päivämäärä ja testaaja täytetään tarkasti. Tulos on tärkeä siksi, että osataan toistaa virhetila. Päivämäärä pitää täyttää, jotta tiedetään, missä vaiheessa virhe on löydetty. Kun sovellus on testausvaiheessa ja suoritetaan regressiotestausta useamman kerran ja korjataan virheitä, sovelluksen versiot päi-

vittyvät tiheään tahtiin. Saattaa olla, että virhe on jo korjattu, mutta koska päivämäärä on epäselvä, virhettä joudutaan etsimään turhaan. Testaaja pitää olla täytettynä, jotta tiedetään missä olosuhteissa virhe esiintyy, mikäli virheraportin Tuloskentän sisältö on epäselvä.

5 JOHTOPÄÄTÖKSET JA POHDINTA

Testien suunnittelua ja testaamista alettiin tehdä vasta, kun palotarkastussovelluksen uudistusprojekti oli loppusuoralla. Täten testauksesta ei tullut aivan kirjaoppisuoritusta. Lisäksi on havaittu, että testitapausten lukeminen etukäteen, auttaa ohjelmoijaa havainnollistamaan, miten tietyn ominaisuuden pitää suoriutua, sekä mitä käyttöliittymältä odotetaan.

Testaajan pitäisi olla eri henkilö, kuin henkilö, joka on ohjelmoinut ohjelman. Ihminen on usein sokea omille virheilleen. Projektin testaaja ja testien laatija ei ollut tehnyt varsinaista kehitystyötä, joten itse testaaminen ja testauksen laatiminen oli varsin objektiivista.

Ensimmäinen testaaminen suoritettiin, aikataulusyistä, samalla kun testitapaukset luotiin. Tämä onnistui, koska testaamisen suunnittelu aloitettiin vasta, kun sovelluksen päivitysprojekti oli lähes valmis.

Testaamisessa ei löytynyt paljon virheitä. Suurin osa virheistä oli termimuutoksia, joita ei vielä oltu ehditty päivittämään sovellukseen. Tämä johtui siitä, että asiakas hyväksyi viimeisen määrittelyn erittäin myöhäisessä vaiheessa. Pieni virhemäärä johtuu kahdesta syystä. Ensimmäinen syy on ohjelman ikä. Ohjelma on ehtinyt olla markkinoilla niin kauan aikaa, että vakavat viat ovat karsiutuneet pois. Toinen syy pieneen virhemäärään on, että ohjelmasta on olemassa vain yksi tuotantoversio kerrallaan. Kun jokaiselle palolaitokselle ei ole erikseen räätälöityä sovellusta, jota pitää erikseen testata ja kehittää, niin virheiden määrä pysyy pienenä.

Testien suunnittelu aloitettiin varsin myöhään. Testaamisen suunnittelu pitäisi aloittaa projektin alkuvaiheessa määrittelydokumenttien avulla. Ohjelman kehittäjät voivat käyttää testidokumentaatiota, ymmärtääkseen uusia ominaisuuksia paremmin. Parempi ymmärrys kehitettävistä ominaisuuksista vähentää väärinkäsityksiä ja virheiden määrää.

Kuten käytännössä monesti tapahtuu, testaamiselle ei jäänyt kovinkaan paljon aikaa, kun projektin muut osa-alueet eivät olleet ajoissa valmiina. Tästä johtuen testaus aloitettiin myöhään. Lisäksi osa määrittelydokumentaatiosta sai asiakkaan hyväksynnän, kun projekti oli lähes valmis, joten osa termimuutoksista valmistui vasta ensimmäisen testikierroksen jälkeen.

Testauksen tavoitteet (luku 3.3) saavutettiin suurimmalta osin. Kaikki uudet toiminnot toimivat testeissä määritelmien mukaisesti, lukuun ottamatta osa termimuutoksista. Termimuutosvirheet ovat hyväksyttäviä, sillä ne eivät estä ohjelman käyttöä mitenkään. Testit saatiin suoritettua 1.5.2011 mennessä, kun testaus suoritettiin samaan aikaan kuin testitapaukset luotiin. Perustoimintoja ei ehditty testata juuri ollenkaan. Tämä on ainut testauksen tavoite, joka ei täyttynyt. Aikataulu ei yksinkertaisesti riittänyt testaamaan kaikkia perustoimintoja. Lisäksi aika ei riittänyt perustoimintojen testitapausten laatimiseen. Testausdokumentaatio ei ole monimutkainen, sitä on helppo käyttää pohjana tulevaisuuden testejä varten. Näin ollen on onnistuttu luomaan hyvä testausdokumentaatiopohja tulevia testejä varten.

LÄHTEET

EtestingHub 2007. Viitattu 15.3.2011. http://www.etestinghub.com

Haikala, Märijärvi 2004. Ohjelmistotuotanto. 10. uud.painos. Helsinki. Talentum.

Partanen 2009. Laadukkaan ohjelmistotestauksen piirteet. Viitattu 25.4.2011. http://urn.fi/URN:NBN:fi:amk-201003064971

Pyhäjärvi, Pöyhönen 2011. Ohjelmistotestaus. Viitattu 4.3.2011. http://www.testauskirja.com/materiaalit

Pelastuslaki 13.6.2003/468. Säädös säädöstietopankki Finlexin sivuilla. Viitattu 19.4.2011. http://www.finlex.fi/fi/laki/ajantasa/2003/20030468

Pelastuslain uudistushanke. Sisäasiainministeriö. Viitattu 19.4.2011. http://www.intermin.fi/suomi/pelastuslaki

Saarinen 2008. Testaus osana ohjelmistoprojektia. Viitattu 25.4.2011. http://urn.fi/URN:NBN:fi:amk-201003065011

Sorvo 2010. Testauksen liittäminen ohjelmistoprojektiin. Viitattu 25.4.2011. http://urn.fi/URN:NBN:fi:amk-2010110414166

LIITTEET

Liite 1. Testitapaus: 001 – Tarkastusväli muutetaan kohdekohtaiseksi

- **Testitapauksen nimi:** 001 – Tarkastusväli muutetaan kohdekohtaiseksi
- **Laatija:** Mikko Palomaa
- **Pvm:** 29.3.2011

ID	Syöte	Odotettu tulos	Tulos
001.1.1	Luokittelut => Tarkastus-luokittelut => Tarkastus-välit	Tarkastusvälit -ikkuna avautuu	
001.1.2	Tarkasta Tunnuksen va-lintalista.	Tarkastusvälit 6kk, 12kk, 24kk, 36kk, 48kk, 60kk, 96kk, 120kk, 180kk, 240kk sekä ”Ei erillinen tarkastuskohde” löytyvät valikosta.	
001.2.1	Paina Tarkastusvälit -ikkunan Uusi-painiketta.	Kaikki kentät tyhjenevät ja Uusi-painike muuttuu harmaaksi.	
001.2.2	Paina Peruutus-painiketta.	Ikkunaan ilmestyy jonkun talletetun tarkastusvälin tiedot.	
001.2.3	Paina Tunnus-kenttää.	Kaikki kentät tyhjenevät ja Uusi-painike muuttuu harmaaksi.	

001.2.4	Täytä kaikki kentät seuraavasti: Tunnus: 055 Nimi suomeksi: 55 kuukautta Nimi ruotsiksi: 55 månader Tark.väli: 55 Paina Tallenna-painiketta.	Ilmoitus-ikkuna, tulee esiin, jossa lukee ”Lisäys suoritettu!”. Sulje ilmoitus-ikkuna. Tarkastusvälit -ikkunan alareunassa lukee ”Lisäys suoritettu”. Tallennettu tarkastusväli jää esiin ja se löytyy nyt myös Tunnuksen vieressä olevasta valintalistasta nimellä ”055 55 kuukautta”.	
001.2.5	Poista tai lisää merkkejä kentässä Nimi suomeksi, Nimi Ruotsiksi tai Tark.väli	Tunnus, valintalista sekä Uusi-painike muuttuvat harmaiksi. Paina Peruuta-painiketta.	
001.2.6	Valitse valintalistasta yläpuolella luotu ”055 55 kuukautta” ja paina tämän jälkeen Poista-painiketta.	Vahvistus-ikkuna, ilmestyy, joka kysyy ”Haluatko varmasti poistaa?”	
001.2.7	Paina Vahvistus-ikkunan	Vahvistus-ikkuna sulkeu-	

	Ei-painiketta.	tuu ja valittu tarkastusväli on vieläkin valittuna.	
001.2.8	Valitse valintalistasta yläpuolella luotu ”055 55 kuukautta” ja paina tämän jälkeen Poista-painiketta. Paina Vahvistus-ikkunan Kyllä -painiketta.	Ilmoitus-ikkuna, tulee esiin, jossa lukee ”Poisto suoritettu”. Sulje ilmoitus-ikkuna. Tarkastusvälit -ikkunan alareunassa lukee ” Poisto suoritettu”. Ikkunaan ilmestyy jonkun muun talletetun tarkastusvälin tiedot eikä aiemmin luotua ”055 55 kuukautta”-tarkastusväliä enää löydy valintalistasta.	
001.3	Paina Tarkastusvälit -ikkunan Sulje-painiketta.	Tarkastusvälit-ikkuna sulkeutuu.	
001.4	Tallenna uusi tarkastusväli täyttämättä Tunnus-kenttää.	Ilmoitus-ikkuna, ilmestyy, jossa lukee ”Luokittelua ei voi tallentaa ilman tunnusta. Anna luokittelulle tunnus ja tallenna uudelleen.”	
001.5	Täytä Tunnus-kenttä täy-	Tunnus-kenttään mahtuu	

	teen erilaisilla merkeillä	vain 5 merkkiä.	
001.6	Täytä Nimi suomeksi sekä Nimi ruotsiksi – kentät täyteen erilaisilla merkeillä.	Kenttiin mahtuu 40 merkkiä.	
001.7	Täytä Tark.väli -kenttä täyteen merkeillä.	Kenttään mahtuu 3 numeroa. Kirjaimia ei voi täyttää.	

Liite 2. Virheraporttimalli

- **Virheraportti:** Testitapausten luonnin yhteydessä löytyneet virheet
- **Laatija:** Mikko Palomaa
- **Pvm:** 29.3.2011

Osa testitapauksista tarvitsee samassa testitapausdokumentissa löytyvän aiemman testitapauksen tietoja, jotta virhetilanne saadaan toistettua.

ID	**Syöte**	**Odotettu tulos**	**Tulos**	**Pvm**
002.7	Toista testitapaus 002.2.4. Luokittelut => Käyttötarkoitukset	Käyttötarkoitukset –ikkuna aukeaa. Käyttötark.ryhmä valintalistasta löytyy luotu vaihtoehto ” 055 Testi 55”.	”055 Testi 55” käyttötarkoitusryhmä ei ilmesty valintalistaan ennen kuin käynnistää Palotarkastus –ohjelman uudestaan.	29.3.2011