Kuntokirjuri

Testaussuunnitelma

Miika Alonen
Jarkko Laine
Jesse Honkanen
Veli-Matti Huovinen
Jani Jäntti

Versio 1.1

9.5.2008

Jakelu:

Asiakas - Jukka Rantala

Ohjaaja – Erkki Pesonen

Opponoiva ryhmä 1

Kuopion yliopisto

tietojenkäsittelytieteen laitos

Dokumentin versiohistoria:

Versio	Pvm	Tekijä	Muutos
0.1	28.2.2008	JH	Sisällön mietintä
0.2	3.3.2008	JH	Hieman tekstiä
0.3	4.3.2008	JH	Lisää tekstiä
0.4	6.3.2008	JH	Testitapauksia ja testauksen etenemisestä
0.6	7.3.2008	JH	Dokumentin jako lukuihin ja yhtenäistäminen
0.7	8.3.2008	JH	Testitapausten kirjaamista, kappaleiden uudelleenkirjoitusta
0.8	10.3.2008	JH, MA, VH	Korjailua, testitapausten taulukointia, ulkoasun muokkausta
0.9	10.3.2008	MA	Korjailua ja säätöä
1.0	11.3.2008	JH, VH	Arvosteltava versio. Testitapausten tarkennusta, viime hetken korjailuja, liitteiden liittäminen
1.1	9.5.2008	JH	Opponoinnissa löydettyjen virheiden korjaaminen, julkaisukuntoon laittaminen

Tekijöiden lyhenteet:

MA	Miika Alonen
JJ	Jani Jäntti
JH	Jesse Honkanen
VH	Veli-Matti Huovinen
JL	Jarkko Laine

SISÄLLYSLUETTELO

1 JOHDANTO

1.1 Tarkoitus ja kattavuus

Tässä dokumentissa on kuvattu Kuntokirjuriohjelman testausprosessin kaikki vaiheet. Dokumentissa kuvataan testauksen lähestymistavat, kattavuus, testien suorittaminen, testausvastuut, testauksessa syntyvät dokumentit ja aikataulu. Testitapaukset on kehitetty ohjelman vaatimusmäärittelyn sekä toiminnallisen ja teknisen määrittelyn pohjalta. Testauksen tarkoituksena on varmistua siitä, että toteutettu ohjelma vastaa spesifikaatioitaan ja toimii oikeellisesti. Tehtävän testauksen laatu riippuu käytettävissä olevasta ajasta ja testaajien osaamistasosta.

Kuvaus tehdystä testauksesta löytyy Testausraportti-dokumentista.

1.2 Yleiskatsaus dokumenttiin

Luvussa 2 on kuvattu testauksessa käytettävä ympäristö ja työkalut
Luvussa 3 on määritelty testauksen osallistujat ja vastuut
Luvussa 4 on kuvaus testauksesta syntyvistä dokumenteista
Luvussa 5 on kuvattu testauksen vaiheet ja testien suorittaminen
Luvussa 6 on arvioitu testauksen aikataulua ja työmäärää
Luvussa 7 on testauksessa käytettävät testitapaukset
Luvussa 8 on listattuna testauksen ulkopuolelle jäävät ominaisuudet
Luvussa 9 on testauksen hyväksymiseen ja hallintaan liittyvää tietoa
Liitteissä on testauspöytäkirjan malli ja järjestelmän vaatimukset

1.3 Testauksen tavoitteet

Testauksen tavoitteena on spesifikaatioiden mukainen ja vakaa ohjelma. Tavoitteena on löytää ja korjata ohjelman toiminnan kannalta vakavimmat virheet. Testauksella nostetaan tehdyn ohjelman laatua, mutta testaus voidaan itsessään nähdä myös laadullisena ilmiönä. Projektiryhmän kokemattomuudesta johtuen ei voida odottaa korkeaa testauksen laatua. Tavoitteena on siis myös, että projektiryhmä oppii jotain testauksesta ja sen merkityksestä. Huonosti menneistä asioista voidaan ottaa oppia tulevaisuuden varalle.

1.4 Käytetyt termit ja määritelmät

Testaus – Käsitteen laajuus riippuu käyttöyhteydestä, yleensä systemaattista virheiden etsintää ajamalla ohjelmaa

Testitapaus – Toimenpide tai toimenpiteiden sarja, jolla varmistutaan ohjelman oikeasta toiminnasta ja vaatimusten täyttämisestä

Mustalaatikkotestaus – Testaus, jossa testataan osan rajapintaa ilman tietoa sisäisestä toteutuksesta

Lasilaatikkotestaus – Testaus, jossa tarkastellaan ohjelman sisäistä toimintaa eri syötteillä

Regressiotestaus – Testitapausten uudelleen suorittamista ohjelman uudelle versiolle

Moduuli – Kokonaisuuden osa, jolla on määritelty liitäntä ympäristöönsä

Katselmointi – Dokumentin tai ohjelmakoodin läpikäymistä virheitä etsien

Verifiointi – ”Rakennammeko tuotetta oikein?” eli vastaako tuote spesifikaatioitaan

Validiointi – ”Rakennammeko oikeaa tuotetta?” eli tehdäänkö tuote vaatimusten ja toiveiden mukaan

Bugi – Ohjelmavirhe, ero oikean ja väärän ohjelman toiminnan välillä

2 TESTAUSYMPÄRISTÖ

Tässä luvussa on kuvattu testauksessa tarvittava ympäristö ja dokumentit. Luvusta löytyy myös ohjeet testaukseen valmistautumiseen.

2.1 Tarvittavat koneet ja ohjelmat

Testauksessa käytetään projektiryhmän jäsenten omia koneita, sillä Kuopion yliopiston koneilla ei ole Javan versiota 1.6.

Testausympäristöön tulee olla asennettuna **Javan versio 1.6** tai uudempi. Testauksessa voidaan käyttää **Netbeansin** ajotyökaluja, mutta ne eivät ole välttämättömiä jokaisessa vaiheessa. Virheenjäljitys tehdään samalla ohjelmalla kuin toteutuskin, eli Netbeansin versiolla 6.

Koska käytetyt ohjelmat ja Java ovat alustariippumattomia, ei testauksessa käytettävälle käyttöjärjestelmälle aseteta rajoitteita. Suositeltavaa olisi kuitenkin käyttää Windows-ympäristöä (XP tai Vista), sillä ohjelma on suunniteltu erityisesti sille. Ohjelman toimivuutta tulee kuitenkin testata myös Linux-ympäristössä viimeistään järjestelmätestausvaiheessa.

Testauksessa käytetään alkuun uudehkoja koneita, joista löytyy riittävästi muistia ja levytilaa. Testauksen loppupuolella voidaan toimivuutta kokeilla myös hitaammilla laitteilla.

Lisäksi testauksessa tarpeellisia asioita voivat olla sekuntikello nopeusmittauksiin ja muistiinpanovälineet testauspöytäkirjan pitoon. Suorituskykyä voidaan mitata myös erillisillä ohjelmilla.

2.2 Tarvittavat tiedostot

Testaukseen tarvitaan tämän dokumentin tuorein versio ja testipöytäkirja, johon kirjataan tehtävät testit. Molemmat määrittelydokumentit on hyvä olla saatavilla, sillä niistä voi katsoa testauksen kannalta oleellisia asioita.

Testaukseen tarvitaan myös tuorein ohjelmaversio. Tuoreimman ohjelmaversion pystyy ainakin testauksen loppuvaiheessa lataamaan projektin **SourceForge** -sivulta [1].

Lisäksi useimmissa testauksissa voidaan käyttää tiedostoa, johon on kirjoitettu valmiita tietokantaan tehtäviä syötteitä. Näin vältytään tarpeelta keksiä itse syötettäviä esimerkkiarvoja. Tämä tiedosto löytyy sovitusta paikasta.

Testauksessa voidaan tarvita myös aiemmin tehtyjen testien tiedostoja ja tuloksia. Sen vuoksi aiemmista testeistä on hyvä säilyttää kaikki mahdollisesti tarpeellinen.

2.3 Testauksen valmistelu

Ennen testausta varmistetaan, että tarvittavat ohjelmat on asennettuna ja tarvittavista dokumenteista on saatu uusimmat versiot. On myös hyvä varmistaa, että käytössä on riittävästi levytilaa ja muistia. Testausta varten luodaan levylle oma hakemisto, johon tiedostot tallennetaan. Ennen testausta on päätettävä, mitä ohjelman osaa tai ominaisuutta testataan ja miten kattavasti. Testauspöytäkirjaan voi kirjoittaa tehtävät testitapaukset ennen testauksen aloittamista, jotta testaus sujuu joutuisammin. Testausta varten on ohjelma aloitettava puhtaalta pöydältä, niin ettei edellisestä testauksesta jää jäänteitä.

3 TESTAUKSEN ORGANISOINTI JA RAPORTOINTI

3.1 Testausryhmä

Testaukseen osallistuvat projektiryhmän jäsenet, opponoiva ryhmä, asiakas ja mahdollisesti joitain ulkopuolisia henkilöitä. Jokainen projektiryhmän jäsen ei osallistu kaikkiin testausvaiheisiin. Toteuttajan ei tulisi testata omaa koodiaan, mutta joudumme joustamaan tästä periaatteesta testausprosessin nopeuttamiseksi. Onhan toteuttaja lopulta itse vastuussa moduulinsa toimivuudesta. Toteuttaja pystyy myös helpommin ja nopeammin paikantamaan ja korjaamaan virheet. Uudelleentestauksen voi tehdä joku muu kuin toteuttaja. Testaus voidaan tehdä myös parityönä, jolloin toinen parin jäsen on toteutuksen tehnyt henkilö. Kenelläkään ryhmän jäsenellä ei ole aiempaa kokemusta oikeasta testauksesta, joten testauksesta ei tehdä liian monimutkaista ja ammattimaista.

3.2 Vastuualueet henkilöittäin

Jesse Honkanen

- Testauksen suunnittelu
- Testaussuunnitelman ylläpito
- Testausraportin laadinta
- Koodien katselmoinnit
- Tietokannan testaus
- Järjestelmätestaus

Miika Alonen (käyttöliittymän toteuttaja)

- Integraatiotestaus
- Ohjelman kokoaminen järjestelmätestausta varten
- Korjaukset ohjelmaan

Jani Jäntti (tietokannan toteuttaja)

- Tietokannan testaus
- Tietokantakorjausten tekeminen
- Järjestelmätestaus

Jarkko Laine (projektipäällikkö, käyttöliittymän toteuttaja)

- Käyttäjätestauksen järjestäminen ja valvonta
- Vastuu ohjelman käytettävyydestä
- Testausvastaava (hallinnointi ja vastuu)
- Järjestelmätestaus

Veli-Matti Huovinen (käyttöohjeen laatija)

- Esimerkkisyötteet
- Testausavustaja
- Järjestelmätestaus
- Tietokantatestaus

4 VAADITTAVA TULOSAINEISTO

Testaustilanteessa sattuneet tapahtumat kirjataan testauspöytäkirjaan (malli liitteessä 1). Testauspöytäkirjaan tulee testaajan nimi, testitapaukset, ilmoitus vastasiko tulos odotuksia ja viittaukset virheraportteihin. Testauspöytäkirjat toimitetaan sovittuun paikkaan ryhmän saataville.

Testauksesta syntyvät lokitiedostot otetaan talteen testaushakemistoon testaajan levylle. Myös testauksen aikana syntyneet muut tiedostot, kuten tietokannat ja tietokantaraportit, tallennetaan. Näitä ei välttämättä tarvitse laittaa muiden saataville, mutta ne voivat osoittautua hyödyllisiksi.

Testauksen aikana on hyvä ottaa kuvaruutukaappauksia sopivista tilanteista. Näitä kuvaruutukaappauksia voidaan hyödyntää käyttöohjeessa tai projektin verkkosivulla. Myös kummallisista virhetilanteista voidaan ottaa kuvaruutukaappaukset ja liittää nämä osaksi testauspöytäkirjaa.

Testauksesta kootaan tiivistelmä yhteen testausraporttiin. Testausraporttiin poimitaan keskeiset asia testauspöytäkirjoista ja virheraporteista. Testausraporttiin tulee myös arvioita testauksen onnistumisesta, testauksessa tehdyistä virheistä ja testauksen kattavuudesta.

4.1 Virheraporteista

Virheiden raportoinnissa käytetään hyödyksi **SourceForge**:ssa [1] olevia toimintoja. Testauksessa löytyneistä virheistä tehdään yksilöidyt virheraportit sivustolle. Osan toteuttaja on ensisijaisesti vastuussa virheiden korjaamisesta. Tilanne ei muutu vaikka toteuttaja testaisikin itse. Testaus on silti dokumentoitava ja mielellään myös virheraportit täytettävä. Virheraporttien täyttö voi auttaa myöhemmin ohjelmaa testattaessa, sillä niistä nähdään mitä on korjattu ja miten. Pienistä, esimerkiksi kosmeettisista virheistä ei välttämättä tarvitse täyttää omia virheraporttejaan vaan virheiden listaus samaan tiedostoon voi riittää. Virheraporttien luonnissa kannattaa muutenkin käyttää omaa harkintaa (kuten koko testauksessa ylipäätään).

Löydetyt virheet nimetään seuraavasti:

BUG<virheen tyyppi>.<juokseva nro>@<testitapauksen tunniste>_<testaajan nimi>

Virheen tunniste voi siis olla esimerkiksi BUG2.1@TC2.1-1_Jesse

Testitapauksen tunniste kirjoitetaan kokonaisuudessaan. Jos löydetään virhe, joka ei liity mihinkään testitapaukseen kirjoitetaan uusi testitapaus tai jätetään kylmästi koko testitapauksen tunniste pois. Juoksevan numeroinnin käytössä täytyy olla tarkkana, ettei kaksi virhettä saa samaa tunnistetta.

Virheiden hallinnassa voidaan käyttää myös omaa järkeä, mutta tehdyt korjaukset on hyvä merkitä sivustolle.

Virheiden tyypit:

1. Metodin virheellinen palautusarvo
2. Ristiriita vaatimusten kanssa
3. Ristiriita suunnitelmien kanssa
4. Virheellinen tuloste
5. Virheellinen testitapaus
6. Käytettävyysvirhe
7. Muu käyttöliittymävirhe
8. Edellisiin sopimaton virhe

Virheiden tunniste laitetaan täytetyn lomakkeen otsikkokenttään, jotta virheiden hallinta helpottuu.

Koodin **katselmoinnissa** löydetyt virheet voidaan kirjoittaa kaikki samaan tiedostoon eli virheraportteja ei ole pakko luoda. Tiedostoon kirjoitetaan koodin versionumero, metodinnimi ja rivi josta virhe löydettiin sekä virheen kuvaus. Myös muita tarpeellisia tietoja voidaan kirjoittaa ylös. Katselmoinnissa tarkkaillaan koodausvirheiden lisäksi muuttujien nimeämistä ja käyttöä, koodin selkeyttä sekä koodin kommentointia. Myös turhan koodin paikantaminen kuuluu katselmointiin. Katselmoinnista ei välttämättä tarvitse kirjoittaa edes tiedostoa, vaan riittää palauttaa toteuttajalle tulostettu koodi, josta löytyy merkintöjä. Erillisen dokumentin kirjoittaminen on kuitenkin suositeltua.

5 TESTAUSSTRATEGIA

Ohjelma testataan osissa edeten erillisten osien testauksesta koko järjestelmän testaukseen. Tässä luvussa on määritelty testattavat osat ja testauksen vaiheistaminen. Luvussa on myös ohjeita eri testausvaiheiden läpivientiin.

5.1 Testattavat osat

Testattavia osia on tiivistettynä kaksi: tietokanta ja käyttöliittymä. Raporttitoiminnon voi ajatella olevan myös erillinen osa ohjelmaa. Ohjelma ei sisällä tietokantametodien ja tapahtumankäsittelijöiden lisäksi juuri muuta koodia, joka vaatisi testausta. Tietokantametodit katselmoidaan ja testataan kattavasti. Käyttöliittymän tapahtumankäsittelijät katselmoidaan ja niiden toimintaa testataan integrointitestausvaiheessa. Järjestelmätestausvaiheessa testataan järjestelmän toimintaa kokonaisuutena. Raportit testataan vasta integrointivaiheessa tai järjestelmätestausvaiheessa.

5.2 Testausten suorittaminen

Ohjeet testaukseen valmistautumiseen löytyy luvusta 2. Valmistelun jälkeen koneella tulee olla testattava osa valmiina testaukseen ja testauspöytäkirjan tulee olla avoimena. Useimmiten testaus on helpointa suorittaa yhdeltä istumalta, jolloin testaukseen tulee varata riittävästi aikaa. Suunniteltu testaus ei saa olla summittaista testitapausten suoritusta vain tarkoituksena aiheuttaa ohjelman kaatuminen - apina- ja gorillatestaus on asia erikseen. Testitapaukset suoritetaan järjestyksessä ensiksi hyväksyttävät syötteet ja sen jälkeen eri tavoin virheelliset syötteet. Saadut tulokset kirjataan ylös pöytäkirjaan ja virheiden löytyessä täytetään virheraportti. Testauspöytäkirjaan tulee testitapauksen tunnus ja saatu tulos, sekä ilmoitus vastaako tulos odotettua. Testauksen aikana keksityistä uusista testitapauksista kirjoitetaan kuvaus testauspöytäkirjaan. Väärä asenne testaukseen on sellainen, että vain suunnitellut testitapaukset suoritetaan. Suunnitteluvaiheessa tehdyt testitapaukset eivät välttämättä ole läheskään kattavia, joten testaajan oma aivotoiminta on hyvin tärkeää. Harkintaa voidaan käyttää myös testitapausten karsimisessa ja tärkeydessä.

Bugin löytyessä voi testaaja itse selvittää missä tämä virhe sijaitsee, mutta kesken testausta tehtävät korjaukset eivät ole suotavia. Tästä periaatteesta voidaan joustaa vain, jos testaaja on itse vastuussa myös korjausten tekemisestä ja korjauksilla on kiire. Korjaukset tehdään yleensä vasta, kun ollaan saatu ajettua läpi sen kerran aiotut testitapaukset. Virheen sijainti kirjoitetaan ylös virheraporttiin, jotta sen korjaus on helppoa myöhemmin.

Ohjeet testauksen hyväksyntään, keskeyttämiseen tai lopettamiseen löytyy luvusta 9.

5.3 Regressiotestaus

Regressiotestauksessa suoritetaan vanhoja testitapauksia uudelleen. Regressiotestauksen tarkoituksena on varmistua, että ohjelmaan tehdyt muutokset ovat korjanneet ongelman eivätkä ole aiheuttaneet uusia ongelmia. Jokainen testausvaihe on siis hyvä suorittaa vähintään kahteen kertaan. Regressiotestauksessa ei ole välttämätöntä suorittaa kaikkia testitapauksia uudestaan. Testauspöytäkirjasta voidaan poimia tarpeelliset asiat uudelleentestattaviksi. Projektissa regressiotestausta tehdään tarpeen mukaan käsityönä. Automatisoitua testausta ei ole tarkoitus kehittää, mutta esimerkiksi Junit-ohjelma tarjoaa mahdollisuuden siihen.

5.4 Moduulitestaus

Ennen varsinaista testausta joku muu kuin toteuttaja tarkastaa koodin. Katselmoinnissa löytyneet virheet listataan tiedostoon ja korjataan. Sekä tietokantamoduulin metodit, että tapahtumankäsittelijät katselmoidaan.

Tietokantamoduulin toimintaa päästään testaamaan katselmoinnin jälkeen. Tietokantamoduulin metodit testataan kattavasti, sillä tietokannan oikea toiminta on kriittistä. Tärkeimmät metodit vaativat tarkemman testauksen kuin vähemmän tärkeät. Testausta varten on tehtävä joko erillinen ajuriohjelma tai käytettävä Javalle sopivia testiohjelmia kuten JUnit-työkalua [2]. Ensisijaisesti testaus tehdään mustalaatikkotestauksena, mutta testitapausten keksimisessä voidaan käyttää apuna lähdekoodia. Virheiden paikantaminen vaatii tietenkin lähdekoodin tarkastelua. Testauksessa on hyvä tarkkailla myös syntyviä tiedostoja ja niiden päivitystiheyttä, jotta pystytään sanomaan, milloin tiedot on kirjoitettu levylle.

Käyttöliittymää ei ole välttämätöntä testata tarkasti vielä tässä vaiheessa, mutta karkeat toteutusvirheet ja suuret käytettävyysongelmat on hyvä korjata. Testauksessa keskitytään käyttöliittymän ulkoasuun ja käytettävyyteen. Testauksessa voidaan käyttää apuna esimerkiksi käyttöliittymän testitapauksia ja toiminnallisuuksiin liittyviä testitapauksia. Mikäli aikaa on käytettävissä, voidaan käyttöliittymälle kehittää oma tynkäluokka tietokannasta. Tehtyjä testejä ei ole välttämätöntä dokumentoida jos aika ei riitä. Ryhmä ja mahdollisesti myös asiakas hyväksyvät käyttöliittymän ennen kuin voidaan siirtyä testauksessa eteenpäin.

5.5 Integrointitestaus

Integrointitestauksessa liitetään tietokanta kiinni käyttöliittymään. Ennen testien suorittamista kannattaa katselmoida läpi tietokantaan liittyvät tapahtumankäsittelijät ja poistaa mahdollinen turha

koodi. Testauksessa keskitytään tarkkailemaan, että käyttöliittymä käyttää tietokantaa oikein. Testaus suoritetaan käyttöliittymästä käsin. Tässä testauksen vaiheessa keskitytään tietojen lisäämiseen, muokkaamiseen ja hakemiseen. Lisäksi testataan tyyppien ja attribuuttien hallinta. Muiden tietokantatoimintojen testaus voidaan jättää järjestelmätestaukseen. Koska tietokantakomponentin toiminta on testattu jo moduulitestausvaiheessa, voidaan tietokannan odottaa palauttavan järkeviä vastauksia. Tässä vaiheessa on tärkeää kiinnittää huomiota myös poikkeustilanteiden käsittelyyn ja käyttäjälle annettaviin ilmoituksiin.

Raporttityökalun testauksen voi tehdä vasta integrointivaiheessa, sillä JasperReports[3] lukee tiedot suoraan käytetystä JavaDB-tietokannasta. Suora tietokannasta lukeminen voi aiheuttaa monenlaisia ongelmia: käytetyt SQL-kyselyt voivat olla väärin muotoiltuja tai suoraan tietokannasta luetut tiedot näyttävät vääränlaisilta muotoilematta. Raporttityökalua testattaessa saatu raportti on tärkeässä asemassa ja sen vuoksi saadut raportit on hyvä tallentaa levylle. Raportista tarkastellaan osien sijoittelua, muotoilua, graafien ulkoasua ja tulostettavuutta.

5.6 Järjestelmätestaus

Järjestelmätestauksessa koko järjestelmää testataan kokonaisuutena. Järjestelmätestauksessa testataan kaikki ohjelman toiminnot ja tarkastetaan vastaavatko toiminnot kirjattuja vaatimuksia ja onko ne toteutettu suunnitelmien mukaan. Tässä vaiheessa testataan myös järjestelmän muita ominaisuuksia, kuten asennusta, käytettävyyttä, suorituskykyä, turvallisuutta, siirrettävyyttä ja vakautta. Käytännössä suurin osa testitapauksista on suunniteltu järjestelmätestausta varten.

Järjestelmätestausta varten ohjelmasta on oltava olemassa toimiva versio, joka voidaan antaa muillekin testattavaksi. Tässä versiossa tulee olla mukana asennusohjelma ja valmis käyttöohje. Järjestelmätestaukseen osallistuu projektiryhmän lisäksi opponoiva ryhmä ja asiakas, joten tapahtuu paljon samanaikaista testausta. Kaikki järjestelmätestaukseen osallistujat saavat ohjelman mukana käyttöohjeen lisäksi tämän dokumentin. Opponoiva ryhmä ja asiakas keräävät omaa listaa löytämistään ongelmista ja toimittavat korjausehdotuksensa sovittuun päivämäärään mennessä.

Järjestelmätestaus kannattaa jakaa osiin, niin että eri henkilöt voivat testata eri ominaisuuksia. Päällekkäistä testausta tulee välttää ajanpuutteen vuoksi. Virheidenhallinnan toimivuus on tässä vaiheessa erityisen tärkeää. Ohjelmasta ei ole hyvä julkaista uutta versiota kesken järjestelmätestauksen. Käytännössä joidenkin testien suorittaminen voi vaatia uutta ohjelmaversiota, joten virheiden korjausta ei voida jättää testauksen jälkeiseen aikaan. Korjauksia järjestelmään tehdään siis sitä mukaan, kun virheitä löytyy.

Koska asiakas osallistuu järjestelmätestauksen, kuuluu järjestelmätestaukseen osaksi hyväksymistestaus, jossa tarkastetaan onko kaikki ohjelman vaatimukset toteutettu hyväksyttävästi.

5.7 Käyttäjätestaus

Käyttäjätestaus tehdään järjestelmätestauksen jälkeen, mikäli aikaa on. Testaukseen otetaan mukaan loppukäyttäjän näkökulma. Käyttäjätestauksen tarkoituksena on etsiä erityisesti ohjelman käytettävyysvirheitä ja varmistaa, että ohjelma vastaa loppukäyttäjän tarpeita. Käyttäjätestaukseen osallistuu toteutuksen ulkopuolisia henkilöitä, joille järjestetään valmisteltu testaustilaisuus. Testaustilaisuudessa testihenkilöt testaavat ohjelman toimintaa ja projektiryhmän jäsenet tarkkailevat heidän toimiaan. Testihenkilöille voidaan antaa valmiiksi mietittyjä tehtäviä tehtäväksi. Testitilaisuudessa havaitut ongelmat analysoidaan ja dokumentoidaan mahdollisen jatkokehityksen varalle. Korjauksia ei ehkä pystytä tekemään enää projektin aikana. Käyttäjätestauksesta voidaan tehdä myös vähemmän muodollinen, jolloin erillistä testaustilaisuutta ei järjestetä. Käyttäjätestaukseen liittyvistä suunnitelmista päätetään myöhemmin.

Mikäli testaushenkilöt ovat tyytyväisiä ohjelmaan, voidaan projektia pitää onnistuneena ja projektiryhmä voi onnitella itseään hyvästä työstä. Sen sijaan, jos ohjelma koetaan tarpeettomaksi tai liian hankalaksi käyttää, voidaan miettiä mikä on mennyt vikaan.

Beta-testausta ei projektin aikana suoriteta, vaan beta-vaihe alkaa, kun testaus päättyy ja ohjelma julkaistaan verkossa. Projektiryhmä ei ole vastuussa enää betatestauksessa löytyneiden virheiden korjaamisesta.

6 TESTAUKSEN AIKATAULU JA TYÖMÄÄRÄT

Tarkempi kuvaus testausvaiheista löytyy luvusta 5.

Testaukseen ja korjaukseen tarvittavat työmäärät ovat suoraan yhteydessä löydettyjen virheiden määrään ja tehtyyn dokumentointiin. Virheitä on eri asteisia vakavuudeltaan ja vaikeudeltaan, joten joidenkin virheiden korjaus voi tapahtua parissa minuutissa, kun taas toisten virheiden paikannus ja korjaus voi viedä päiviä. Testaukseen tarvittavassa ajassa on hyvä ottaa huomioon myös testitapausten keksimiseen ja dokumenttien kirjoittamiseen tarvittava aika.

On määritelty, että testaus aloitetaan mahdollisesti jo ennen pääsiäislomaa ja *testaus päättyy viimeistään **18.4.2008***. Koko testaukseen on varattu aikaa siis maksimissaan noin kuukausi. Testaus aloitetaan mielellään heti, kun toteutettavat osat ovat valmiita testaukseen.

Moduulitestaukseen on varattu aikaa kaksi viikkoa ja se sattuu pahasti päällekkäin pääsiäisloman kanssa. Ihmisiä ei voida pakottaa testaamaan loman aikana, mutta loman jälkeen testaus on saatava nopeasti alkuun.

Ensimmäiseksi testataan tietokantamoduuli. Moduulin testaus on jaettu kolmeen vaiheeseen: koodin katselmointiin, kaikkien metodien testaukseen ja ainakin yhteen regressiotestauskierrokseen. Välissä tehdään tietenkin tarvittavat korjaukset. On otettava huomioon testitapausten kehittämiseen ja dokumentointiin menevä aika. Tietokantamoduuli on nopeimmillaan testattu kahdessa työpäivässä, mutta pahimmillaan aikaa voi mennä kaksikin viikkoa.

Integrointitestausta varten on varattu korkeintaan viikko, jonka aikana tietokanta ja käyttöliittymä laitetaan toimimaan yhdessä. Korjaukset ohjelmaan on tehtävä saman viikon aikana. Mikäli vakavia virheitä ei juurikaan löydy kannattaa siirtyä järjestelmätestaukseen mahdollisimman nopeasti.

Järjestelmätestaukseen varattu aika riippuu edellisten vaiheiden valmiiksi saamisesta. Maksimissaan aikaa voi odottaa olevan kaksi viikkoa. Toteutus ei saa olla liian hidasta, jotta järjestelmätestaukseen jää aikaa. Järjestelmätestaukseen päästään vasta kun ohjelmasta on olemassa toimiva ja asennettava versio. Testitapausten määrältä järjestelmätestaus on laajin ja ei ole suoritettavissa yhdeltä istumalta. Testaus kannattaa jakaa osiin ja hajauttaa projektiryhmän jäsenille. Järjestelmätestaukseen osallistuvat myös asiakas ja opponoiva ryhmä, joten aikataulu tulee tehdä myös heille selväksi. Ainakin yksi ryhmän jäsen keskittyy virheiden paikannukseen ja korjausten tekemiseen samalla kun muut testaavat.

Käyttäjätestaus suunnitellaan ja toteutetaan viimeiseksi, jos se koetaan tarpeelliseksi, ja siihen on jäänyt riittävästi aikaa. Yksi ryhmän jäsen riittää käyttäjätestauksen valmisteluun. Käyttäjätestausta

varten testihenkilöille voidaan tehdä valmis tehtävälista testauksen helpottamiseksi. Käyttäjätestaus tehdään ja dokumentoidaan yhden päivän aikana.

Testausraporttia laaditaan testauksen rinnalla. Koko testaus päättyy testausraportin valmistumiseen. Testausraportin laatimiseen menee aikaa vähintään muutama työpäivä. Testausraporttiin tulee yhteenveto testauksen tapahtumista ja järjestelmän korjauksista. Testausraportin laatija voi olla sama henkilö kuin tämän testaussuunnitelman kirjoittaja. Testausraportinkin olisi hyvä olla valmis määriteltyyn testauksen päättymispäivään mennessä.

7 TESTITAPAUKSET

Tässä luvussa on kerrottu testauksessa käytettävät testitapaukset. Testitapaukset on yksilöity ja priorisoitu. Testitapauksissa kuvataan, mitä tehdään ja mitkä ovat odotetut tulokset.

7.1 Yleistä

Kirjatut testitapaukset on laitettu loogiseen järjestykseen, mutta testaaja voi halutessaan jättää joitain testitapauksia suorittamatta tai tehdä testit eri järjestyksessä. Testitapaukset on tarkoitettu testauksen apuvälineiksi, eikä niiden orjallinen noudattaminen ole itseistarkoitus. Erityisesti käyttöliittymän testauksessa ja regressiotestauksessa voi testaaja käyttää omaa luovaa ajattelukykyään. Testitapauksia voidaan myös keksiä lisää testauksen aikana. Jos testitapausta ei voida suorittaa, kirjataan testipöytäkirjaan siihen syy. Jotkut testitapaukset voidaan joutua kokonaan hylkäämään tarpeettomina ohjelman rakenteen kehittyessä.

Testitapausten tunnukset ovat muotoa:

TC<testitapausluokan nro>.<vaatimuksen nro>-<testitapauksen nro>

Esimerkiksi profiilin luonnin ensimmäinen testitapaus saisi tunnuksekseen TC2.1-1 ja uusi testitapaus merkintöjen lisäysten väliin TC3.7-3b.

Testitapausluokkien numerointi on seuraava:

1. Tietokantamoduuli
2. Profiilitoiminnot
3. Merkintöjen lisäys
4. Merkintöjen selaus ja muokkaus
5. Tyyppien ja attribuuttien hallinta
6. Raportit
7. Käyttöliittymä
8. Asennus, poisto ja siirrettävyys
9. Muut ei-toiminnalliset ominaisuudet
10. Erikoistapaukset

Vaatimusnumerot ovat vaatimusmäärittelyssä olevia numeroita, mikäli testitapaus liittyy johonkin vaatimukseen. Vaatimuksiin liittymättömät testitapaukset saavat numeron 0. Liitteessä 2 on lista vaatimusmäärittelyssä olevista vaatimuksista.

Testitapauksen numero on juokseva. Mikäli uusi testitapaus tulee väliin, tulee testitapauksen numeron jälkeen pieni aakkonen alkaen b:stä.

Testitapaukset on luokiteltu kolmeen kriittisyysluokkaan. Numeroinnissa 1 tarkoittaa korkeinta ja 3 matalinta. Ensimmäisen kriittisyysluokan testitapausten on mentävä läpi, jotta ohjelma voidaan hyväksyä. Toisen luokan testitapaukset on hyvä suorittaa ja niistä on hyvä saada oikea tulos. Kolmannen luokan testitapauksista osan voi jättää kokonaan suorittamatta, eivätkä niistä löydetyt virheet ole yleensä kriittisiä.

7.2 Tietokantamoduuli

Tietokantamoduulin testitapauksia ei kirjoiteta tähän dokumenttiin, vaan niistä tehdään erillinen dokumentti. Testaaja itse määrittää tietokannan testitapaukset. Monet eri alilukujen alla olevat testitapaukset kelpaavat myös tietokannan testaukseen. Jokaiselle tietokannan metodille on annettava syötteenä sekä oikeita, että vääriä arvoja. Tärkeimpien metodien syötteiden raja-arvot on myös testattava.

7.3 Profiilitoiminnot

Tunnus	TC2.1-1	**Prioriteetti**	1
Kuvaus	Luodaan profiili pelkällä käyttäjätunnuksella ja muut kentät jätetään oletusarvoikseen		
Odotettu tulos	Profiili ilmestyy listaan		

Tunnus	TC2.1-2	**Prioriteetti**	1
Kuvaus	Luodaan profiili, jolla sekä käyttäjätunnus että salasana		
Odotettu tulos	Profiili ilmestyy listaan		

Tunnus	TC2.1-3	**Prioriteetti**	2
Kuvaus	Luodaan profiili, jonka käyttäjätunnus on liian lyhyt		
Odotettu tulos	Profiilia ei luoda ja ohjelma ilmoittaa käyttäjälle käyttäjätunnuksen olevan liian lyhyt.		

Tunnus	TC2.1-4	**Prioriteetti**	2
Kuvaus	Luodaan profiili, jonka käyttäjätunnus on liian pitkä		
Odotettu tulos	Profiilia ei luoda ja ohjelma ilmoittaa käyttäjälle käyttäjätunnuksen olevan liian pitkä.		

Tunnus	TC2.1-5	**Prioriteetti**	2
Kuvaus	Luodaan profiili, jonka salasanat eivät täsmää. Laitetaan salasanat täsmäämään		
Odotettu tulos	Ohjelma ilmoittaa käyttäjälle, että salasanat eivät täsmää ja pyytää käyttämään syöttämään salasanat uudestaan. Profiili ilmestyy listaan.		

Tunnus	TC2.1-6	**Prioriteetti**	2
Kuvaus	Yritetään luoda profiili, jonka niminen profiili jo löytyy		
Odotettu tulos	Ilmoitetaan käyttäjälle, että samanniminen profiili on jo olemassa. Kysytään käyttäjältä, haluaako hän tallentaa profiilin olemassa olevan profiilin päälle. Tallennus päälle onnistuu.		

Tunnus	TC2.2-1	**Prioriteetti**	1
Kuvaus	Kirjaudutaan ensimmäiseksi luodulla profiililla, jolla ei ole salasanaa		
Odotettu tulos	Ohjelma käynnistyy etusivulle.		

Tunnus	TC2.2-2	**Prioriteetti**	1
Kuvaus	Kirjaudutaan ulos ohjelmasta kirjautumisikkunaan		
Odotettu tulos	Ohjelma sulkeutuu ja kirjautumisikkuna ilmestyy.		

Tunnus	TC2.2-3	**Prioriteetti**	1
Kuvaus	Yritetään kirjautua profiiliin, jolla on salasana, väärällä salasanalla		
Odotettu tulos	Kirjautuminen epäonnistuu ja ohjelma ilmoittaa väärästä salasanasta.		

Tunnus	TC2.2-4	**Prioriteetti**	1
Kuvaus	Kirjaudutaan profiiliin, jolla on salasana		
Odotettu tulos	Ohjelma käynnistyy etusivulle.		

Tunnus	TC2.3-1	**Prioriteetti**	1
Kuvaus	Käynnistetään profiilin asetusten muokkaus		
Odotettu tulos	Asetusikkuna avautuu erilliseen ikkunaan		

Tunnus	TC2.3-2	**Prioriteetti**	2
Kuvaus	Muutetaan profiilin tallennettuja muita tietoja. Tallennetaan muutokset.		
Odotettu tulos	Tiedot tallentuvat ja säilyvät		

Tunnus	TC2.3-3	**Prioriteetti**	2
Kuvaus	Yritetään muuttaa profiilin salasana väärin		
Odotettu tulos	Ohjelma ilmoittaa väärin syötetystä salasanasta		

Tunnus	TC2.3-4	**Prioriteetti**	2
Kuvaus	Muutetaan profiilin salasana oikein. Tallennetaan muutokset.		
Odotettu tulos	Tietokannan salasana muuttuu. Muutos voi kestää.		

Tunnus	TC2.3-5	**Prioriteetti**	3
Kuvaus	Muutetaan käyttöliittymää. Jos muutokset eivät tapahdu välittömästi kirjaudutaan ulos ja takaisin sisään		
Odotettu tulos	Profiiliin tehdyt muutokset kuvastuvat käyttöliittymän ulkoasuun.		

Tunnus	TC2.6-1	**Prioriteetti**	2
Kuvaus	Yritetään poistaa profiili väärällä salasanalla oman profiilin poistotoiminnolla.		
Odotettu tulos	Ohjelma ilmoittaa väärästä salasanasta ja toiminto päättyy		

Tunnus	TC2.6-2	**Prioriteetti**	2
Kuvaus	Poistetaan profiili, johon ollaan kirjauduttu.		
Odotettu tulos	Profiiliin liittyvät tiedostot häviävät levyltä. Ohjelma palaa takaisin kirjautumisikkunaan. Poistettu profiili ei näy listassa.		

Tunnus	TC2.4-1	**Prioriteetti**	1
Kuvaus	Kirjaudutaan sisään ohjelmaan. Viedään profiili tiedostoon vientitoiminnolla. Tallennetaan profiiliin tietoja. Kirjaudutaan ulos ohjelmasta.		
Odotettu tulos	Syntyy odotusten kaltainen tiedosto haluttuun paikkaan.		

Tunnus	TC2.4-2	**Prioriteetti**	2
Kuvaus	Yritetään tuoda profiilia tiedostosta, jolla ei ole oikeanlainen tiedostopääte		
Odotettu tulos	Ei pitäisi olla edes mahdollista yrittää		

Tunnus	TC2.4-3	**Prioriteetti**	3
Kuvaus	Yritetään tuoda profiilia tiedostosta, joka on oikeanlainen tiedostopääte, mutta ei ole Kuntokirjuri-ohjelman tiedosto.		
Odotettu tulos	Ohjelma joko tunnistaa vääräntyyppisen tiedoston ja ilmoittaa siitä tai sitten tiedosto purkautuu ohjelman kansioon oikein. Profiililistaan ei saa mielellään ilmestyä mitään kummallista.		

Tunnus	TC2.4-4	**Prioriteetti**	2
Kuvaus	Yritetään tuoda viety profiili tiedostosta. Profiili on jo olemassa. Jos tuonti epäonnistuu, poistetaan olemassa oleva profiili.		
Odotettu tulos	Ohjelma ilmoittaa profiilin olevan jo olemassa. Ohjelma kysyy kirjoitetaanko päälle. Päällekirjoitus onnistuu ohjelmakansioon.		

Tunnus	TC2.4-5	**Prioriteetti**	1
Kuvaus	Tuodaan viety profiili tiedostosta, jos edellinen tuonti epäonnistui.		
Odotettu tulos	Profiilin tiedostot ilmestyvät ohjelman kansioon. Profiili ilmestyy listaan.		

Tunnus	TC2.2-5	**Prioriteetti**	1
Kuvaus	Kirjaudutaan tuotuun profiiliin.		
Odotettu tulos	Kirjautuminen onnistuu samalla tavoin kuin luodussa profiilissa.		

Tunnus	TC2.0-1	**Prioriteetti**	2
Kuvaus	Poistetaan ylimääräisiä profiileja käsin projektikansiosta		
Odotettu tulos	Profiilien tiedot eivät näy enää kirjautumisikkunan listassa.		

Tunnus	TC2.5-1	**Prioriteetti**	1
Kuvaus	Kirjaudutaan ulos ohjelmasta, niin että ohjelma sulkeutuu		
Odotettu tulos	Ohjelma sulkeutuu, eikä sen tuottamia prosesseja jää pyörimään taustalle.		

7.4 Merkintöjen lisäys

Tunnus	TC3.7-1	**Prioriteetti**	1
Kuvaus	Lisätään oikein muotoiltuja merkintöjä tietokantaan käyttäen merkintöjenlisäystoimintoa. Valmiita merkintöjä, joita voidaan syöttää on kirjoitettu tiedostoon.		
Odotettu tulos	Merkintöjen lisäyksessä ei tapahdu ongelmia. Ohjelma ilmoittaa merkinnän lisäyksen onnistumisesta huomaamatta.		

Tunnus	TC3.7-2	**Prioriteetti**	2
Kuvaus	Yritetään lisätä tietokantaan väärin muotoiltuja merkintöjä. Yritetään lisätä liian pitkiä merkintöjä tai esimerkiksi merkkijonoja luvuiksi.		
Odotettu tulos	Ohjelma ilmoittaa väärin muotoilluista merkinnöistä eikä niiden lisäys onnistu.		

Tunnus	TC3.7-3	**Prioriteetti**	3
Kuvaus	Yritetään lisätä tietokantaan samoja tietoja		
Odotettu tulos	Ei pitäisi olla edes mahdollista		

Tunnus	TC3.7-4	**Prioriteetti**	1
Kuvaus	Lisätään merkintöjä omille tyypeille. Tietokannassa on oltava omia tyyppejä tämän tekemiseksi.		
Odotettu tulos	Lisäysten ei pitäisi poiketa muista lisäyksistä.		

Tunnus	TC3.8-1	**Prioriteetti**	2
Kuvaus	Lisätään tietokantaan tavoitteita. Tarkastetaan näkyvätkö tavoitteet lisätyissä tiedoissa ja käyttääkö ohjelma tavoitemerkintöjä oikein.		
Odotettu tulos	Tavoitteiden lisäys onnistuu. Merkinnöistä erottaa onko kyseessä tavoite vai ei. Ohjelma käyttää tavoitemerkintöjä oikein.		

7.5 Merkintöjen selaus ja muokkaus

Tunnus	TC4.10-1	**Prioriteetti**	1
Kuvaus	Haetaan tietokannasta yhdelle päivälle tehdyt merkinnät		
Odotettu tulos	Merkinnät ilmestyvät listaan oikein.		

Tunnus	TC4.10-2	**Prioriteetti**	1
Kuvaus	Haetaan tietokannasta viikon merkinnät		
Odotettu tulos	Merkinnät ilmestyvät listaan oikein aikajärjestyksessä.		

Tunnus	TC4.10-3	**Prioriteetti**	2
Kuvaus	Haetaan tietokannasta kaikki merkinnät		
Odotettu tulos	Kaikki merkinnät ilmestyvät listaan aikajärjestyksessä.		

Tunnus	TC4.10-4	**Prioriteetti**	1
Kuvaus	Suodatetaan viikon merkinnöistä vain tietyn tyyppiset		
Odotettu tulos	Muut merkinnät katoavat näkyvistä. Merkinnät näkyvät oikein aikajärjestyksessä.		

Tunnus	TC4.10-5	**Prioriteetti**	2
Kuvaus	Vaihdetaan aikaväliä suodatuksen ollessa päällä		
Odotettu tulos	Suodatusasetukset eivät muutu. Aikavälin merkinnät näkyvät oikein.		

Tunnus	TC4.11-1	**Prioriteetti**	1
Kuvaus	Muokataan suodatettuja merkintöjä oikein.		
Odotettu tulos	Muokkauksen päätyttyä tiedot tallentuvat tietokantaan ja merkinnän muutokset tulevat voimaan myös käyttöliittymään.		

Tunnus	TC4.11-2	**Prioriteetti**	2
Kuvaus	Yritetään muuttaa merkinnän tietoja väärin.		
Odotettu tulos	Ohjelma ilmoittaa virheistä tietojen muutossa. Ohjelma korostaa vir-		

	heelliset kentät.

Tunnus	TC4.11-3	**Prioriteetti**	3
Kuvaus	Muutetaan jo muutettuja merkintöjä.		
Odotettu tulos	Tuloksen ei pitäisi erota edellisistä muokkauksista.		

Tunnus	TC4.11-4	**Prioriteetti**	1
Kuvaus	Poistetaan merkintöjä yksitellen.		
Odotettu tulos	Poistetut merkinnät myös poistuvat eivätkä jää näkymään käyttöliittymään.		

Tunnus	TC4.11-5	**Prioriteetti**	2
Kuvaus	Poistetaan useampi merkintä kerralla		
Odotettu tulos	Sama tulos kuin yksittäisiä merkintöjä poistettaessa.		

7.6 Tyyppien ja attribuuttien hallinta

Tunnus	TC5.14-1	**Prioriteetti**	1
Kuvaus	Lisätään uusi tyyppi, jolle asetetaan kolme valmista attribuuttia		
Odotettu tulos	Uuden tyypin luonti onnistuu. Käyttäjälle ilmoitetaan huomaamattomasti.		

Tunnus	TC5.14-2	**Prioriteetti**	2
Kuvaus	Yritetään lisätä tyyppejä, joiden nimet ovat liian pitkiä tai muuten virheellisiä		
Odotettu tulos	Ohjelma tarkastaa nimien virheellisyyden ja ilmoittaa virheestä.		

Tunnus	TC5.14-3	**Prioriteetti**	2
Kuvaus	Lisätään tyyppi, jolla ei ole lainkaan attribuutteja		
Odotettu tulos	Tyypin lisäyksessä ei ole ongelmia.		

Tunnus	TC5.14-4	**Prioriteetti**	2
Kuvaus	Yritetään lisätä tyyppi, joka on jo olemassa		
Odotettu tulos	Ohjelma ilmoittaa tyypin jo löytyvän. Lisäys ei onnistu.		

Tunnus	TC5.14-5	**Prioriteetti**	1
Kuvaus	Lisätään uusia attribuutteja ainakin kolme		
Odotettu tulos	Attribuuttien lisäys onnistuu		

Tunnus	TC5.14-6	**Prioriteetti**	2
Kuvaus	Yritetään lisätä attribuutteja, joiden nimet ovat liian pitkiä tai muuten virheellisiä		
Odotettu tulos	Ohjelma tarkastaa nimien virheellisyyden ja ilmoittaa käyttäjälle.		

Tunnus	TC5.14-7	**Prioriteetti**	2
Kuvaus	Yritetään lisätä attribuutti, joka on jo olemassa		
Odotettu tulos	Ohjelma ilmoittaa attribuutin jo löytyvän. Lisäys ei onnistu.		

Tunnus	TC5.14-8	**Prioriteetti**	2
Kuvaus	Lisätään uusi tyyppi, johon liittyy luotuja attribuutteja		
Odotettu tulos	Ei pitäisi olla ongelmaa		

Tunnus	TC5.15-1	**Prioriteetti**	1
Kuvaus	Poistetaan itse luotu tyyppi, johon ei liity merkintöjä, eli yksinkertaisin perustapaus		
Odotettu tulos	Ohjelma tarkastaa löytyykö tyypille merkintöjä. Varmistus kysytään joka tapauksessa. Tyypin poisto onnistuu.		

Tunnus	TC5.15-2	**Prioriteetti**	1
Kuvaus	Poistetaan valmis tyyppi, johon ei liity merkintöjä.		
Odotettu tulos	Ei poikkea itse luodun tyypin poistosta.		

Tunnus	TC5.15-3	**Prioriteetti**	1
Kuvaus	Yritetään poistaa itse luotu tyyppi, johon liittyy merkintöjä. Merkintöjä on siis lisättävä ennen testitapauksen ajoa.		
Odotettu tulos	Ohjelma tarkastaa löytyykö tyypille merkintöjä. Varmistus kysytään joka tapauksessa. Tyypin poisto onnistuu.		

Tunnus	TC5.15-4	**Prioriteetti**	1
Kuvaus	Yritetään poistaa valmis tyyppi, johon liittyy merkintöjä. Merkintöjä on siis lisättävä ennen testitapauksen ajoa.		
Odotettu tulos	Ei poikkea itse luodun tyypin poistosta.		

Tunnus	TC5.15-5	**Prioriteetti**	1
Kuvaus	Poistetaan itse luotu attribuutti, joka ei liity tyyppeihin, eli yksinkertainen perustapaus.		
Odotettu tulos	Poistosta kysytään varmistusta. Poisto onnistuu.		

Tunnus	TC5.15-6	**Prioriteetti**	3
Kuvaus	Poistetaan valmis attribuutti, joka ei liity tyyppeihin. Ei välttämättä edes löydy sellaista.		
Odotettu tulos	Ei poikkea itse luodun attribuutin poistosta.		

Tunnus	TC5.15-7	**Prioriteetti**	1
Kuvaus	Yritetään poistaa itse luotu attribuutti, joka liittyy itse luotuun tyyppiin		
Odotettu tulos	Annetaan käyttäjälle virheilmoitus. Poisto ei onnistu ennen kuin tyypit on poistettu.		

Tunnus	TC5.15-8	**Prioriteetti**	1
Kuvaus	Yritetään poistaa valmis attribuutti, joka liittyy yhteen tai useampaan tyyppiin		
Odotettu tulos	Ei poikkea itse luodun attribuutin poistosta.		

7.7 Raportit

Tunnus	TC6.12-1	**Prioriteetti**	1

Kuvaus	Luodaan raportti yhden valmiin tyypin merkinnöistä, ei kuvaajaa tai muuta. Merkintöjä löytyy. Yksinkertainen perustapaus.
Odotettu tulos	Raportti näyttää siltä kuin pitää.

Tunnus	TC6.12-2	**Prioriteetti**	1
Kuvaus	Luodaan raportti yhden valmiin tyypin merkinnöistä, myös kuvaaja. Merkintöjä löytyy riittävästi kuvaajan tekemiseen.		
Odotettu tulos	Kuvaaja näyttää oikeanlaiselta. Myös mitta-asteikot ovat oikein.		

Tunnus	TC6.12-3	**Prioriteetti**	1
Kuvaus	Luodaan raportti useamman tyypin merkinnöistä, ei kuvaajia. Tyypeille löytyy merkintöjä. Tarkastellaan, miten useampi tyyppi vaikuttaa.		
Odotettu tulos	Raportti näyttää hyvältä myös, kun on useampi tyyppi.		

Tunnus	TC6.12-4	**Prioriteetti**	1
Kuvaus	Luodaan raportti useamman tyypin merkinnöistä, myös kuvaajat. Tyypeille löytyy merkintöjä. Tarkastellaan ulkoasua.		
Odotettu tulos	Raportin asettelu toimii ja raportti on luettavissa.		

Tunnus	TC6.12-5	**Prioriteetti**	2
Kuvaus	Luodaan raportti yhdestä tyypistä, jolla ei ole merkintöjä. myös kuvaaja		
Odotettu tulos	Raportti luodaan, mutta kuvaajaa ei ilmesty.		

Tunnus	TC6.12-6	**Prioriteetti**	2
Kuvaus	Luodaan raportti, jossa ei yhtään tyyppiä.		
Odotettu tulos	Ohjelma huomauttaa, ettei yhtään tyyppiä ole valittu. Raporttia ei luoda.		

Tunnus	TC6.12-7	**Prioriteetti**	3
Kuvaus	Yritetään luoda raportti järjettömältä aikaväliltä, mikäli mahdollista.		
Odotettu tulos	Raporttia ei luoda.		

Tunnus	TC6.13-1	**Prioriteetti**	1
Kuvaus	Luodaan terveyskortti yleisimmillä asetuksilla. Tarkastetaan terveyskortin ulkoasu ja osien sijoittelu. Tarvittaessa luodaan useampia terveyskortteja eri asetuksilla.		
Odotettu tulos	Terveyskortista tulee suunnitellun kaltainen. Valitut kentät näkyvät.		

Tunnus	TC6.13-2	**Prioriteetti**	3
Kuvaus	Yritetään luoda terveyskortti, johon ei tule merkintöjä. Ei välttämättä edes mahdollista.		
Odotettu tulos	Ohjelma huomauttaa, ettei terveyskorttiin ole valittu merkintöjä. Terveyskorttia ei luoda.		

7.8 Käyttöliittymä

Tunnus	TC7.17-1	**Prioriteetti**	1
Kuvaus	Käyttöliittymän yleisilmeen ja värimaailman tarkastelu. Katsotaan, että värejä on käytetty johdonmukaisesti. Tarkastetaan myös käyttöliittymässä näkyvät kuvat ja kuvakkeet.		
Odotettu tulos	Käyttöliittymä on miellyttävä katsoa, eikä värimaailma satu silmiin.		

Tunnus	TC7.17-2	**Prioriteetti**	1
Kuvaus	Tarkastetaan painikkeiden sijainti ja koko. Tarkastetaan myös painikkeiden johdonmukainen ja järkevä nimeäminen.		
Odotettu tulos	Painikkeet ovat hyvin sijoiteltuja ja helposti löydettäviä. Painikkeiden nimet ovat kuvaavia.		

Tunnus	TC7.17-3	**Prioriteetti**	1
Kuvaus	Tarkastetaan muiden komponenttien sijainti ja koko.		
Odotettu tulos	Komponentit on sijoiteltu järkevästi, eikä turhaa tyhjää tilaa jää.		

Tunnus	TC7.20-1	**Prioriteetti**	1
Kuvaus	Tarkastetaan käyttöliittymän osissa käytetty kieli loppukäyttäjän näkökulmasta. Ohjeiden kieli tarkastetaan myöhemmin.		
Odotettu tulos	Käyttöliittymän osissa ei käytetä liian hankalia tai kyseenalaisia ilmauksia.		

Tunnus	TC7.20-2	**Prioriteetti**	1
Kuvaus	Tarkastetaan käyttäjälle annetuissa ilmoituksissa käytetty kieli.		
Odotettu tulos	Ilmoitukset ovat kieleltään selkeitä ja kertovat virheen syyn ymmärrettävästi.		

Tunnus	TC7.20-3	**Prioriteetti**	1
Kuvaus	Tarkastetaan käyttäjälle annettavissa ohjeissa käytetty kieli.		
Odotettu tulos	Ohjeistuksesta on apua käyttäjälle, eikä se aiheuta lisää sekaannusta.		

Tunnus	TC7.19-1	**Prioriteetti**	2
Kuvaus	Tarkastetaan ilmoitusten riittävyys. Katsotaan, mitä ilmoitetaan ja missä. Ilmoituksia tarkastetaan myös muita testejä tehtäessä, joten tämä tarkastelu ei ole välttämätön.		
Odotettu tulos	Käyttäjää informoidaan tarvittaessa, eikä näytetä liikaa ilmoituksia.		

Tunnus	TC7.19-2	**Prioriteetti**	2
Kuvaus	Tarkastetaan ohjeistuksen riittävyys. Katsotaan miten nopeasti ohje on löydettävissä ongelmatilanteessa.		
Odotettu tulos	Käyttäjä löytää nopeasti apua ongelmatilanteissa. Ohjeistus toimii.		

Tunnus	TC7.16-1	**Prioriteetti**	3
Kuvaus	Pienennetään ohjelma tilariville. Katsotaan onko ongemia.		
Odotettu tulos	Ei ongelmia.		

Tunnus	TC7.16-2	**Prioriteetti**	2
Kuvaus	Muutetaan jokaisen ikkunan kokoa, mikäli pystytään. Katsotaan, miten käyttöliittymä suhtautuu muutoksiin.		
Odotettu tulos	Ikkunan koon muutos ei laita käyttöliittymää sekaisin tai ikkunan kokoa ei edes pystytä muuttamaan.		

Tunnus	TC7.16-3	**Prioriteetti**	2

Kuvaus	Kokeillaan ohjelmaan määritellyt pikanäppäimet.
Odotettu tulos	Pikanäppäimet toimivat.

Tunnus	TC7.16-4	**Prioriteetti**	3
Kuvaus	Kokeillaan muita näppäinkomentoja, joita ei ole määritelty.		
Odotettu tulos	Ohjelma ei mene sekaisin muista näppäinkomennoista.		

Tunnus	TC7.16-5	**Prioriteetti**	2
Kuvaus	Kokeillaan käyttöliittymän käyttöä näppäimistöllä. Liikutaan lomakkeissa tabulaattorin avulla ja kokeillaan mitä Enter- ja Esc-näppäimet tekevät.		
Odotettu tulos	Käyttöliittymässä liikkuminen näppäimistöllä on luontevaa ja loogista.		

Tunnus	TC7.16-6	**Prioriteetti**	1
Kuvaus	Tarkastetaan käyttöliittymän loogisuus loppukäyttäjän näkökulmasta. Testaajalle tämä voi olla vaikeaa.		
Odotettu tulos	Käyttöliittymä toimii loogisesti.		

Tunnus	TC7.16-7	**Prioriteetti**	1
Kuvaus	Tarkastetaan tärkeimpien toimintojen suorituksen helppous loppukäyttäjän näkökulmasta, eli tarkastetaan käytettävyys. Testaaja etsii mahdollisesti hankalia asioita eläytymällä loppukäyttäjäksi.		
Odotettu tulos	Tärkeimmät toiminnot ovat selkeästi esillä ja helposti käytettäviä.		

Tunnus	TC7.19-3	**Prioriteetti**	2
Kuvaus	Katsotaan voiko käyttäjä jäädä jumiin johonkin. Tarkastetaan onko ohjeistus riittävää ja voiko jokin tilanne vaikuttaa hankalalta loppukäyttäjän näkökulmasta.		
Odotettu tulos	Ohjelmassa ei ole tilanteita, joissa käyttäjä ei pääse etenemään.		

7.9 Asennus, poisto ja siirrettävyys

Tunnus	TC8.0-1	**Prioriteetti**	2

Kuvaus	Asennetaan ohjelma Windowsiin. Katsotaan syntyykö kansio oikeaan paikkaan, syntyykö pikakuvakkeita, ilmestyykö lisää/poista-listaan.
Odotettu tulos	Saadaan asennettua ohjelma Windowsiin normaaliin tapaan.

Tunnus	TC8.0-2	**Prioriteetti**	2
Kuvaus	Poistetaan ohjelma Windowsista. Ei poisteta profiilitiedostoja, jos siihen annetaan mahdollisuus. Tarkastetaan jääkö tiedostoja tai pikakuvakkeita.		
Odotettu tulos	Ohjelma poistuu, eikä siihen jää viittauksia. Profiilitiedostot poistuvat kysymättä tai niiden poistoa kysytään.		

Tunnus	TC8.22-1	**Prioriteetti**	3
Kuvaus	Yritetään asentaa ohjelma Linuxiin pääkäyttäjän oikeuksilla. Tarkastetaan, mihin asennus tapahtuu. Jos asennusta ei tueta, koitetaan muuten ajaa ohjelmaa Linuxissa. Katsotaan toimiiko ohjelma ja mihin tiedostot tallentuvat. Testaillaan muutenkin käyttöä Linux-ympäristössä.		
Odotettu tulos	Asennus Linuxiin onnistuu ja ohjelmaan syntyy pikakuvake valikkoon.		

Tunnus	TC8.22-2	**Prioriteetti**	3
Kuvaus	Poistetaan ohjelma Linuxista toiminnon kautta tai käsin. Tarkastetaan jääkö tiedostoja.		
Odotettu tulos	Tiedostoja ei jää.		

Tunnus	TC8.0-3	**Prioriteetti**	3
Kuvaus	Yritetään asentaa ohjelma eri käyttöjärjestelmiin, vaikka se on jo asennettu.		
Odotettu tulos	Tulee kehote poistaa edellinen asennus tai asennetaan päälle. Profiilitiedostoja ei poisteta päälle asennettaessa.		

7.10 Muut ei-toiminnalliset ominaisuudet

Tunnus	TC9.21-1	**Prioriteetti**	1
Kuvaus	Testataan ohjelman suorituskykyä tavallisessa käytössä. Tehdään aikaa vieviä toimintoja ja mitataan, kauanko niihin menee. Testataan sekä nopeilla, että hitailla koneilla.		
Odotettu tulos	Käyttöliittymä on riittävän nopea hitaammillakin koneilla.		

Tunnus	TC9.21-2	**Prioriteetti**	2
Kuvaus	Tarkastellaan ohjelman muistin käyttöä eri tilanteissa. Muistin käyttöä voi tarkastella monenlaisilla ohjelmilla.		
Odotettu tulos	Muistin käyttö ei nouse hälyttävän korkeaksi.		

7.11 Erikoistapaukset

Tunnus	TC10.0-1	**Prioriteetti**	3
Kuvaus	Ohjelman jättäminen päälle pyörimään kahdeksi vuorokaudeksi. Katsotaan tuleeko ongelmia.		
Odotettu tulos	Ongelmia ei tule.		

Tunnus	TC10.0-2	**Prioriteetti**	3
Kuvaus	Yritetään käynnistää ohjelmaa ilman virtuaaliympäristöä.		
Odotettu tulos	Ajo ei onnistu		

Tunnus	TC10.0-3	**Prioriteetti**	2
Kuvaus	Tapetaan ohjelma kirjautumatta ulos. Käynnistetään ohjelma uudestaan. Katsotaan vaikuttiko ohjelman tappaminen tietokantaan ja siihen tallennettuihin tietoihin.		
Odotettu tulos	Viimeisimmät tiedot saattavat jäädä tallentumatta tietokantaan.		

Tunnus	TC10.0-4	**Prioriteetti**	3
Kuvaus	Apinatestaus, eli toimitaan käyttöliittymässä ilman mitään tolkkua. Tukeva humalatila voi auttaa. Katsotaan kaatuuko ensin ohjelma vai mies.		
Odotettu tulos	Mitä tahansa voi sattua.		

Tunnus	TC10.0-5	**Prioriteetti**	2
Kuvaus	Gorillatestaus, eli tehdään kaikki mahdollinen ohjelman kaatamiseksi. Syötetään virheellisiä syötteitä, mikä keritään ja pahoinpidellään ohjelmaa. Katsotaan saadaanko ohjelma kaatumaan.		
Odotettu tulos	Ohjelma saadaan kaatumaan tarpeeksi yrittämällä. Kaataminen vaatii kuitenkin jonkin verran työtä.		

8 OMINAISUUDET JOITA EI TESTATA

Valmiit avoimen lähdekoodin komponentit

Käytetyt komponentit ovat jo todennäköisesti testattuja. Ainoa mitä pitää testata, on käyttääkö ohjelmamme niitä oikein.

Ohjelman hyödyllisyys

Hyödyllisyyttä ja tarpeellisuutta voidaan kysyä projektin ulkopuolisilta henkilöiltä, mutta mitään tutkimusta ei tehdä.

Ohjelman toiminta rasituksessa

Rasitukselle ei tehdä erillistä testausta. Ei myöskään testata suuria tietomääriä.

Ohjelman toiminta vanhemmissa ympäristöissä

Ohjelmaa ei testata Windows XP:tä vanhemmissa ympäristöissä tai vanhoilla koneilla. Myöskään erikoisia ympäristöjä ei testata. (Solaris jne.)

Levytilan tai muistin loppumisen vaikutukset

Turvallisuus

Tietokannan turvallisuutta ei testata yrittämällä murtautua siihen ulkopuolisilla ohjelmilla. Myöskään tietoturvan riittävyyttä ei testata.

Toipuminen

Toipumista ei testata kattavasti. Toipuminen tarkoittaa ohjelman kykyä selviytyä väkivaltaisesta sulkemisesta. Ongelmia voi lähinnä syntyä tietokannan eheydessä. Joitain tietoja saattaa jäädä tallentumatta levylle. Tietojen tallennusta voidaan kuitenkin jossain määrin testata.

Useampi ohjelman ilmenemä

Ei testata mitä ongelmia useampi ohjelman ajaminen aiheuttaa, mutta testataan kuitenkin onko se mahdollista.

Liian pitkät syötteet

Liian pitkiä syötteitä testataan tietokantatestauksessa, mutta käyttöliittymäpuolella ei niihin kiinnitetä enää juurikaan huomiota.

Testausprosessin laatu

Testausprosessin laatua ei arvioida enempää kuin testausraportin kirjoittaminen vaatii.

Syntyneiden dokumenttien laatu

Testauksessa syntyneistä dokumenteista tarkistetaan, että niistä löytyvät vaaditut tiedot, mutta dokumentin laatua ei arvioida systemaattisesti.

9 TESTAUKSEN HALLINTA

9.1 Testauksen hyväksyminen

Testaaja päättää itse milloin yksittäinen testitapaus on mennyt hyväksyttävästi läpi. Testauksen hyväksymistä ei hallita, mutta testausvastaava voi päättää milloin testaus on hylätty. Testaus voidaan hyväksyä esimerkiksi silloin, kun kriittisimmät testitapaukset ovat menneet läpi.

Alla on joitain yleisiä testausvaiheiden hyväksymiskriteerejä.

Tietokantatestaus on hyväksytty, kun tietokanta on testattu kahteen kertaan ja korjaukset on tehty.

Ennen integrointitestausta projektiryhmä hyväksyy käyttöliittymän ja toteaa sen olevan valmis testaukseen. Käyttöliittymän on oltava realistinen ja toimiva. Paljon on kiinni käyttöliittymän valmistumisen ajankohdasta. Seuraavaan vaiheeseen ei voida siirtyä ennen käyttöliittymän hyväksyntää.

Integroimistestaus on hyväksytty, kun tietokantakomentojen kutsu käyttöliittymästä ei aiheuta virheitä ja käyttöliittymä osaa käsitellä oikein tietokannan palautukset. Integroimistestauksen hyväksyntään kuuluu myös se että tietokannan tiedoista saadaan luotua luettava raportti.

Järjestelmätestaus on hyväksytty, kun aiotut testitapaukset on ajettu ja löytyneet virheet raportoitu ja vakavat virheet korjattu.

Ohjelman lopullisesta hyväksymisestä päättävät projektipäällikkö ja asiakas. Ohjelman hyväksyminen ei vaikuta projektin loppuun viemiseen, mutta tavoitteena on valmis ja julkaisukelpoinen ohjelma. Hyväksyttävän ohjelman on täytettävä vaatimuksissa esitetyt ehdot. Hyväksymistestausta varten voidaan valmistaa testitapauslista, jossa jokainen vaatimus testataan. Ohjelma voidaan hyväksyä listan mennessä hyväksyttävästi läpi.

Koko testaus hyväksytään, kun kaikki vaadittavat dokumentit on kirjoitettu. Muita rajoja testauksen hyväksymiselle ei aseteta.

9.2 Testauksen hylkääminen

Huonosti suoritettu testaus voidaan hylätä ja laittaa testaaja suorittamaan testaus uudelleen. Testaus hylätään esimerkiksi silloin, kun testaaja ei kirjaa ylös testauksen tapahtumia. Hylkäys voi johtua myös riittämättömästä testauksesta. Testauksen hylkäämisestä päättää testausvastaava.

Testauksen lisäksi voidaan hylätä myös yksittäisiä testitapauksia. Testitapauksen hylkääminen on paikallaan esimerkiksi silloin, kun sitä ei voida suorittaa ohjelman muuttuneen toiminnan vuoksi. Testipäiväkirjaan kirjataan myös hylätyt testitapaukset ja selitetään miksi niitä ei ole suoritettu.

9.3 Testauksen keskeyttäminen

Keskeyttäminen ei ole sama asia kuin lopettaminen. Keskeytyksestä voidaan puhua, jos testausta ei saada vietyä loppuun kerralla ja sitä on tarkoitus jatkaa myöhemmin. Testaus voidaan joutua keskeyttämään myös, korjausten tekemisen ajaksi. Testauspöytäkirjaan kirjataan ilmoitus keskeytyksestä. Kun testausta jatketaan, ilmoitetaan siitä samassa pöytäkirjassa. Testauksen jatkaminen edellyttää, että ohjelman tila saadaan palautettua ja ympäristössä ei tapahdu muutoksia keskeytyksen aikana. Testitapaukset voivat vaatia ennalta tapahtuvia asioita.

9.4 Testauksen lopettaminen

Testaus voidaan lopettaa monesta syystä. Testausvastaava tai testaaja itse määrittää milloin testaus voidaan lopettaa hyväksytysti. Testaus voidaan lopettaa esimerkiksi, kun ollaan saatu ajettua kaikki aiotut testitapaukset ilman ongelmia.

Eräs lopetuksen syy on, että ohjelma tai sen osa ei ole vielä valmis testaukseen ja vaatii korjausta. Löytyy siis sellainen virhe, joka estää testauksen jatkamisen. Toinen syy testauksen lopettamiseen on ajan loppuminen. Testausvastaava määrittää päivämäärän, johon mennessä testaus on viimeistään lopetettava.

Testauksen loputtua testaaja toimittaa testauspöytäkirjan ja virheraportit sovittuun paikkaan ryhmän jäsenten saataville.

9.5 Testauksen päättyminen

Kaikki testaus päättyy **18.4** riippumatta siitä, miten keskeneräinen ohjelma on. Tästä päivämäärästä ei voida joustaa ilman erittäin hyvää syytä. Testauksen päättyminen ei siis vaadi testauksen tai ohjelman hyväksymistä.

Testaus voidaan päättää jo aikaisemmin, jos sekä projektipäällikkö että asiakas hyväksyvät ohjelman. Hyvää syytä aiempaan testauksen päättämiseen on kuitenkin vaikea keksiä.

Testauksen päättymisen jälkeen ohjelmaan voi vielä tehdä korjauksia ennen lopullisen version julkistamista. Ohjelma julkistetaan suunnitelmien mukaan, riippumatta sen valmiusasteesta.

LÄHTEET

[1] Kuntokirjurin sivu SourceForgessa, http://sourceforge.net/projects/kuntokirjuri

[2] Junit-testaustyökalu, http://www.junit.org/

[3] JasperReports,
http://www.jasperforge.org/jaspersoft/opensource/business_intelligence/jasperreports/

Lähteet dokumentin kirjoittamisessa

Kuopion Johdatus Testaukseen (JTE) -kurssin luentomoniste vuodelta 2007

TUT:n ohjelmistotekniikan laitoksen dokumenttirunkopankki:

http://www.cs.tut.fi/ohj/dokumenttipohjat/

Wikipedia: http://en.wikipedia.org/wiki/Category:Software_testing

LIITTEET:

Liite 1 – Testipöytäkirjan malli

Testaaja (/t):

Pvm	12.5.2010
Testauksen kohde	Esimerkkimoduuli
Koodin versio	0.01
Testausmenetelmä	Mustalaatikkotestaus
Testausympäristö	

Testitapaukset

Tunnus	xyz
Saatu tulos	Odotettu ( ) / Ei odotettu ( )
Virheen kuvaus	

Uudet testitapaukset

Uusien testitapausten kuvaukset ja saadut tulokset.

Testauksessa syntyneet muut tiedostot

Esim. Lokitiedostot, kuvankaappaukset, paperipöytäkirjat

Testauksen yhteenveto

Liite 2 – Vaatimukset

Toiminnalliset vaatimukset

Vaatimuksen numero	Toiminnon nimi
1	Uuden profiilin luonti
2	Sisäänkirjautuminen
3	Profiilin asetusten muokkaus
4	Profiilin tallennus ja tuonti
5	Uloskirjautuminen
6	Oman profiilin poisto
7	Tietojen lisäys
8	Tavoitteiden asetus
9	Muutettujen tietojen tallennus
10	Vanhojen merkintöjen selaus
11	Aiempien merkintöjen muokkaus
12	Raportin luonti
13	Terveyskortti
14	Uuden tyypin tai attribuutin lisäys
15	Tyypin tai attribuutin poisto

Ei-toiminnalliset vaatimukset

Vaatimuksen numero	Toiminnon kuvaus
16	Ohjelman tulee olla mukava käyttää
17	Ohjelman tulee olla helposti lähestyttävä
18	Ohjelman käytön tulee olla intuitiivista
19	Ohjelma ei jätä käyttäjää pulaan
20	Ohjelmassa käytetään hyvää yleiskieltä
21	Ohjelman tulee olla nopea
22	Järjestelmä tulee voida helposti siirtää eri alustoille
23	Ohjelma suojaa käyttäjän yksityiset tiedot