TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Tietoliikennetekniikan suuntautumisvaihtoehto

Tutkintotyö

Jyri Jokela

MUSTALAATIKKOTESTAUS MOBIILIPÄÄTELAITTEELLA

Tutkintotyö, joka on jätetty opinnäytteenä tarkastettavaksi Jyri Jokelan insinöörin tutkintoa varten Tampereella 17.11.2007

Työn ohjaaja test team leader Jussi-Pekka Peltonen, Flander Oy
Työn valvoja lehtori Ari Rantala, Tampereen ammattikorkeakoulu
Tampere 2007

Tekijä: Jyri Jokela

Työn nimi: Mustalaatikkotestaus mobiilipäätelaitteessa

Päivämäärä: 17.11.2007

Sivumäärä: 46

Hakusanat: Ohjelmistotestaus, mustalaatikkotestaus, mobiilipäätelaitteet

Koulutusohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Tietoliikennetekniikka

Työn valvoja: lehtori Ari Rantala, Tampereen ammattikorkeakoulu

Työn ohjaaja: test team leader Jussi-Pekka Peltonen, Flander Oy

Ohjelmistojen testauksen rooli kasvaa jatkuvasti. Testauksen laatu antaa varsin suoraan vastauksen siihen, kuinka laadukas ohjelmisto kokonaisuudessaan on. Ohjelmistoprojektin testaukseen käytettävät resurssit kertovat lähes suoraan ohjelmistoprojektin johtoportaan kunnianhimosta ja kestävän kehityksen tavoitteista.

Mustalaatikkotestaus on lasilaatikkotestauksen ohella yleisin lähestymistapa ohjelmistoprojektien testaussuoritteille. Opinnäytetyössä selvitetään testaustekniikoiden roolit ja tarpeet nykypäivän testausprojekteissa sekä erityisesti annetaan kattavan kuvaus mustalaatikkotestauksen merkityksestä nykypäivän mobiilipäätelaitetestauksessa. Tarkoituksena on myös osaltaan perustella mustalaatikkotestauksen pätevyyden muihin testaustekniikoihin verrattuna.

Työn lopussa otetaan esille käytännön mustalaatikkotestausesimerkkejä, joiden avulla mobiilipäätelaitteita testataan. Näiden mustalaatikkotestausesimerkkien avulla havainnollistetaan miten mobiilipäätelaitteen käytetyimpiä applikaatioita voitaisiin testata.

Tämä opinnäytetyö ei sinänsä tarjoa mitään ongelmanratkaisuja eri projekteihin, vaan se toimii eräänlaisena oppaana mustalaatikkotestaukseen erikoistuneelle testausalan ammattilaiselle.

Author: Jyri Jokela

Name of the thesis: Black box testing in mobile phones

Date: 17.11.2007

Number of pages: 46

Keywords: Software testing, black box testing, mobile phones

Degree programme: Computer systems engineering

Specialisation: Telecommunications engineering

Supervisor: senior lecturer Ari Rantala, Tampere Polytechnic

Instructor: test team leader Jussi-Pekka Peltonen, Flander Oy

The role of the software testing is increasing constantly. The quality of the testing corresponds to the quality of the whole software. The resources spent to the software testing describe almost straight about the ambitions and the aims of the continuous development of the management level.

Black box testing along with glass box testing is the most common testing technique approach in the software projects. This thesis tries to find out the roles and the needs of the testing techniques in the testing projects at the present time. It also tries to give a wide perspective about the importance of the black box testing nowadays. The thesis has also an intention to explain the advantages of the black box test in comparison with the other testing techniques.

In the end of my thesis, some general black box testing examples will be presentated. These examples shows accurately how test engineer can test mobile phones. Thesis will also show how to test the most useful applications in the phone with the aid of these black box testing techniques.

Actually my thesis doesn’t offer any problemsolvings to different projects. The thesis can be used as a guide to the test engineers who are interested about black box testing in the mobile devices.

ALKUSANAT

Tämä tutkintotyö on tehty Tampereen ammattikorkeakoulun tietoliikennetekniikan insinöörityönä.

Haluan jakaa kiitokset Flander Oy:lle ja työprojektini esimiehelle Jussi-Pekka Peltoselle, joka antoi luvan tehdä tutkintotyöni työprojektiini liittyvästä aiheesta. Kypsyttelin työn aihetta 2007 alkuvuoden ja työn kirjoitus tapahtui kesän ja syksyn 2007 aikana. Erityiskiitokset kuuluvat myös tyttöystävälleni Lenille ja perheenjäsenilleni, jotka antoivat voimia työn valmistumisvaiheessa. Ari Rantala työnvalvojana ansaitsee myös maininnan työn nopeasta tarkastamisesta ja työhön liittyvistä ohjeista.

Opinnäytetyöni ei ole salainen.

Tampereella 17.11.2007

Jyri Jokela

SISÄLLYSLUETTELO

TIIVISTELMÄ I
ABSTRACT II
ALKUSANAT III
SISÄLLYSLUETTELO IV
1 JOHDANTO 1
2 TESTAUSTEKNIIKAT 2
2.1 TESTAUSTEKNIIKOIDEN TARVE 2
2.2 TESTAUSTEKNIIKAN MÄÄRITELMÄ 3
2.3 TESTAUSTEKNIIKOIDEN HYÖDYT 4
2.4 TESTAUSTEKNIIKKATYYPIT 5
2.5 STAATTISET TESTAUSTEKNIIKAT 5
2.6 LASILAATIKKOTESTAUS 6
2.6.1 Lasilaatikkotestauksen periaatteet 6
2.6.2 Lasilaatikkotestauksen huonot puolet 7
2.6.3 Lasilaatikkotestaus ja V-malli 7
2.7 HARMAALAATIKKOTESTAUS 8
3 MUSTALAATIKKOTESTAUS 9
3.1 MUSTALAATIKKOTESTAUKSEN MÄÄRITELMÄ 9
3.2 MUSTALAATIKKOTESTAUKSEN HYVIÄ PUOLIA 10
3.3 MUSTALAATIKKOTESTAUKSEN HEIKKOUKSIA 12
3.4 EI-TOIMINNALLISET TESTAUSTEKNIIKAT 13
4 MUSTALAATIKKOTESTAUS V-MALLIN ERI TESTAUSTASOILLA 14
4.1 VALIDOINTI JA VERIFIONTI 14
4.2 V-MALLIN PERUSIDEA 14
4.3 KUINKA V-MALLI TOIMII KÄYTÄNNÖSSÄ? 15
4.4 V-MALLIN HYVIÄ PUOLIA 15
4.5 V-MALLIN HUONOT PUOLET 15
4.6 TESTAUSTASOT 16
4.6.1 Moduulitestaus 17
4.6.2 Integrointitestaukset 18
4.6.3 Järjestelmätestaus 20
4.6.4 Alfa-testaus 20
4.6.5 Beta-testaus 21
5 MUSTALAATIKKOTESTAUKSEN SUUNNITTELUTEKNIIKOITA 22
5.1 EKVIVALENSSILUOKITTELU 23
5.2 RAJA-ARVOANALYYSI 25
5.3 SYÖTTEIDEN KOMBINAATIOT 26
5.4 VIRHEENARVAUS 26

5.5 TILASIIRTYMÄTESTAUS 27
5.6 SYY-SEURAUS MALLINTAMINEN 29
5.7 SYNTAKSITESTAUS 29
5.8 SATTUMANVARAINEN TESTAUS 29

6 KÄYTÄNNÖN MUSTALAATIKKOTESTAUSTAPOJA 30

6.1 FUNKTIONAALINEN TESTAUS 30
6.2 KENTTÄTESTAUS 30
6.3 UI-TESTAUS 32

7 MOBIILIPÄÄTELAITTEEN APPLIKAATIOIDEN MUSTALAATIKKOTESTAUKSEN SUORITTAMINEN 33

7.1 SMS 33
7.1.1 SMS-applikaation mustalaatikkotestaus 34
7.2 MMS 35
7.2.1 MMS-applikaation mustalaatikkotestaus 35
7.3 GALLERIA 36
7.3.1 Galleria-applikaation mustalaatikkotestaus 36
7.4 WEB-SELAIN 37
7.4.1 WEB-selain-applikaation mustalaatikkotestaus 38
7.5 OSOITELUETTELO 39
7.5.1 Osoiteluettelo-applikaation mustalaatikkotestaus 39
7.6 SUORATOISTO-APPLIKAATIO 39
7.6.1 Suoratoisto-applikaation mustalaatikkotestaus 40
7.7 ÄÄNIPUHELU 40
7.7.1 Äänipuhelu-applikaation mustalaatikkotestaus 41
7.8 VIDEOPUHELU 42
7.8.1 Videopuheluapplikaation mustalaatikkotestaus 42
7.9 BLUETOOTH 42
7.9.1 Bluetooth-applikaation mustalaatikkotestaus 42
7.10 KELLO 43
7.10.1 Kello-applikaation mustalaatikkotestaus 43

8 YHTEENVETO 44

LÄHDELUETTELO 45

1 JOHDANTO

Mustalaatikkotestausta pyritään tarkastelemaan mahdollisimman monesta näkökulmasta työssäni. Työ aloitetaan dynaamisen testauksen ja siihen kuuluvien testaustekniikoiden tarkastelulla kappaleessa 2. Samaisessa kappaleessa mainitaan ohimennen myös staattisesta testauksesta sekä vertaillaan mustalaatikkotestausta lasi- ja harmaalaatikkotestaukseen. Samalla kerrotaan, millä tavoin eri mustalaatikkotekniikat eroavat toisistaan. Jatkossa siirrytään sitten tarkemmin työn olennaisimpaan dynaamiseen testaustekniikkatyyppiin eli mustalaatikkotestaukseen. Tämä tekniikkatyyppi määritellään mahdollisimman pikkutarkasti sekä esitellään sen suurimmat hyödyt ja haitat kappaleessa 3.

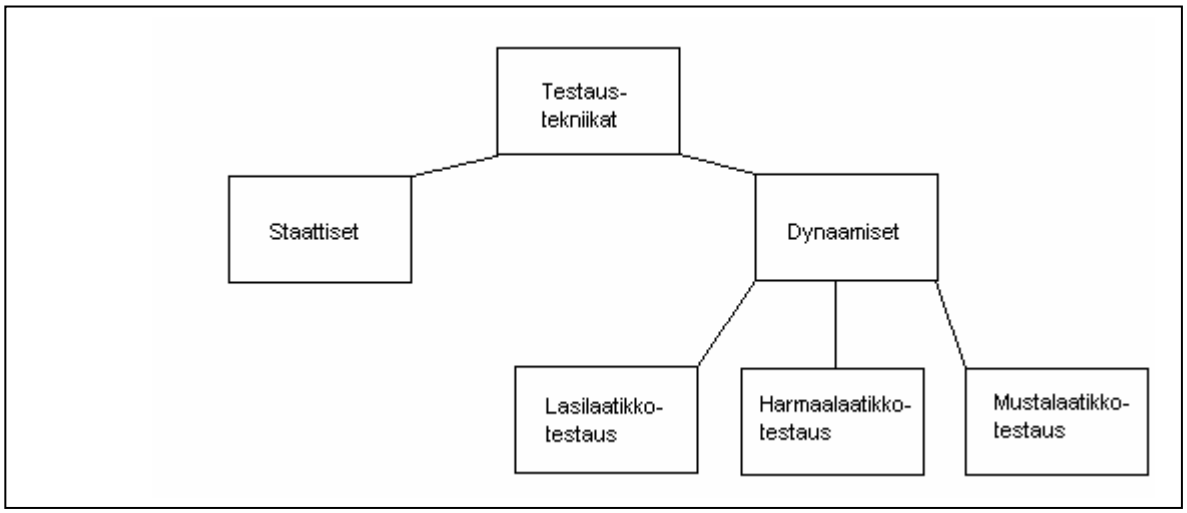
Kappaleessa 4 määritellään V-mallin rooli testauksessa ja pyritään samalla selvittämään mustalaatikkotestauksen roolit ja tarpeet V-mallin eri tasoilla. Myös mustalaatikkotestaus jakautuu vielä alempiin testaustekniikkatasoihin ja näistä kerrotaankin luvusta 5 alkaen. Olennaisimmat mustalaatikkotestauksen suunnittelutekniikat ovat ekvivalenssiluokittelu sekä raja-arvoanalyysi.

Kappaleessa 6 esitellään pääpiirteittäin käytännön mustalaatikkotestauskohteista, joita olen käyttänyt mustalaatikkotestaustekniikoita työurani aikana. Kappaleessa 7 otetaan tarkempaan syyniin vielä 10 erilaista applikaatiota, joilla on suuri merkitys nykypäivän matkapuhelimissa ja selvitän, millä tavoin eri mustalaatikkotestauskohteilla näitä voidaan testata.

Olen päässyt tutustumaan erilaisiin mustalaatikkotestaustekniikoihin Flander Oy:ssä viettämäni työurani aikana. Flander Oy on vuonna 1997 perustettu ohjelmistojen testaukseen ja kehitykseen panostava firma. Asiakkaina Flander Oy:llä ovat matkapuhelinvalmistajat, ohjelmistotoimittajat, operaattorit ja palveluntarjoajat. Flander Oy:n päämääränä on tarjota asiakkaalle korkealaatuisia palveluita, jotka vastaavat mahdollisimman tarkasti asiakkaan vaatimuksia. Flander Oy:n kasvu oli vuosina 2000–2004 80 % vuodessa ja kasvua tapahtuu edelleen. Nykyään Flanderilla on konttoreita ympäri Suomea sekä yksi konttori niin Kiinassa kuin Ruotsissa. Työntekijöitä Flanderilla on tällä hetkellä noin 330. /17/

2 TESTAUSTEKNIIKAT

Opinnäytetyö aloitetaan havainnollistamalla mustalaatikkotestauksen sijaintia testauksen ja testaustekniikoiden hierarkiassa. Kuvasta 1 selviää, että mustalaatikkotestaus kuuluu yhdessä lasi- ja harmaalaatikkotestauksen kanssa dynaamisiin testaustekniikoihin. Staattiset ja dynaamiset testaustekniikat taas muodostavat yhdessä testaustekniikoiden päälohkon. Tässä kappaleessa pohjustetaan kaavion ylimpänä olevien yleisten testaustekniikoiden roolia, yleistä tarvetta sekä hyötyjä testaussuorituksen kannalta. Tässä kappaleessa tarkastellaan myös staattisia testaustekniikoita sekä dynaamisista testaustekniikoista harmaalaatikko- ja lasilaatikkotestausta. Mustalaatikkotestaukseen keskitytään tarkemmin tulevissa kappaleissa.



Kuva 1: Testaustekniikoiden jaottelu

2.1 Testaustekniikoiden tarve

Ohjelman testaaminen 100-prosenttisesti on käytännössä mahdotonta, koska testattavia ohjelman sisäänmenoja, esivaatimuksia ja niiden kombinaatioita on lähes ääretön määrä. Tästä syystä testaajan täytyy valita hyvin pieni osajoukko kaikista mahdollisista testitapauksista. Osajoukko kannattaa valita niin, että valitun joukon arvoilla löydetään mahdollisimman paljon virheitä järjestelmän toiminnasta.

Testaajien omat kokemukset ja kokeilut ovat osoittaneet sen, että osajoukon valitseminen sattumanvaraisesti ei ole kovin tehokas tapa löytää virheitä. Osajoukon valinnan täytyy siis perustua juuri edellisiin testauskokemuksiin ja muutenkin järjelliseen ajatustoimintaan. Testaustekniikat pyrkivät hyödyntämään juuri näitä edellä mainittuja järjellisiä ratkaisuja testauksen suorittamisessa. /1/

2.2 Testaustekniikan määritelmä

Testaustekniikka on proseduuri, joka helpottaa testiryhmää valitsemaan ja suunnittelemaan oikeanlaisen testitapausosajoukon testattavalle järjestelmälle koko syöteavaruudesta. Eri tekniikat tarjoavat erilaisia näkökulmia testattavalle järjestelmälle. Jokainen tekniikka pyrkii asettamaan testaajalle sääntöjä ja ohjenuoria, joita hänen tulisi seurata eri testiolosuhteissa ja - tapauksissa. Tekniikat perustuvat joko käyttäytymis- tai rakenteelliseen malliin järjestelmästä. Toisin sanoen ne perustuvat järjestelmän käyttäytymisen tai rakenteen ymmärtämiseen. Käyttäytymisellä tarkoitetaan toiminnallisuutta ja ei-toiminnallisia asioita, kuten suorituskykyä tai järjestelmän helppokäyttöisyyttä. Rakenteellisuudella taas tarkoitetaan sitä että tiedetään, miksi järjestelmä toimii niin kuin toimii.

On olemassa monia erilaisia testaustekniikoita, ja kaikki ovat omalla tavallaan hyviä virheiden löytämisen kannalta. Kaiken kaikkiaan, testitekniikan avulla on helppo tunnistaa hyvä testitapaus.

Hyvä testitapaus ottaa huomioon seuraavat ominaisuudet:

- Tehokkuus, testitapauksella on potentiaalia löytää virheitä
- Esimerkillisyys, edustaa muitakin testitapauksia
- Ylläpidon helppous
- Taloudellisuus, sen käyttö ei maksa paljon /1/

2.3 Testaustekniikoiden hyödyt

Eri ihmiset, jotka käyttävät samaa tekniikkaa samassa systeemissä, päätyvät lähes varmasti erityyppisiin testitapauksiin, mutta kaikilla testaajilla on silti yhtä suuri todennäköisyys löytää virheitä. Tämä johtuu siitä, että testitekniikka ohjastaa testaajia samanlaisiin näkökulmiin testattavasta järjestelmästä ja tekemään samankaltaisia olettamuksia testauksen yhteydessä. Täytyy kuitenkin myös huomioida se, että jokaisella testaajalla on tietty subjektiivinen käsitys järjestelmän toiminnasta testitekniikoita käytettäessä, jonka kautta testitekniikoiden käyttö ja lopputulokset vaihtelevat testaajasta riippuen.

Tekniikat pyrkivät parhaassa tapauksessa haastamaan ennakkoluuloja, joita on ennen testisuoritusta asetettu järjestelmälle. Näin ollen testitekniikoiden avulla saatetaan löytää täysin uusi näkökulma testauksen kohteeseen ja täysin uuden tyyppisiä ohjelmistovirheitä saatetaan löytää.

Olennaisin asia testitekniikoiden käytössä on se, että testaus saataisiin entistä tehokkaammaksi. Tehokkuus tarkoittaa lähinnä sitä, että virheitä löydetään vähemmällä vaivalla ja pienemmillä resursseilla. Jos kaksi erilaista testisettiä löytävät saman virhemäärän testattavasta ohjelmasta, niin ratkaisevaksi tekijäksi muodostuvat testaukseen kuluvat kustannukset. Hyvä testaus ei siis ainoastaan maksimoi tehokkuutta vaan myös minimoi kustannuksia.

Koska tekniikka keskittyy tietyntyyppisten virheiden löytämiseen, on hyvin todennäköistä, että testit löytävät enemmänkin tämäntyyppisiä virheitä. Valitsemalla asianomaiset testitekniikat on mahdollista kontrolloida vielä tarkemmin sitä, mitä testataan ja näin ollen vähentää rinnakkaista testausta eri testitapausten välillä. Testaajien luottamus oikean testauksen tekemiseen on myös vahva testitekniikoita käytettäessä, sillä testattavat asiat ovat juuri ne seikat, jotka ovat eniten testauksen tarpeessa. /1/

2.4 Testaustekniikkatyypit

On olemassa monia erilaisia ohjelmiston testaustekniikkatapoja, joilla jokaisella on omat vahvuudet ja heikkoudet. Jokainen yksittäinen tekniikka on hyvä etsimään tietynlaisia virheitä, mutta se voi olla taas heikompi löytämään toisenlaisia virheitä. Esimerkiksi tekniikka, jonka tarkoituksena on löytää ylin ja alin arvo sisäänmenosyötteistä, löytää huomattavasti helpommin virheitä raja-arvoanalyysin mukaisista virheistä kuin virheistä, jotka liittyvät esimerkiksi syötteiden kombinaatioihin. Ohjelmistotestauksen eri elämänkaaren vaiheissa tehdyt testaukset tulevat löytämään erityyppisiä virheitä, sen mukaan missä elämänkaaren vaiheessa niitä käytetään. Esimerkiksi komponenttitestaus löytää enemmän koodaukseen liittyviä virheitä kuin järjestelmän suunnitteluvirheitä.

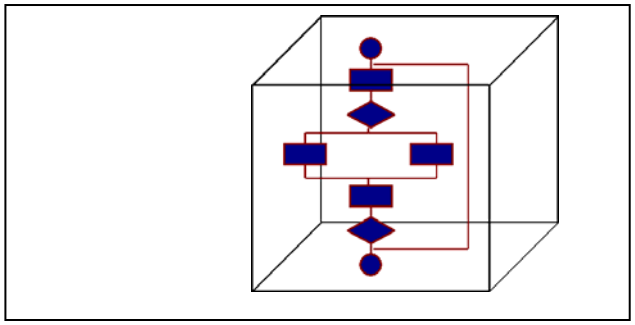
Jokainen testaustekniikka jakaantuu moneen eri alakategoriaan. On varsinaisesti olemassa vain kaksi pääkategoriaa: staattinen ja dynaaminen. Kuitenkin dynaamiset tekniikat on jaettu vielä kahteen alalohkoon musta- ja lasilaatikkotestaukseen. Hieman epävirallisempi tekniikka on harmaalaatikkotestaus, joka sijoittuu musta- ja lasilaatikkotestauksen välimaastoon. Mustalaatikkotestaustavat voidaan jakaa vielä toiminnallisiin ja ei-toiminnallisiin tekniikoihin. /1/

2.5 Staattiset testaustekniikat

Staattiset testaustekniikat eivät tutki ohjelmistokoodia lainkaan. Niitä käytetään yleisesti ennen kuin mitään testejä on tehty ohjelmistolle. Niitä voidaan kutsua myös ei-suoritteellisiksi tekniikoiksi. Useimmilla staattisilla testaustekniikoilla voidaan testata erityyppisiä dokumentteja, kuten lähdekoodia, arkkitehtuuria, suunnitteludokumentteja sekä toiminnallisuus- ja vaatimusspesifikaatioita. Testaus tapahtuu siten, että ohjelmaa ei suoriteta lainkaan testauksen yhteydessä. Staattisia testaustapoja ovat esimerkiksi erityyppiset arvioinnit (*audit, assessment*), katselmoinnit (*technical reviews*), tarkastukset (*inspection*) ja läpikäynnit (*walkthrough*). /1/

2.6 Lasilaatikkotestaus

Mustalaatikkotestauksen ohella käytetyin dynaaminen testaustekniikka lienee lasilaatikkotestaus (*Clear/Glass/White box testing, structural testing*). Lasilaatikkotestausta kutsutaan usein myös rakenteelliseksi – tai valkolaatikkotestaukseksi. Käsitteellä ”lasilaatikko” kuvataan sitä, että testaaja voi nähdä ohjelmiston rakenteen sisälle testauksen yhteydessä. Tästä esimerkkinä on kuvassa 2 nähtävillä oleva järjestelmä, jossa kuvan mukaisesti kuvitteellisen ohjelmiston rakenne ja polut ovat nähtävillä testaajalle. Kuvitteelliset rakenteet on merkitty eri symbolein kuvaan. Tässä kappaleessa tarkastelen lähemmin lasilaatikkotestausta eli sen yleistä periaatetta ja sen hyödyllisiä puolia. /1, 3, 7, 8/



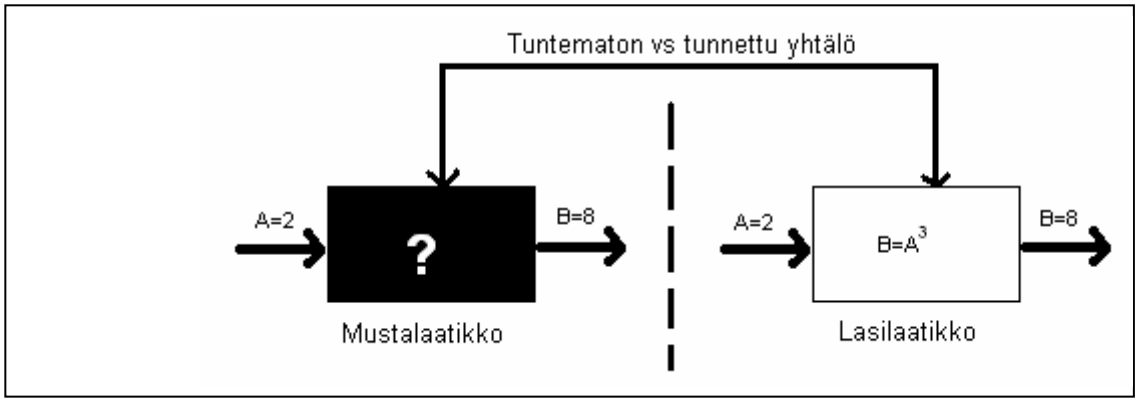
Kuva 2: Lasilaatikkotestauksessa tunnetaan testattavan kohteen sisältö

2.6.1 Lasilaatikkotestauksen periaatteet

Rakenteellisessa testauksessa käytetään ohjelmiston sisäistä rakennetta hyödyksi, kun mietitään testitapausten luontia ja testidatan valintaa. Kun tiedetään koodin ongelmakohta, voidaan keskittyä siihen ja siihen liittyviin testitapauksiin. Tämä tekniikka vaatii testaajalta tietämystä siitä, kuinka ohjelmisto toteutetaan ja millaiset toimintaperiaatteet ohjelmistolla on. Näin ollen testaajalta vaaditaan myös jonkin verran ohjelmointitaitoja, jotta hän voisi vaivattomasti tunnistaa ohjelmapolut, jotka esiintyvät ohjelmakoodin sisällä, ja sitä kautta muodostaa helpommin lasilaatikkotestitapauksia.

Testaaja siis valitsee erilaisia testitapaussyötteitä, joiden perusteella hän päättelee koodin sisäisen kulun ohjelmakoodissa, minkä avulla taas voi päätellä oikeat ulostulot eri syötteille. Lasilaatikkotestaus tarkastelee siis pelkistäen sitä, että onko koodi hyvin

rakennettua ja tehokasta. Vaikeat silmukkarakenteet, globaalit muuttujat, turhat koodirivit ja sekavat ehtolausemuodostelmat ovat esimerkkejä huonoista rakenteista, jotka selviävät lasilaatikkotestauksen avulla. Testaus suoritetaan rivi riviltä-tyylillä, joka on edelleenkin monien mielestä tehokkain rakenteellisen testauksen muoto. Tyypillistä lasilaatikkotestauksessa on myös se, että mahdolliset ongelmat löytyvät melko nopeasti. Kuvassa 3 on nähtävillä mustalaatikko- ja lasilaatikkotestauksen periaatteellinen ero. /1, 3, 7, 8/



Kuva 3: Mustalaatikkotestaus vs lasilaatikkotestaus

2.6.2 Lasilaatikkotestauksen huonot puolet

Lasilaatikkotestauksen huono puoli on se, että testaus on kovin hidasta ja sitä kautta kallista. Lasilaatikkotestaus vaatii usein tueksi muitakin testausmenetelmiä ja näin ollen sen suoritus yksittään on lähes mahdotonta. Testaustyöhön vaaditaan kovan luokan ammattilaisia, jotka ovat mahdollisesti paneutuneet juuri tiettyyn kieleen ja ympäristöön ja jotka taitavat niin testaamisen kuin koodaamisen. Rakenteellinen tekniikka voi keskittyä esimerkiksi erilaisten silmukoiden testaukseen ohjelmistoissa. Testitapauksen mukaan silmukoiden testauksen toistojen määrä saattaa nousta hyvinkin suureksi, joka vie täten aikaa ja resursseja. /1, 3, 7, 8/

2.6.3 Lasilaatikkotestaus ja V-malli

Lasilaatikkotestausta suoritetaan perinteisesti V-mallin kuvaamilla alatasoilla, kuten komponenttitestaustasolla ja pienellä integrointitestaustasolla. Tämä johtuu siitä, että kun yksiköt on integroitu systeemiin, lasilaatikkomenetelmät muuttuvat

mahdottomiksi käyttää eikä niillä saavuteta toivottua tulosta. Toisin sanoen lasilaatikkomenetelmillä keskitytään yksityiskohtaisten asioiden tarkasteluun, kun taas mustalaatikkotestauksessa keskitytään suurempien kokonaisuuksien tarkasteluun. Näin ollen mustalaatikkotestaus soveltuu kaikille testaustasoille ja erityisesti V-mallin ylätasoille. /1, 3, 7, 8/

2.7 Harmaalaatikkotestaus

Lasi- ja mustalaatikkotestauksen lisäksi käytetään joissain yhteyksissä myös mustalaatikko- ja lasilaatikkotestaustekniikan yhdistelmää eli harmaalaatikkotestausta. Sekä musta- että lasilaatikkotestaukset ovat tärkeitä testauskokonaisuuden kannalta. Yksinään ne eivät siltikään kata kaikkea mahdollista, sillä mustalaatikkotestaus ei paljasta tiettyjä virhetyyppejä kuten tietovirta- tai raja-arvovirheitä lähdekooditasolla. Tiedonkulkuongelmat järjestelmän päästä päähän saattavat jäädä myös huomaamatta mustalaatikkotestausmenetelmin. Lasilaatikkotestaus ei taas huomaa mahdollisia riskejä käyttöjärjestelmässä, ohjelmiston määrityksissä, laitteiden yhteensopivuuksissa, aikaan liittyvissä virheissä ja käytettävyydessä. Harmaalaatikkotestauksen avulla tällaiset ongelmat saadaan paljastettua helpommin.

Pitää muistaa, että musta- ja lasilaatikkotestaus eivät silti ole toisiaan poissulkevia testaustapoja, vaan ne täydentävät parhaimmillaan toisiaan eri soveltamisalueilla. Näitä kahta tekniikkaa päästään parhaiten täydentämään harmaalaatikkotestauksen menetelmin, jotka sijoittuvat mustalaatikko- ja lasilaatikkotestauksen väliin. Harmaalaatikkotestauksessa tai toiselta nimeltään hybriditekniikassa pyritään yhdistämään molempien testaustapojen hyvät puolet. Harmaalaatikkotekniikat ovat käyttökelpoisia kaikilla testauksen tasoilla. Niiden tarkoituksena on selvittää käyttäjäpuolen lopputulos, järjestelmäkohtaista teknistä tietoa sekä käyttöjärjestelmää. Tämän kaltainen testaus arvioi myös sovelluksen suunnittelun mukaisesti järjestelmän osien yhteensopivuutta. Erityisen hyvin tämän tyyppinen testaus sopii web-sovellusten testaukseen, koska web-sovellukset koostuvat monista ohjelmisto- ja laitteistopuolen komponenteista. Edellä mainitut komponentit pitää testata järjestelmän suunnittelun pohjalta, jotta voidaan arvioida niiden toimivuutta ja yhteensopivuutta. /11/

3 MUSTALAATIKKOTESTAUS

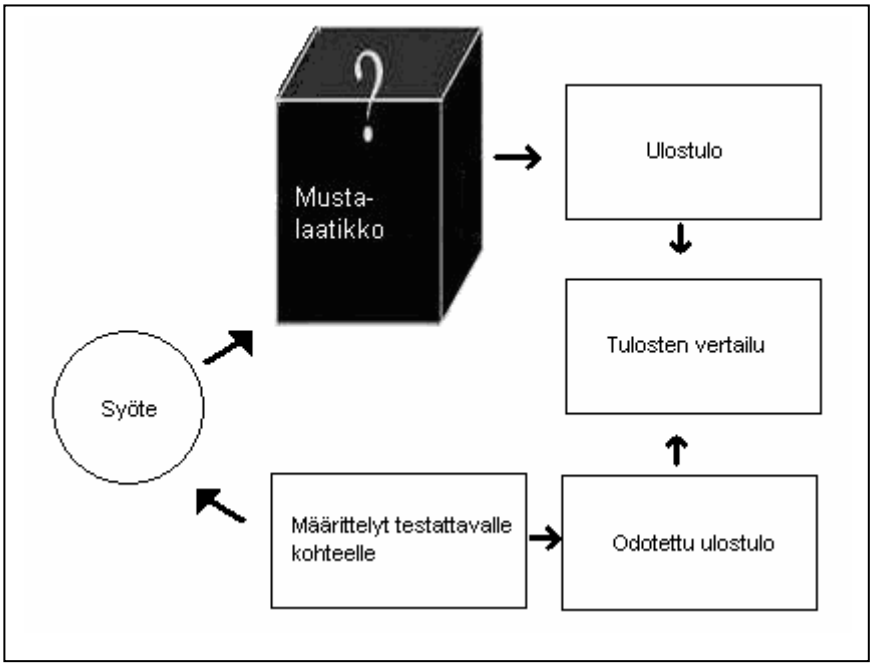
Tässä kappaleessa keskitytään työn olennaisimpaan testaustekniikkaan eli mustalaatikkotestaukseen. Kappaleessa määritellään mustalaatikkotestaus mahdollisimman yksityiskohtaisesti sekä kerrotaan mustalaatikkotestauksen hyvät ja huonot puolet. Lopussa tarkastellaan pikaisesti mustalaatikkotestaukseen kuuluvia ei-toiminnallisia testaustekniikoita.

3.1 Mustalaatikkotestauksen määritelmä

Mustalaatikkotestauksessa (*black box testing*) testattavaa ohjelmaa tai järjestelmää tarkastellaan mustana laatikkona, jonka sisäisestä rakenteesta tai detaljeista testaaja ei tiedä juuri mitään. Tämän tyyppisessä testauksessa ohjelmalle tai järjestelmälle annetaan syötteitä (*input*) ja näiden syötteiden käyttäjille näkyviä tuloksia (*output*) verrataan spesifikaatioiden ja vaatimusten mukaisiin ohjearvoihin tuloksen oikeellisuuden varmistamiseksi.

Merkittävän roolin muodostavat myös tuotteelle asetetut laatuvaatimukset, joiden mukaan asetetaan tulosarvot kelvollisiksi tai epäkelvollisiksi. Jos siis tietty ohjelman osio toimii ulospäin katsottuna oikein spesifikaatioiden ja vaatimusten määräämissä rajoissa, niin sen lähdekoodia ei tarvitse muokata. Mustalaatikkotestauksessa keskitytään siis suurelta osin ulkoisesti havaittavien toiminnallisuuksien ja piirteiden testaamiseen ja sen takia sitä kutsutaankin usein toiminnallisuustestaukseksi. Mustalaatikkotestauksessa tarkastellaan samalla myös ohjelman kommunikointia muiden ohjelmakomponenttien kanssa. Jossain yhteyksissä on myös todettu, että mustalaatikkotestauksen tehtävä korostuu testausprojektin suunnitteluvaiheessa ja sen vaikutusta on vaikea huomata testien implementointivaiheessa. Mustalaatikkotestauksen toimintaperiaate on kuvattu kuvassa 4. Mustalaatikkotestauksen periaatekuvaan voisi mahdollisesti lisätä myös funktionaalisen testauksen kriteereiden kohteita eli funktioita, kuten

mustalaatikkotestaukseen kulunut aika ja vaikkapa ympäristö, jossa ohjelmistoa on testattu. /2, 3, 4/



Kuva 4: Mustalaatikkotestauksen yleiskuva

3.2 Mustalaatikkotestauksen hyviä puolia

Mustalaatikkotestauksen etuna on se, että se soveltuu ainakin jossain määrin kaikille testauksen tasoille. Mustalaatikkomenetelmät löytävät myös ohjelmasta puuttuvan toiminnallisuuden sekä virheelliset suorituspolut toisin kuin esimerkiksi lasilaatikkotestaus.

Mustalaatikkotestauksen hyvänä puolena voidaan pitää myös sitä, että se lähestyy myös paremmin laatukäsitettä, jossa asiakas määrittelee testattavan moduulin osan, jonka testaajat testaavat huolella ja tarkasti. Asiakas voi myös määritellä toleranssin, jonka arvojen mukaisesti ohjelman on toimittava. Näin ollen mustalaatikkotestaukselle voidaan määrittää tietyt kriteerit ja spesifikaatiot, joiden rajoissa sen oletetaan toimivan ja jonka perusteella testausprosessia voidaan tehostaa. Jos ohjelmaan halutaan tehdä muutoksia, niin ohjelma voidaan ajaa samoilla

kriteereillä uudestaan ja nähdään heti, onko ohjelma pysynyt vaatimuksissa vai ilmeneekö uusia ongelmia.

Mustalaatikkotestaus sopii huomattavasti paremmin pitkien koodirivien tutkimiseen kuin esimerkiksi lasilaatikkotestaus. Yksi paremmuuden tekijöistä on ekvivalenssiositusten kaltaisten keinojen käyttömahdollisuus. Niissä iso testauskokonaisuus voidaan jakaa pienempiin osiin ja näin ollen helpottaa testauksen kokonaismäärää. Hyväksi puoleksi voidaan lukea myös se, että testaajan ei tarvitse tietää teoriassa applikaation ohjelmointikielestä tai - tavasta mitään. Riittää siis, että testauksen suorittaa testausalan ammattilainen. Testausalan ammattilainen osaa organisoida ja sisäistää testiprojektin kokonaisuudessaan ja samalla suorittaa testitapaukset mahdollisimman systemaattisesti ja huolellisesti.

Muutenkin on todella tärkeää testauksen lopputuloksen takia, että testaaja ja ohjelman kehittäjä ovat 2 eri henkilöä, jolloin on huomattavasti todennäköisempää löytää virheitä applikaatiosta. Jos applikaation ohjelmoija olisi myös saman applikaation testaaja, niin asioita tulisi tarkasteltua helposti liian subjektiivisesti ja liian kapeasta näkökulmasta, jolloin tietyntyyppiset virheet jäisivät helposti huomaamatta. Mustalaatikkotestauksessa molemmat henkilöt toimivat itsenäisesti puuttumatta toistensa työntekoon, koska tuloksia syntyy parhaiten tällä tavoin.

Testien näkökulmaksi valitaan loppukäyttäjän näkökulma, jolloin voidaan varmistaa parhaiten, että loppukäyttäjälle haitalliset virheet saadaan minimoitua. Mustalaatikkotestaus on myös helppo ja nopea aloittaa, sillä testitapaukset voidaan luoda heti, kun testattavan applikaation spesifikaatio on valmis. Mustalaatikkotestauksen yhteydessä on helppo paljastaa myös spesifikaation ristiriidat ja epäjohdonmukaisuudet.

Nykytrendi ohjelmistokehityksessä on oliokeskeinen, eli ohjelmointi tapahtuu eri olioiden pohjalta. Mustalaatikkotestaus soveltuu tämän tyyppiseen ohjelmistokehitykseen täydellisesti, sillä ohjelmiston eri moduulit voidaan testata sellaisinaan eikä näin ollen koko koodirakennetta tarvitse käydä läpi kerralla. /3, 5/

3.3 Mustalaatikkotestauksen heikkouksia

Ongelmana mustalaatikkotestauksessa pidetään yleisesti sitä, että testauksen kattavuus on tietyissä tapauksissa melko suppea, jos testitapausten suunnittelija ei ole onnistunut testauksen määrittelyssään tarpeeksi hyvin. Tämä johtuu siitä, että syötevaihtoehtojen määrä on käytännössä ääretön ja näiden kaikkien vaihtoehtojen testaaminen kestää liian kauan. Tästä syystä ei voida ikinä olla täysin varmoja, että applikaatio olisi täysin virheetön.

Mustalaatikkotestauksessa saatetaan myös keskittyä hieman epäoleellisempiin asioihin joidenkin applikaatioiden toiminnan kannalta, kuten kuormitukseen, suorituskykyyn ja käyttöliittymän yleiseen toimintaan. Näitä kutsutaan ei-toiminnallisiksi testaustekniikoiksi.

Mustalaatikkotestauksen optimaaliseen tekemiseen vaaditaan erittäin tarkkoja ja johdonmukaisia spesifikaatioita applikaatiosta. Jos spesifikaatiot ovat köykäisiä, niin testitapaukset saattavat keskittyä epäolennaisiin asioihin tai kattaa liian vähän käsitellystä aihepiiristä. Näin ollen testaussuoritus saattaa jäädä vajaaksi monen polun testaamattomuuden vuoksi ja virheiden jälkikäteen ilmenemisen todennäköisyys kasvaa. Mustalaatikkotestauksessa saattaa esiintyä myös turhaa toistoa, jos tietty osa koodista on testattu jo koodauksen yhteydessä ja testaajalle ei ole ilmoitettu asiasta. Huonoksi puoleksi mustalaatikkotestauksessa voidaan sanoa myös se, että testausta ei voida kohdistaa tiettyihin koodin osiin, joiden tiedetään sisältävän suurella todennäköisyydellä virheitä, koska mustalaatikkotestauksessahan ei tiedetä koodin sisällöstä paljoakaan. Tietyt koodin osaset ovat niin kompleksisia, että niissä voi myös olla tästä syystä lukematon määrä virheitä. Mustalaatikkotestaukseen ei ole muutenkaan kohdistettu tähän mennessä niin paljon tutkimusresursseja kuin esimerkiksi lasilaatikkotestaukseen, minkä vuoksi mustalaatikkotestauksesta ei ole saatu tehokäyttöön eikä toisaalta sen mahdollisuuksia ole tarkkaan selvitetty.
/3, 5/

3.4 Ei-toiminnalliset testaustekniikat

Ei-toiminnalliset testaustekniikat muodostavat osan mustalaatikkotekniikoista. Ei-toiminnalliset puolet järjestelmästä (tunnetaan myös laatunäkökulmina) sisältävät toiminnallisuus-, käytettävyys-, kuljetettavuus- ja ylläpitoaspekteja. Tämän kategorian tekniikat tarkastelevat sitä, kuinka hyvin systeemi tekee jotakin. Se ei siis itsessään tarkastele sitä, mitä systeemi tekee tai kuinka se toimii käytännössä. Tämän kategorian tekniikat eivät ole kovin ohjeellisia ja muodollisia. Muiden kategorioiden testit ovat riippuvaisia järjestelmän tyypistä eli siitä, mitä järjestelmä tekee, ja testeille varatuista resursseista. Ei-toiminnallisessa testauksessa vastaavasti tällaista riippuvuutta ei ole merkittävästi. /1/

4 Mustalaatikkotestaus V-mallin eri testaustasoilla

Tässä kappaleessa käsittelen V-mallin roolia testausprojektissa sekä sen hyviä ja huonoja puolia. Otan V-mallin jokaisen testaustason erikseen tarkasteluun ja pyrin tarkastelemaan mitä roolia mustalaatikkotestaus näyttelee kullakin testaustasolla.

4.1 Validointi ja verifionti

Ohjelmiston laatu pyritään selvittämään ja takaamaan validoinnin ja verifioinnin avulla. Tämän tyyppinen prosessi käydään läpi muun muassa testauksen V-mallin avulla. Validoinnilla pyritään tarkastamaan, toimiiko ohjelmisto asiakkaan vaatimusten mukaisesti ja sopiiko sovellus käyttötarkoitukseensa. Näihin kysymyksiin vastaaminen käy helpoimmin, kun ohjelmiston vaatimukset on kirjoitettu mahdollisimman tarkasti erilliseen dokumenttiin. Verifioinnilla selvitetään ohjelman toiminnan oikeellisuus vertaamalla verifiontituloksia tuotteen spesifikaatioon. Johtopäätöksenä voidaan siis sanoa, että validointi pyrkii vastaamaan kysymykseen, onko kehitetty tuote oikea? Verifioinnilla taas testataan, onko tuote kehitetty oikein? /1/

4.2 V-mallin perusidea

Testauksen kannalta V-malli on yksi suosituimmista mallinnuksista. Se soveltuu erityisesti projekteihin, joissa hyödynnetään vesiputousmallin ideaa. V-malli määrittelee yhdenmukaisen kehitys- ja testausprosessin ohjelmistoalan projekteihin. Prosessi etenee V-mallin mukaan seuraavassa järjestyksessä: vaatimukset, spesifikaatiot projektissa, spesifikaatiot järjestelmässä, spesifikaatiot suunnittelussa ja itse koodaus. Testauksen vaiheet taas tapahtuvat seuraavasti: moduulitestaus, pieni integrointitestaus, järjestelmätestaus, suuri integrointitestaus sekä hyväksymistestaukset, joihin kuuluvat alfa- ja beta-testaukset. V-mallissa testaus ei siis ole enää oma erillinen työvaiheensa, vaan se on jaoteltu erillisiin testausosiin ja sitä kautta integroitu eri kehitysprosessin osavaiheisiin.

Jokaisen ohjelmistotuotantoprosessin työvaiheen yhteydessä siis suunnitellaan oma räätälöity testipaketti ja V-mallissa testaus tapahtuu V-mallin osoittamalla ohjelmistokehitysprosessin tasolla. V-malli on kuvattu kuvassa 5. Testattavan kokonaisuuden koko suurenee siirryttäessä V-mallissa ylöspäin.

Kuten kuvasta 5 nähdään, niin käyttäjävaatimukset testataan hyväksymistestauksissa eli siis alfa- ja beta-testauksissa. Projektin spesifikaatioiden pohjalta tehdään suuren integrointitestauksen suunnitelmat ja testaukset. Järjestelmävaatimuksia mietittäessä ohjelmistoprojektin määrittelyvaiheessa suunnitellaan järjestelmätestaus. Arkkitehtuurisuunnitelmia kehiteltäessä suunnitellaan järjestelmän integrointitestaus. Toteutusvaiheen yhteydessä suunnitellaan moduulitestaus. /1/

4.3 Kuinka V-malli toimii käytännössä?

V-mallissa testausvaiheet on liitetty yhteen suunnitteluvaiheiden kanssa. Testaus voidaan aloittaa saman tien, kun vaatimukset ja suunnitelmat on saatu tehtyä testattavalle tasolle. Testauksessa saatuja tuloksia verrataan tarkkoihin, asiakkaalta tulleisiin, vaatimuksien mukaisiin arvoihin. Vaatimukset ovat enemmän yksityiskohtaisempia V-mallin alatasoilla kuin sen ylemmillä tasoilla.
/1, 3, 6, 9/

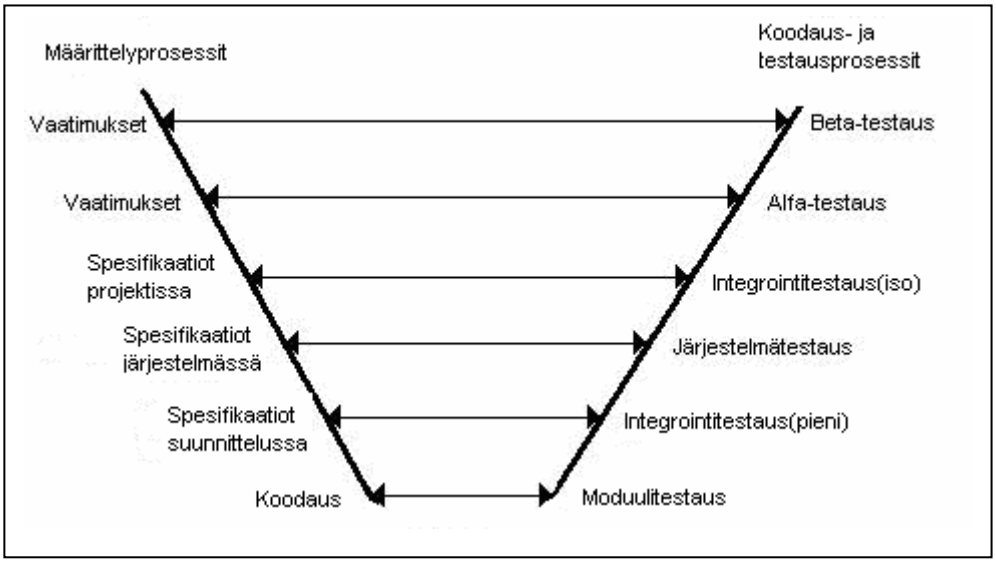
4.4 V-mallin hyviä puolia

V-mallissa on mahdollisuus toteuttaa suunnitelmat ja dokumentaatiot jo aikaisessa vaiheessa. Joissain tapauksissa se on mahdollista jo ennen koodin kirjoittamista ja tästä syystä koodin muodostaminen on helpompaa ja tehokkaampaa, kun tiedetään, mitkä ovat ohjelmiston tavoitteet. /1, 3, 6, 9/

4.5 V-mallin huonot puolet

Ohjelman kehittäjä voi kehittää koodia siltä pohjalta, että se läpäisee testitapaukset, mutta jättää samalla ongelmia testauksen ulkopuolelle. Tällöin olisi erityisen tärkeää,

että ohjelmiston kehittäjä ja testaaja olisivat eri henkilöitä. V-mallissa käytettyjen tynkien ja ajureiden käyttö lisää kustannuksia huomattavasti. /1, 3, 6, 9/

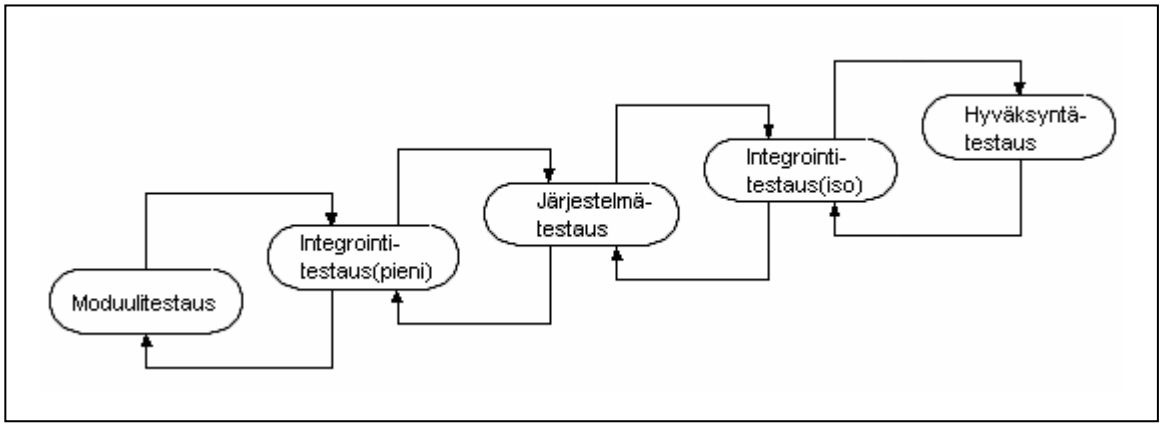


Kuva 5: V-malli

4.6 Testaustasot

V-mallin oikea puoli koostuu testausprosessin eri osasista eli testaustasoista (testing levels). Tarkemmin käsiteltynä testausprosessin toimintakaavio koostuu kuvan 6 mukaisesti. Tästä kuvasta näkee, että testaustasojen välillä on nuolet molempiin suuntiin. Tämä tarkoittaa sitä, että jos testaus menee alemmalta tasolta läpi, niin testaukset jatkuvat yhtä ylemmällä tasolla. Jos taas joku testattava kohde ei läpäise testaustasonsa määrittelemiä vaatimuksia, se palautetaan testaustasolle, jossa virheen uskotaan tapahtuneen. Tämän niin kutsutun regressiotestauksen mukaisesti virheet paikannetaan ja korjataan, minkä jälkeen se testataan uudelleen. Samalla testataan virhettä edeltäneet testit uudelleen, jotta voidaan varmistua, että uudet muutokset eivät ole synnyttäneet uusia virheitä alemmille tasoille.

Kuten jatkossa tulemme huomaamaan, niin siirryttäessä V-mallissa alimmalta tasolta ylöspäin testauksen luonne muuttuu lasilaatikkotestauksesta mustalaatikkotestauksen suuntaan. /2, 6, 10/



Kuva 6: Testaustasot

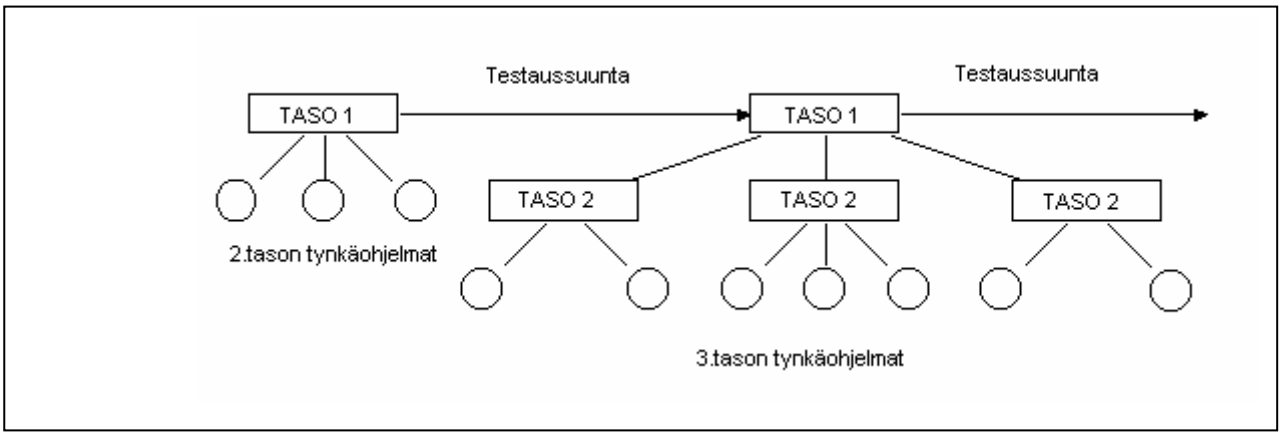
4.6.1 Moduulitestaus

Testattava järjestelmä koostuu monesta yksiköstä eli moduulista. Moduuli koostuu yleensä 100–1000 ohjelmarivistä. Moduulitestauksessa (*unit testing*) testataan näitä ohjelmistoprojektin ohjelmayksiköiden pienimpiä osasia, kuten funktioita ja niistä muodostettavia kokoelmia eli moduuleita. Joissain yhteyksissä testattavana ovat myös alijärjestelmät, luokat ja tai luokkaryppäistä tunnetut klusterit. Tavoitteena on löytää selviä virheitä sekä ristiriitoja yksiköiden ja moduulien määrittelyssä ja toiminnassa ja korjata ne. Tässä testausvaiheessa tarvitaan yleensä ajureita (*test driver*) ja tynkäohjelmia (*test stub*), koska tyypillisesti näin pienet ohjelmayksiköt eivät toimi itsenäisesti. Ajurit ja tynkäohjelmat korvaavat ne yksiköt, joita aliohjelma kutsuu. Edellä mainituista työkaluista käytetään usein myös nimeä testipedit (*test beds*). Moduulitestauksessa hyödynnetään lähes kokonaan lasilaatikkotestausmenetelmiä, joten mustalaatikkotestauksen osuus jää kovin pieneksi. Testauksessa keskitytään lähinnä yksikön sisältämien loogisten suorituspolkujen testaukseen. Tätä vaihetta pidetään yleensä koodausta tukevana testausvaiheena, joten näitä kahta työmenetelmää käytetään varsin usein rinnakkain. Moduulitestauksen suorittaa usein ohjelman kehittäjä itse. /2, 6, 8, 10/

4.6.2 Integrointitestaukset

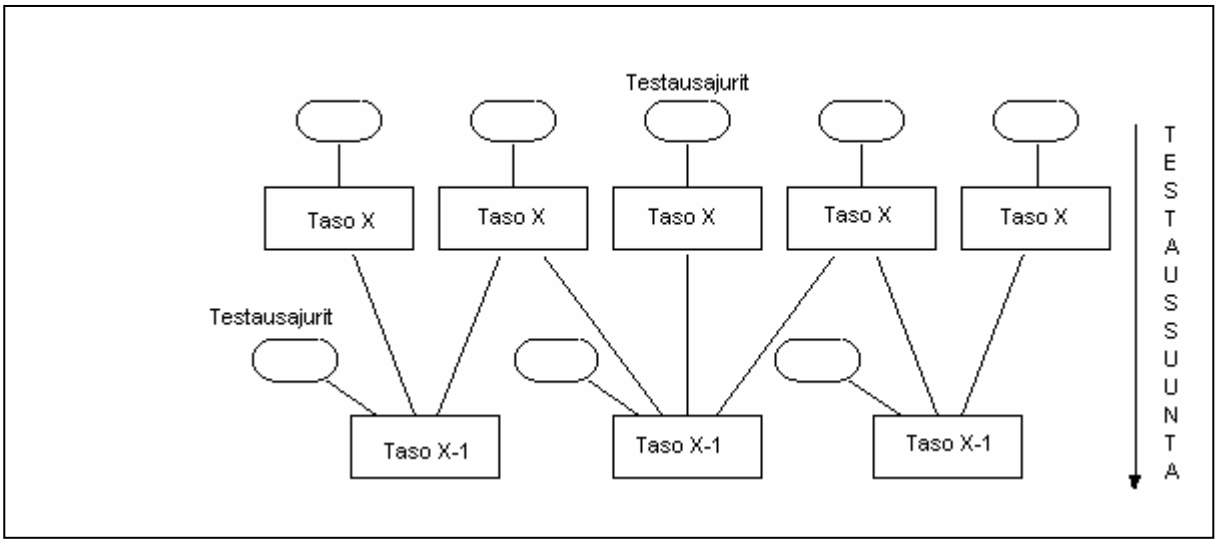
Kun moduulitestaus on saatu suoritettua, niin voidaan aloittaa integrointitestaus. Integrointitestauksella (*integration testing*) tarkoitetaan testausvaihetta, jossa moduuleita tai moduuliryhmiä kootaan yhteen ja testataan. Testaus jatkuu niin kauan, kunnes koko systeemin integrointitestaus on saatu tehdyksi. Tarkoituksena on varmistaa, että yhdistetyt komponentit toimivat spesifikaatioiden mukaan sekä samalla löytää ja korjata mahdollisimman paljon moduulien yhteensopivuusongelmia. Testaus on lähinnä eri komponenttirajapintojen testausta, sillä yksiköiden testaus on suoritettu jo moduulitestauksessa. Tarkoitus on myös testata, toimiiko tietojen ja kontrollien välitys rajapinnan yli moitteettomasti. Integraatiotestauksen avulla havaitaan, jos tietoa katoaa rajapinnoilla, jotkut yksiköt eivät sovellu toimimaan toisten yksiköiden kanssa, arkkitehtuuri ei toimi tai jokin muu syy aiheuttaa ongelmia ohjelmiston toiminnassa. Integrointivaiheessa hyödynnetään sekä musta- että lasilaatikkotestausmenetelmiä.

Integrointitestausvaiheessa voidaan hyödyntää kolmea eri testausmetodia. Ylhäältä alas-integroinnissa (*Top-down*) testaus aloitetaan ohjelmiston kontrollihierarkian korkeimmista tasoista ja korvataan samalla kutsutut yksiköt tynkäohjelmilla. Top-down-testaus on kuvattu kuvassa 7. Nämä tyngillä varustetut yksiköt testataan ja onnistuneen testauksen jälkeen siirrytään pykälää alemmaksi ohjelmistohierarkiassa testaamaan seuraavaa yksikköä. Edellisen portaan testauksessa käytetyt tyngät korvataan seuraavan tason yksiköillä. Integrointia jatketaan niin kauan, kunnes koko järjestelmä on integroitu yhteen ja testattu.



Kuva 7: Top-down-testaus

Alhaalta ylös (*Bottom-Up*)-integrointi aloittaa testauksen ohjelmistohierarkian alimmista palasista, jotka kiedotaan yhteen pala palalta ylempien komponenttien kanssa. Tällaisia yhteenmuodostumia kutsutaan klustereiksi. Tämän tyyppisessä testauksessa käytetään hyväksi ajureita, jotka korvaavat väliaikaisesti ylemmät komponentit. Bottom-Up-testaus on nähtävillä kuvassa 8.



Kuva 8: Bottom-Up-testaus

Kertarysäys (*Big Bang*)-integrointi ei ole varsinaisesti integrointimetodi, mutta on silti varsin usein käytössä yksiköiden integroinnissa. Tämän tyyppisessä integrointimetodissa kerätään kaikki mahdolliset ohjelmiston komponentit yhteen ja integrointitestaus aloitetaan edellyttäen, että kokonaisuus saadaan toimimaan. Tässä menetelmässä ei siis tarvita tynkiä ja ajureita. /2, 6, 10 /

4.6.3 Järjestelmätestaus

Järjestelmätestauksessa (*system testing*) pyritään selvittämään, täyttääkö valmis systeemi sille asetetut vaatimukset sille asetetussa ympäristössä. Vaatimukset määritellään määrittely- tai asiakasdokumentaatiossa. Tarkoituksena on myös korjata ja selvittää ristiriitaisuudet systeemin ja sitä koskevan määrittelyn väliltä. Testattava systeemi voi koostua esimerkiksi ohjelmasta, laitteistosta, tietokannasta, tietoliikenteestä tai ulkoisista liittymistä, mutta olennaista on se, että testauksessa on koko järjestelmä. Tässä testauksen vaiheessa tutkitaan myös hyvin useasti ohjelman suorituskykyä, rasituskestävyyttä, tietoturvallisuutta, toipumista, verkkotoimivuutta ja yleistä käytettävyyttä systeemissä. Järjestelmätestaukseen voidaan myös usein liittää kenttätestaussuoritteet. Tyypilliset virheet, joita tässä vaiheessa löydetään, liittyvät heikkoon yleissuorituskykyyn, alhaiseen läpimenokapasiteettiin, riittämättömään toipumiseen ulkoisista poikkeustilanteista tai tietoturvaan liittyviin ongelmiin. Tämän testaustason huono puoli on se, että testiympäristön luonti vie paljon elintärkeitä resursseja projektilta.

Järjestelmätestaus olisi hyödyllisintä ulkoistaa, jotta systeemin sisäiset yksityiskohdat olisivat mahdollisimman vähäisessä tiedossa testaajalla. Järjestelmätestaus suoritetaan siis lähes yksinomaan mustalaatikkotestauksen keinoin. /2, 6, 8, 10/

4.6.4 Alfa-testaus

Ensimmäinen hyväksymistestaustapa on Alfa-testaus (*Acceptance / Alpha-testing*). Alfa-testauksessa tarkastetaan, täyttääkö esimerkiksi mittatilaustyönä tehty ohjelmisto asiakkaan asettamat vaatimukset tuotteen käyttöön tarkoitetussa ympäristössä. Alfa-testaus jatkuu, kunnes molemmat osapuolet ovat tyytyväisiä tuotteen ominaisuuksiin. Alfa-testauksessa testaus ja kehittely tapahtuvat tuotteen toimittajan tiloissa. Testauksen aikana ohjelmiston kehittäjä merkitsee muistiin asioita ja toimenpiteitä, joita testaaja suorittaa testausprosessin aikana. Kaikki ohjelman epänormaalit toiminnat merkitään muistiin, jotta ohjelman kehittäjä voi korjata ne testausprosessin jälkeen. Alfa-testaus on kokonaan mustalaatikkotestausta, koska testaus tapahtuu

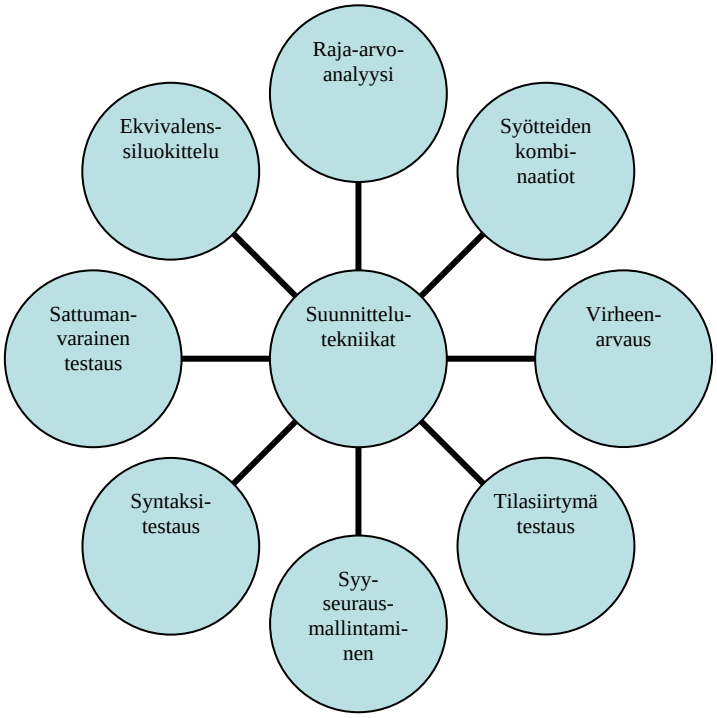
loppukäyttäjän näkökulmasta ja näin ollen ohjelman rakenteeseen ei ole sallittua tutustua. /2, 6, 10, 12/

4.6.5 Beta-testaus

Toinen hyväksymistestaustapa on Beta-testaus (*Product- / Beta- testing*). Beta-testauksessa jaetaan kaupallinen tuote mahdollisille asiakkaille tai joskus kaikille halukkaille ennen kyseisen tuotteen julkaisua. Asiakkaat käyttävät ja testaavat tuotetta omissa tiloissaan ilman ohjelmiston kehittäjää ja raportoivat mahdollisista virheistä ja parannusehdotuksista tuotteen kehittäjälle. Beta-testaajien valinta on olennaista, sillä kyseiseen ryhmään tulee kuulua juuri V-mallin tälle tasolle erikoistuneet testaajat. Beta-testauksen jälkeen ohjelmaa muokataan tarpeen mukaan. Tämän jälkeen julkaistaan uusi beta-versio tai lopullinen kaupallinen versio. Olennaisinta Beta-testauksissa on löytää kokoonpano- ja yhteensopivuusongelmia. Beta-testauksessa asiakkaat suorittavat testauksen mustalaatikkotestauksen teesien mukaan. /2, 6, 10, 12/

5 Mustalaatikkotestauksen suunnittelutekniikoita

Normaalisti puhuttaessa testauksesta tarkoitetaan juuri toiminnallisuuteen perustuvia menetelmiä. Ohjelman täydellinen testaaminen eli kaikkien toiminnallisuuksiin perustuvien ohjelman osasten testaus on käytännössä mahdotonta. Tällöin pitäisi verifioida kaikki mahdolliset syötteet ja tulokset. Kyseisiä vaihtoehtoja voi olla pahimmassa tapauksessa ääretön määrä. Taloudellisesti ja ajallisesti kannattavaa on siis päättää tietty määrä syötteitä, joilla mustalaatikkotestausta suoritetaan. Yleisimmin syöteavaruus jaetaan tiettyihin osajoukkoihin, joita sitten hyödynnetään testattavien arvojen valinnassa. Kyseisiä menetelmiä ovat muun muassa ekvivalenssiluokittelu (*equivalence partitioning*) sekä raja-arvoanalyysi (*boundary value analysis*), jotka molemmat perustuvat mahdollisten syötteiden jakoon osajoukkoihin, joissa jokaisen syötteen oletetaan käyttäytyvän samalla tavoin. Näin ollen jokaisen osajoukon yhden syötteen testattua voidaan todeta koko osajoukon arvojen vastaavan samaa lopputulosta ja toimivan näin korrektisti. Suurin hyöty on siis se, että syöteavaruus voidaan testata huomattavasti pienemmällä testitapausten määrällä. Kriittisesti voidaan kuitenkin ajatella, että käytännössä on mahdoton tietää sitä, että käsitteleekö ohjelma samaan tapaan kaikki tietyn osajoukon syötteistä. Tästä syystä luokittelu on aina testaajan kokemukseen ja näkemykseen perustuva käsitys ohjelman toiminnasta erilaisilla syötteillä ja testaajan onkin syytä tehdä tietynlaisia pistokokeita syötteillä, jotka sijoittuvat raja-arvojen sisälle. Tarpeen mukaan testaaja voi joutua muuttamaan raja-arvoja. Kaikki tässä kappaleessa mainitut mustalaatikkotestauksen suunnittelutekniikat on eritelty ohjelmistokomponenttien testausstandardissa BS7925-2. Kyseiset suunnittelutekniikat on havainnollistettu kuvassa 9. /2/



Kuva 9 : Mustalaatikkotestauksen suunnittelutavat

5.1 Ekvivalenssiluokittelu

Mustalaatikkotestauksessa testitapausten valinta perustuu syöteavaruuden jakamiseen samaan tapaan toimiviin ekvivalenssiluokkiin. Oletuksena on, että jos ohjelma yhdellä ekvivalenssiluokan edustajalla, niin se toimii myös muilla saman luokan edustajilla. Testauksessa otetaan siis vähintään yksi arvo jokaisen luokan arvoista. Tämän jälkeen käydään kaikki luokat läpi mahdollisimman järjestelmällisesti. Tällaista testiaineiston valintaa sanotaan ekvivalenssiluokitteluksi.

Esimerkkinä esittelyssä on epävirallinen ihmisten ikäluokkien jaottelu. Siinä jaetaan ihmiset iän mukaan 11 ryhmään. Nimesin jokaisen ryhmän syötteen mukaisesti ja määrittelin jokaisen ryhmän ala- ja ylärajan. Taulukon oikean puolimmaisimmassa sarakkeessa on esitelty jokaisesta luokasta esimerkkitapaus, joka edustaa koko luokan ulostuloa. Taulukko 1 esittelee ekvivalenssiluokittelun ihmisen eri ikäluokista. Tämän tyyppisen jaottelun tuloksena voitaisiin kehitellä ohjelma, joka voisi kysyä käyttäjältä syötteenä lukua, joka vastaa ihmisen ikävuosia. Ohjelma tulostaisi silloin vastaavan luokan nimen käyttöjärjestelmään.

Jokaisesta luokasta on esiteltynä yksi testitapaus. Jokaisella testitapauksella on siis yhtä hyvä todennäköisyys paljastaa virhe ohjelmassa. Tärkeätä on valita myös epäkelvollisia syötteitä, kuten edellä mainittu negatiivinen arvo, tyhjä syöte, kirjaimet ja erikoismerkit. Ekvivalenssiositus minimoi tehokkaasti testitapauksia.

Taulukko 1: Ekvivalenssiluokittelu ihmisen ikäluokista

Luokan nimi	Syötteen alaraja	Syötteen yläraja	Esimerkki testitapauksesta
Epäkelvot syötteet	- ∞	- 1	-10
Vastasyntynyt	0	1	1
Tarhaikäinen	2	6	4
Ala-asteikäinen	7	13	10
Yläasteikäinen	14	16	15
Lukioikäinen	17	19	18
Korkeakouluikäinen	20	25	22
Nuoret aikuiset	26	34	30
Keski-ikäiset	35	50	38
Ikääntyneet	51	64	55
Eläkeläiset	65	∞	80

Syöteavaruuden osittamiseen perustuvan ajattelutavan periaatteellinen ongelma on osituksen oikeellisuus. Testaajan mielestä yhtenäiseltä näyttävä ekvivalenssiluokka saattaakin todellisuudessa koostua useammista luokista. Esimerkkiohjelmankin tapauksessa olisi edellä esitetty ositus todennäköisesti liian karkea. On esimerkiksi mahdollista, että laskenta tapahtuu eri tavalla, kun lopputulos ei mahdu 16 bitin kokonaislukuun. ”Normaalien” tapausten ekvivalenssiluokka jakautuisi näin ainakin kahteen osaan, ja testiaineistoon on liitettävä uuden luokan tyypillinen tapaus ja luokkien rajoilla olevia tapauksia. Joskus on mahdollista myös, että virheellinen syöte aiheuttaa virheen ulostulossa, jonka jälkeen lopuilla syötteillä ei jatketa enää testaamista. /8, 13/

5.2 Raja-arvoanalyysi

Ekvivalenssiluokittelu ja raja-arvoanalyysi liittyvät olennaisesti toisiinsa. Ekvivalenssiluokittelussa syöteavaruus jaetaan eri luokkiin, joilla jokaisella on omat ylä- ja alarajansa. Raja-arvoanalyysissä testisyötteet valitaan ekvivalenssiluokittelun mukaisilta raja-arvoalueilta. Syy siihen, miksi valinta kohdistuu juuri rajoilla oleviin syötteisiin johtuu siitä, että mahdolliset ongelmakohdat osuvat juuri näille edellä mainituille raja-arvokohdille paljon useammin kuin esimerkiksi luokan keskivälin arvolle. Tässä analyysissä valitaan tämän lisäksi usein virheitä aiheuttavia arvoja kuten 0. Raja-arvoanalyysillä testaus saadaan kohdistettua usein virheitä aiheuttaviin kohtiin ja näin testauksen minimointia saadaan hyödynnettyä.

Tyypillinen ohjelmoijan virhe voisikin olla vaikkapa seuraavanlainen tapauksessa, jossa kysytään tietyn kuukauden päivämäärän järjestyslukua:

```
if (date >= 0 && date < 32)
```

Tässä tapauksessa ohjelmoija on valinnut vahingossa myös alaraja-arvon 0 mukaan, joka olisi helposti paljastunut raja-arvoanalyysin periaatteiden mukaan. Oikeata koodaustapaa olisi ollut tietysti:

```
if (date > 0 && date < 32)
```

Raja-arvoanalyysissä kiinnitetään huomiota myös tulosteisiin syötteiden lisäksi. Taulukossa 2 on esitelty raja-arvoanalyysin mukaiset raja-arvosyötteet kohdan 5.1. mukaisen esimerkkitapauksen perusteella. Testitapaukset valitaan siis siten, että valitaan ala- tai ylärajasyöte ja lisätään/vähennetään tuosta raja-arvosta/raja-arvoon yksi. /8, 13/

Taulukko 2: Raja-arvoanalyysin mukaiset raja-arvot esimerkkitapauksessa

Luokan nimi	Raja-arvot
Epäkelvot syötteet	-2, -1
Vastasyntynyt	0,1
Tarhaikäinen	2,3,5,6
Ala-asteikäinen	7,8,12,13
Yläasteikäinen	14,15,16
Lukioikäinen	17,18,19
Korkeakouluikäinen	20,21,24,25
Nuoret aikuiset	26,27,33,34
Keski-ikäiset	35,36,49,50
Ikääntyneet	51,52,63,64
Eläkeläiset	65,66

5.3 Syötteiden kombinaatiot

Näkökohdaksi täytyy ottaa myös se, että yksityiset syötteet saattavat olla kelvollisia, mutta näiden syötteiden tietyt kombinaatiot voivat aiheuttaa virhetilanteen. Esimerkkinä voisi olla ohjelma, joka kysyy kahta kokonaislukua syötteenä ja tulostaa näiden tuloksen. Tällaisessa tapauksessa lopputuloksena saattaisi olla ylivuoto, koska ohjelmalle on suunniteltu joku tietty yläraja syötteen suuruudelle. Ongelmia saattaa aiheuttaa myös tilanne, jossa useampi käyttää jaettua ohjelmaa samaan aikaan. Tällöin tuloksena saattaa olla tiedon eheyden menettäminen tai ohjelman totaalinen ”jäätyminen” tai lukittuminen. Kombinaatioiden testaus kuuluu usein osaksi rasitustestausta. Siinä vikatilanne yleensä aiheutuu liian suuren syöttömäärän myötä. Paljon syötteitä syötettäessä saattaa systeemi kaatua jo sallitulla syötemäärällä. Tämän tyyppiset testaukset on hyvä testata erilaisten rasitus- ja luotettavuustestien teesien mukaisesti. /13/

5.4 Virheenarvaus

Virheenarvausmetodissa luotetaan luontaisiin kykyihin, testauskokemukseen ja sovellusaluetuntemuksiin. Syötearvaukset eivät välttämättä kuulu mihinkään

ekvivalenssiluokkaan eivätkä raja-arvoihin. Testauskokemuksen ansiosta testaajalla on hallussaan esimerkiksi vanhoja testiraportteja ja asiakasreklamaatioita, joiden perusteella on helppo miettiä tulevaisuuden testitapauksia eli toisin sanoen arvata virheitä.

Yleisimmät virhesyötteet lienevät seuraavankaltaisia:

- Syötteenä tai tuloksena nolla
- Ei yhtään syötettä tai tulostetta
- Negatiivinen syöte
- Liian vähän tai liian paljon syötteitä.
- Numero syötteenä kirjaimia vaativassa syötekentässä

Uutena testaajana virheenarvausmenetelmän käyttö ei ole niin hyödyllistä, kuin kokeneella testaajalla, joka osaa ennakoida mahdollisia virheitä. /1/

5.5 Tilasiirtymätestaus

Tilasiirtymätestaus (*state transition testing*) poikkeaa hieman muista mustalaatikkotestausnäkökulmista, sillä tässä testaustyypissä on käsittelyssä sovelluksen tilat. Tämä testaus soveltuu erityisesti niille menetelmille, joiden vaatimuksetkin on jo määritelty tilakaavioina. Tilasiirtymätestausta käytetään silloin, kun testattava systeemi käsitetään äärelliseksi tilakoneeksi. Se tarkoittaa sitä, että systeemillä voi olla ainoastaan äärellinen määrä erilaisia tiloja ja siirtymät yhdestä tilasta toiseen on määritelty tilakoneen sääntöjen mukaisesti. Tämän kaltaiseen malliin perustuu koko systeemi ja testitapaukset. Äärellisen tilasysteemin perusperiaatteeseen kuuluu se, että systeemillä voi olla erilainen tulos, vaikka systeemille syötettäisiinkin samanlainen sisäänmenon syöte. Tämä johtuu siitä, että systeemin tila on muuttunut edellisen sisäänmenon jälkeen.

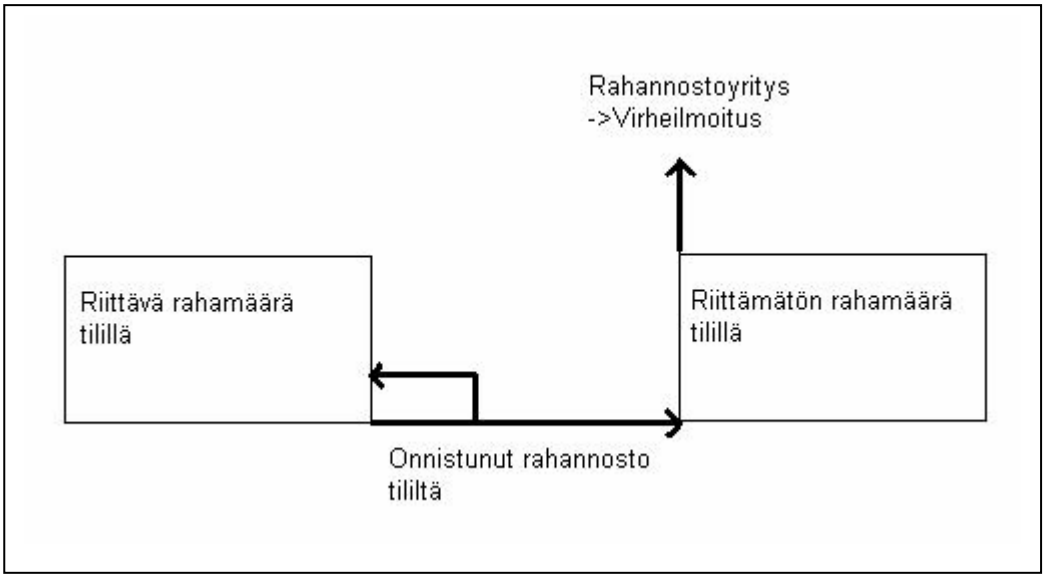
Esimerkiksi, jos haluat nostaa 100€ pankkiautomaatista, sinulle voidaan antaa rahaa yhdessä tapauksessa. Myöhemmin voit myös yrittää nostaa samaisen summan automaatilta, mutta tällä kertaa automaatti voi hylätä vaatimuksesi. Syynä voisi olla esimerkiksi se, että tilillä ei ole tarvittavaa määrää rahaa saatavissa kyseisellä hetkellä.

Hylkäys johtuu siis siitä, että pankkitilisi tila on muuttunut edellisestä nostokerran vuoksi. Tilannetta on kuvattu tilasiirtymäkaaviossa kuvassa 10. Tilanmuutosmallilla on neljä perusosasta:

- Tilat, joihin ohjelma voi asettua (Riittävä/riittämätön määrä rahaa)
- Siirtymät tilalta toiselle (Kaikki siirtymät eivät ole sallittuja)
- Tapahtumat, jotka aiheuttavat siirtymän (rahan nosto tililtä)
- Tapahtumat, jotka johtuvat siirtymästä (virheilmoitus / rahan myöntäminen)

Pitää myös huomata, että siirtymässä ei tarvitse siirtyä toiseen tilaan, vaan siirtymän jälkeen voidaan jäädä samaankin tilaan. Testitapausten muodostus tilanmuutosmallin pohjalta on selvästi mustalaatikkotestauspohjaista. Testien kattavuusalueen laskennalla taas viitataan lasilaatikkotestaukseen. Kaiken kaikkiaan tilanmuutosmallien pohjalta tapahtuva testaus käsitetään kuitenkin mustalaatikkotestaustekniikaksi.

Testauksessa testataan pääsy eri tiloihin sekä tyypillisimmät polut peruskäytössä. Spesiaalisimmissa tapauksissa katsotaan myös vähemmän käytetyt polut, jotka olisivat mahdollisesti jääneet kehittäjältä testaamatta. Myös systeemin palautuminen virhetilanteista sekä negatiivisista testitapauksista tulisi testata mahdollisimman huolellisesti. Negatiivisilla tapauksilla pyritään testien epäonnistumisiin käyttämällä laittomia tiloja ja siirtymiä niiden välillä. /1/



Kuva 10: Tilasiirtymäkaavio rahan nostosta

5.6 Syy-seuraus mallintaminen

Ohjelmistotestauksessa syy-seuraus-mallintaminen (*cause-effect graphing*) on kaavio, joka on varustettu syitä ja seurauksia kuvaavilla nuolilla ja tiloilla. Tämä kaavio siis esittää graafisesti syille seuraukset. Syitä voidaan ajatella ohjelman sisäänmenoiksi ja seurauksia ohjelman ulostuloiksi. Yleensä mallinnuksessa on esitetty syyt vasemmalla puolella ja seuraukset oikealla puolella kuvaa. Ohjelman rajoitukset voidaan lisätä syiden ja seurausten lomaan. Ne esitetään rajoitussymbolein käyttäen katkoviivaa.

Kuvion suunta on seuraavanlainen:
Syyt - - > välissä olevat solmukohdat - - > Seuraukset

Malli voidaan aina uudelleenjärjestää siten, että vain yksi solmu on sisäänmenon ja ulostulon välillä. Syy-seurausmallinnus on hyödyllinen kevennetyn totuustaulun luomisen yhteydessä. /14/

5.7 Syntaksitestaus

Syntaksitestausta (*syntax testing*) suoritetaan lähinnä testauksen alatasoilla. Syntaksitestauksessa testataan syntakseja, joissa käsitellään eri elementtien vuorovaikutusta. Alemmilla testaustasoilla testataan pelkästään syntakseja, mutta ylemmillä V-mallin testaustasoilla testataan sekä syntakseja että testattavan ohjelmiston logiikan toimivuutta. /9/

5.8 Sattumanvarainen testaus

Kannattavaa on tehdä myös pistokokeita. Eli tarkoituksena on arvata virheitä summittain ilman mitään logiikkaa. Sattumanvaraisessa testauksessa (*random testing*) valitaan syöteavaruudesta arvoja täysin umpimähkään, joilla suoritetaan testiajot. Tämä tapa ei ole paras mahdollinen, koska tämän avulla ei saada varmuutta siitä, että tuleeko kohde testattua tarpeeksi hyvin. Tällä tekniikalla ei myöskään opita tulevaisuutta varten mitään uutta, sillä sattumanvaraisen testauksen pohjalta ei voida muodostaa mitään ohjenuoria tulevaisuuden testeille. /13/

6 KÄYTÄNNÖN MUSTALAATIKKOTESTAUSTAPOJA

Seuraavassa esitellään käytännön mustalaatikkotestaustapoja, joihin olen päässyt tutustumaan työurani aikana erityyppisissä testausprojekteissa. Käsittelyssä ovat funktionaalinen testaus (*functional testing*), ulkomaan verkossa tapahtuva kenttätestaus(*field testing*) sekä UI-testaus (*User Interface-testing*)

6.1 Funktionaalinen testaus

Funktionaalisen testauksen tarkoituksena on todeta se, että testattava ohjelmisto toimii spesifikaatioden, tuotekuvausten ja käyttöohjeiden mukaisesti. Kaikkien ohjelmistoon kuuluvien ominaisuuksien tulisi esiintyä puhelimessa ja toimia moitteettomasti. Funktionaalinen testaus tulisi aina tehdä ulkopuolisen testaajan toimesta, joka yrittää omaksua loppuasiakkaan näkökulman asiaan. Hyvin usein funktionaalista testausta pystytään myös automatisoimaan. Funktionaaliseen testaukseen kuuluu myös jossain määrin kuormitustestausta, jossa jäljitellään asiakkaan ohjelmiston toimintaympäristöä mahdollisimman tarkasti ja pyritään samalla selvittämään se, että selviääkö ohjelmisto kovasti tehoa ja resursseja vaativista vaatimuksista. Testausta suoritetaan raskailla syötemäärillä ja kuormituksilla. Ohjelmiston tehokkuus on muutenkin tarkkailussa tämän tyyppisissä testauksissa. Testauksen tarkoituksena on tarkastaa se, että tekeekö ohjelmisto asiat riittävän tehokkaasti esimerkiksi kokonaisajan, maksimivasteaikojen ja muistitilan käytön näkökulmista. Tietyissä mobiilipäätelaitetestauksissa funktionaalisessa testauksessa tarkastetaan myös edellisellä ohjelmistoversiolla löytyneet virheet ja verifioidaan kyseisten virheiden esiintyminen nykyisellä ohjelmistoversiolla.

6.2 Kenttätestaus

Kenttätestauksessa on tarkoitus testata mobiilipäätelaitteen toimivuutta ulko- tai kotimaan matkapuhelinverkossa eri operaattorien SIM-korteilla (*Subscriber Identity Module*). Testitapaukset jaetaan stationäärisiin ja liikkuviin testitapauksiin (*mobile test cases*) sekä heikossa kentässä (*weak field testing*) ja verkkovierailua (*roaming*)

hyödyntäviin testitapauksiin. Testaus tapahtuu siis karkeasti sanottuna mobiilipäätelaitteen käytön todellisessa ympäristössä ja todennäköisien käyttötarkoituksien mukaisesti. Testaajina toimivat ohjelmiston kehityksestä riippumattomat testaajat, joiden tarkastelunäkökulmana on loppukäyttäjän puhelimen peruskäyttöön kuuluva näkökulma.

Kenttätestauksen voidaan sanoa olevan järjestelmätestausta. Siinä siis testataan mobiilipäätelaitetta kokonaisuutena, joskin tiettyihin osa-alueisiin keskitytään toisia paremmin. Näitä osa-alueita ovat erityisesti kaikki pakettidatan toimintaan liittyvät toimenpiteet ja näitä käyttävien applikaatioiden yhteiskäyttö. Voidaankin siis sanoa, että testauksessa on ohjelmiston lisäksi laitteisto ja järjestelmän kanssa yhteistyössä toimivat ulkoiset järjestelmät. Kenttätestauksessa vaaditaan, että järjestelmälle asetetut toiminnalliset ja ei-toiminnalliset vaatimukset sekä käyttötapaukset on esitelty mahdollisimman selkeästi.

Stationäärisiä testitapauksia suoritetaan testauskohteessa useimmiten laboratorio-olosuhteissa joko asiakkaan omissa työtiloissa tai hätätapauksessa vaikkapa hotellihuoneessa. Testitapaukset liittyvät usein funktionaaliseen testaukseen, jossa testataan päätelaitteen toimivuutta testauskohteen verkossa. Testitapauksiin ei siis koskaan kuulu applikaatiot, jotka toimivat offline-tilassa. Usein tapana on tehdä myös stressitestausta eli eri toimintojen pitkän ajan tai monen toiston testausta.

Liikkuvat testit on yksinkertaisinta suorittaa liikkuvan kulkuneuvon kuten auton kyydissä. Liikkuvissa testeissä testataan myös samankaltaisia asioita kuin stationäärisissä testeissä, mutta testeihin liitetään *Location arean* ja järjestelmien välisten kanavanvaihtojen (*Inter System Handover*) muutoksia. Location arealle ei varsinaisesti ole suomennosta, mutta se tarkoittaa pienintä yksikköä, josta alueelliset matkapuhelinkeskukset pitävät kirjaa. Näin pyritään selvittämään toimiiko puhelin moitteettomasti tällaisilla ongelma-alueilla. Autolla liikuttaessa näille muutosalueille on helppo navigoida ja näin tehdä useita toistoja testitapauksille. Edellä mainittujen alueiden muutoksia on helppo seurata kenttätestinäyttöä hyödyntäen.

Heikon kentän testauksissa testataan puhelimen toimimista alueilla, joissa signaalia ei ole saatavilla. Verkon toimimiseen liittyvät testaukset verifioidaan menemällä joissakin tapauksissa signaalittomaan paikkaan ja toisissa tapauksissa tulemalla kyseiseltä alueelta pois. Näiden siirtymisosuuksien aikana suoritetaan verkon toimintaan liittyviä toimintoja.

Verkkovierailua koskevat testitapaukset suoritetaan ”vieraan” maan SIM-kortilla. Testauksessa voi olla muun muassa soitonsiirtoihin ja soitonestoihin liittyvät suoritteet verkkovierailua koskien.

Kenttätestauksessa pyritään jäljittämään virheiden syyt erityisen Trace-ohjelman avulla. Trace-ohjelma tallentaa kaikki verkolle lähetetyt viestit erityiseen loki-tiedostoon ja näin ollen lokia tutkimalla ohjelmiston suunnitteluryhmä saa helpommin selvitettyä ongelman syyn ohjelman toimintojen virhetapauksissa.

6.3 UI-testaus

UI-testauksessa on tavoitteena testata puhelimeen ohjelmoitua asiakkaan määrittelemää asetuspakettia. Puhelimen toiminnallisuuteen liittyviä testitapauksia ei suoriteta kovinkaan paljoa tämän tyyppisessä testauksessa. Voidaan myös helposti todeta, että UI-testaus on mustalaatikkotestausta, koska testaajan ei tarvitse kiinnittää huomiota muun muassa ohjelmistossa käytettyyn ohjelmointikieleen. Testaus tapahtuu testitapausten pohjalta, jotka on suunniteltu eri päätelaitteiden toiminnallisuuksien pohjalta.

Testattavat asetuspaketit on määritelty operaattorin toimesta mieleisikseen. Paketti koostuu viidestä eri kategoriasta, jotka sisältävät tarkemmin aihepiiriin liittyviä asetuksia. Kategoriat ovat tuoteprofiiliin liittyvät asetukset, yhteyspisteisiin liittyvät asetukset, testattavan paketin grafiikkaan liittyvät asetukset, selaimen kirjanmerkkiasetukset ja muut asetukset.

7 MOBIILIPÄÄTELAITTEEN APPLIKAATIOIDEN MUSTALAATIKKOTESTAUKSEN SUORITTAMINEN

Seuraavassa kappaleessa otetaan käsittelyyn mobiilipäätelaitteen 10 tyypillistä applikaatiota. Jatkossa pyritään selvittämään sen, millä eri tavoin kyseistä applikaatiota testataan eri mustalaatikkotestaustavoilla. Kyseiset mustalaatikkotestaustavat ovat tulleet erityisen tutuiksi minulle työvuosieni aikana erilaisten testausprojektien yhteydessä. Nuo 10 applikaatiota, joita tulen käsittelemään ovat tekstiviesti-applikaatio, multimediaviestiapplikaatio, galleria, web-selain, osoiteluettelo, suoratoistoapplikaatio, äänipuheluapplikaatio, videopuheluapplikaatio, Bluetooth ja kelloapplikaatio. Otan esille eri applikaatioiden mustalaatikkotestausmuodoissa tyypillisimpiä testitapauksia, joita voi hyödyntää kaikkien mobiilipäätelaitevalmistajien mustalaatikkotestauksissa niin funktionaaliseen testaukseen, kenttätestaukseen ja UI-testaukseen liittyvissä projekteissa. Tarkoituksenani ei siis suinkaan ole esittää kaikkia mahdollisia testitapauksia kyseisiin applikaatioihin liittyen, sillä teoriassa mustalaatikkotestitapausten määrä on ääretön ja riippuu lähinnä testien luojan näkökulman laajuudesta.

7.1 SMS

Tekstiviestit eli SMS:t (*Short Message Service*) ovat 160 merkin pituisia viestejä, jotka lähetetään tyypillisesti mobiilipäätelaitteesta toiseen. Nykyään viestejä voidaan ketjuttaa moneen 160 merkin osaseen. Viestit lähetetään tekstiviestikeskukseen, jossa ne odottavat vastaanottavan päätelaitteen tuloa verkkoon. Viestit pyritään lähettämään saman tien, kun kohdepuhelin on valmis vastaanottamaan viestejä. Tekstiviesteillä pystytään kommunikoimaan nopeasti ajasta ja paikasta riippumatta.

Idea tekstiviestistä sai alkunsa vuonna 1984 norjalaisen J.Audestadin johdolla, joka johti GSM WP3 – alatyöryhmää (*Global System for Mobile communications*). Ensimmäinen tekstiviesti uskotaan lähetetyn joulukuussa 1992 tietokoneelta Vodafonen GSM-verkkoon kuuluneeseen matkapuhelimeen. Suomessa tekstiviestit yleistyivät vuoden 1994 aikana, mutta alkuun ainoastaan puhelimeen päin. Vuonna 1995 lähetys onnistui myös toiseen suuntaan.

Uusi tekstiviestinnän trendi on vuorosuuntainen verkkokeskeinen tekstiviestintä Internetiin yhteydessä olevan ohjelmistoapplikaation ja matkapuhelimen välillä. /15/

7.1.1 SMS-applikaation mustalaatikkotestaus

Ehkä yleisin tapa testata tekstiviestien toimivuutta funktionaalisessa testauksessa on sen testaus samaan aikaan muiden toimintojen ja applikaatioiden kanssa. Tällainen toiminto voisi olla vaikkapa valinnainen internet-yhteys (*dial-up*), äänipuhelu, videopuhelu, MMS, netin selailu tai suoratoisto. Kaikki nämä testaustyypit on usein tapana suorittaa sekä 2G- että 3G-alueilla, joille molemmille on erilaiset testispesifikaatiot. Testit voivat vaihdella myös siten, että testataan joko SMS:n vastaanottoa tai lähetystä eri toimintokombinaatioilla. SMS:n ja jonkun toisen applikaation yhteisaikaisesta toiminnasta esimerkkinä voisi olla vaikkapa kameran käyttö ja SMS-viestin vastaanotto samaan aikaan. Myös vaikkapa muistikortin formatointi SMS-viestin vastaanoton yhteydessä voisi kuulua tähän kategoriaan. SMS-viestivalikossa löytyy myös monia variaatioita, joita on myös syytä testata funktionaalisen testauksen yhteydessä. Sellainen voisi olla esimerkiksi se, että vastaanotettuun SMS-viestiin vastataan ja lisätään vastauksen yhteydessä muutama muukin vastaanottaja vastaanottajalistaan. Sen jälkeen lähetetään kyseinen viesti. SMS-viestiä voidaan hyväksikäyttää muiden applikaatioiden testauksen yhteydessä. Testauksessa voi olla esimerkiksi kalenterimerkinnän lähetys SMS-viestinä. Joitakin triviaalitapauksiakin on SMS-testauksessa. Tällainen voisi olla esimerkiksi 0-luokan SMS:n lähetys ja vastaanotto. Tämä tarkoittaa viestiä, joka näkyy ruudulla ainoastaan kerran ja jonka jälkeen se poistetaan automaattisesti. Suorituskyvyn testausta SMS-viestien osalta testataan vaikkapa siten, että mitataan 160-merkkisen SMS-viestin avaamiseen kuluva aika ja verrataan aikaa spesifikaatioihin.

Kenttätestauksessakin voidaan testata edellä mainitun kaltaisia monen toiminnon toimintaa samanaikaisesti. Testaus voi tapahtua ulkomaidenkin verkossa stationäärisesti, mutta myös liikkeessä ja heikoissa kentissä. Liikkeessä on tärkeää huomata, että samat toiminnot toimivat Location Arean ja verkonvaihdon muuttumisen yhteydessä. Verkonvaihdolla tarkoitan siirtymistä 2G-alueelta 3G-alueelle tai toisinpäin. Muita asioita, joita testataan usein kenttätesteissä, ovat esimerkiksi

välitystietojen toimivuus SMS-viestin lähetyksen yhteydessä. Myös pitkitettyjen viestien eli yli 160 merkin viestien lähetys ja vastaanotto kuuluu usein testisetteihin. Heikon kentän testauksesta esimerkkinä voisi olla vaikkapa sellainen, että SMS-viesti luodaan heikon kentän alueella offline-tilassa, jonka jälkeen siirrytään vahvaan kenttään ja online-tilaan sekä tarkistetaan samalla, että onnistuuko SMS-viestin lähetys moitteettomasti.

UI-testauksessa voidaan esimerkiksi tarkastaa se, että ovatko välitystietoja ja lähetetyn/vastaanotetun viestin voimassaoloaikoja koskevat asetukset oikein ohjelmoitu. Samoin tarkastetaan se, onko lähetetyn viestin aikaleima oikea.

7.2 MMS

MMS (*Multimedia Messaging Service*) toimii tekstiviestin tavoin, mutta tekstin lisäksi MMS:ään voidaan lisätä myös multimediaobjekteja, kuten kuvia, ääntä, videota, esityskalvoja ja muotoiltua tekstiä. MMS toimii samassa ympäristössä kuin web-selaus puhelimella, suoratoisto, ja valinnaisen internet-yhteyden muodostus eli pakettikytkentäisten verkkopalveluiden (GPRS, *General Packet Radio Service*) yhteydessä. MMS:n standardoinnista ovat huolehtineet *3GPP, 3GPP2 ja Open Mobile Alliance (OMA)*. Mahdollisuus lähettää ja vastaanottaa MMS-viestejä sisältyy jokaiseen nykypäivän puhelinmalliin. /16/

7.2.1 MMS-applikaation mustalaatikkotestaus

MMS:n testitavat ovat lähes samanlaisia SMS:n testauksen kanssa. Tässäkin tapauksessa testataan MMS:n toimivuutta muiden toimintojen ja applikaatioiden kanssa. Näitä toimintoja voisivat olla vaikkapa dial-up, äänipuhelu, videopuhelu, SMS, netin selailu tai suoratoisto. Nämäkin testaustyypit on usein tapana suorittaa sekä 2G- että 3G-alueilla, joille molemmille on erilaiset testispesifikaatiot. Testit voivat vaihdella myös siten, että testataan joko MMS:n vastaanottoa tai lähetystä eri toimintokombinaatioilla. Itse MMS – applikaation toimivuustesteihin voi kuulua esimerkiksi MMS:n sisältämän presentaation vastaanotto ja sen avaaminen tai

vaikkapa SMS-viestiin vastaaminen MMS-viestillä, johon lisätään vieläpä uusi kuva kamera-applikaation avulla. Erikseen voidaan testata kuvia, ääntä, videota, esityskalvoja ja muotoiltua tekstiä sisältäviä MMS-viestien vastaanottoa ja lähetystä. Eri applikaation yhteistoiminta voi ulottua vaikkapa kameran ja MMS:n vastaanoton tai videoeditorin ja MMS:n lähetyksen yhteistoiminnan testaukseksi.

Kenttätestauksessakin voidaan testata edellä mainitun kaltaisia monen toiminnon toimintaa samanaikaisesti. Testaus voi tapahtua ulkomaidenkin verkossa stationäärisesti, mutta myös liikkeessä ja heikoissa kentissä. Liikkeessä on tärkeää huomata, että samat toiminnot toimivat Location Arean ja verkonvaihdon muuttumisen yhteydessä. Verkonvaihdolla tarkoitan siirtymistä 2G-alueelta 3G-alueelle tai toisinpäin. Kenttätestauksen triviaalitestauksiin kuuluu SIM-työkalupakin testaus, jonka avulla tilataan MMS-asetukset vieraan maan verkossa sijaitsevaan puhelimeen.

UI-testauksen olennaisin asia on tarkastaa, että MMS:n yhteysosoite on kohdillaan. Asetuksissa vaihtelee myös multimedian hakuun liittyvät asetukset, nimettömien MMS-viestien hyväksyminen sekä mainosten vastaanottoon ja välitystietoihin liittyvä halukkuus. Myös MMS-viestien voimassaoloaikaa voidaan säädellä, kuten myös MMS-viestissä käytettävää maksimikokoa. MMS-viestin luontitilaa koskevat asetukset on myös syytä tarkastaa testauksen yhteydessä.

7.3 Galleria

Galleria-applikaatio (*Gallery-application*) on portaali, jonka kautta käyttäjä pääsee käsiksi puhelimen muistiin tai muistikortille tallennettuihin kuva-, video-, musiikki- äänite- ja esitystiedostoihin. Sen kautta on helppo tehdä esimerkiksi soittolista puhelimessa sijaitsevista suosikkikappaleista. Vastaanotetut multimedialiitteet johdetaan aina Galleria-applikaatioon.

7.3.1 Galleria-applikaation mustalaatikkotestaus

Galleriaa koskevissa funktionaalisissa testauksissa testataan muun muassa musiikkikappaleista koostuvan soittolistan luomista ja muokkaamista. Tarkoituksena voi olla myös tarkastaa, että onko puhelimen muistissa olevan kuvan asettaminen

puhelimen taustakuvaksi mahdollista. Kuvaeditorin testaus kuuluu myös olennaisesti galleria-applikaation testaussuoritukseen. Kuvaeditorin kautta tarkastetaan, että onko puhelimen muistissa olevan kuvan muokkaus mahdollista. Toiminnallisuustestauksiin galleria-applikaation osalta kuuluvat esimerkiksi gallerian avaamiseen kuluva aika ja kymmenen standardin mukaisen kuvan poistamiseen kuluva aika.

Kenttätestauksessa galleria-applikaation testaukseen ei panosteta kovinkaan paljoa, sillä galleriassa ei ole kovinkaan paljoa osasia, joiden toiminta riippuu ulkomaisista verkoista, vaan lähes kaikki testit voidaan suorittaa kotimaassa stationäärisin testein. Esiasennettujen teema- , kuva- ja äänilinkkien hyödyntäminen galleria-applikaation kautta on silti hyvä tarkistaa kenttätestauksen yhteydessä.

UI-testauksessa tarkastetaan esiasennettujen kuvien, videoiden ja musiikkikappaleiden oikeellisuus. Tarkastuksen alla ovat myös muun muassa kuvan online-välityspalvelu ja kuvien tulostuspalveluiden toiminta.

7.4 WEB-selain

WEB-selain (*WEB-browser*) on paranneltu versio vanhoissa puhelinmalleissa käytetystä WAP-selaimesta (*Wireless Application Protocol-browser*), jolla pystyi selaamaan ainoastaan yksinkertaisia WAP-sivuja. Uusissa puhelimissa vakiona oleva WEB-selain perustuu avoimelle lähdekoodille muun muassa sivujen renderöimisen osalta. Renderöimisellä tarkoitetaan kuvan luomista mallista tietokoneohjelman avulla. Selaimen sivurakenne on uudelleenjärjestetty niin, että ne mahtuvat puhelimen ruutuun periaatteessa sellaisina, kuin ne näkyvät tietokoneen ruudullakin. Selauksen yhteydessä on kätevää käyttää kursoria, jonka avulla sivun vierittäminen eri suuntiin ja linkkien painallus onnistuu helposti. Joissain puhelinmalleissa on myös tarjolla *Mini Map*, jonka avulla sivuston sisällön saa pienennettyä esikatselukuvaksi ja jonka avulla eri kohdassa sijaitseviin paikkoihin on helppo suunnata. WEB-selaimeen voi lisätä omia kirjanmerkkejä ja surffailla myös vaikkapa salatuille sivuille. HSDPA-soluissa (*High Speed Downlink Packet Access-cells*) WEB-selain toimii huomattavasti kitkattomammin. /17/

7.4.1 WEB-selain-applikaation mustalaatikkotestaus

Tässäkin tapauksessa testataan WEB-selailun toimivuutta muiden toimintojen ja applikaatioiden kanssa. Näitä toimintoja voisivat olla vaikkapa dial-up, äänipuhelu, videopuhelu, SMS, MMS tai suoratoisto. Nämäkin testaustyypit on usein tapana suorittaa sekä 2G- että 3G-alueilla, joille molemmille on erilaiset testispesifikaatiot. Testit voivat vaihdella myös siten, että testataan joko MMS:n vastaanottoa tai lähetystä eri toimintokombinaatioilla. Testauksessa on myös esimerkiksi tapauksia, joissa varmistetaan se, että onnistuuko WEB- surffailu sivuilla, joissa soi jokin taustamusiikki. Vaihtoehtona voi olla myös uuden puhelinteeman lataus nettisivujen kautta ja sen jälkeen teeman asennus puhelimeen. Tarkoitus on myös selvittää usein, että toimiiko kirjanmerkkien kohdalla kaikki oikein. Kirjanmerkkejä saatetaan yrittää poistaa, tallentaa ja modifioida.

Kenttätestauksessakin voidaan testata edellä mainitun kaltaisia monen toiminnon toimintaa samanaikaisesti. Testaus voi tapahtua ulkomaidenkin verkossa stationäärisesti, mutta myös liikkeessä ja heikoissa kentissä. Liikkeessä on tärkeää huomata, että samat toiminnot toimivat Location Arean ja verkonvaihdon muuttumisen yhteydessä. Verkonvaihdolla tarkoitan siirtymistä 2G-alueelta 3G-alueelle tai toisinpäin. Kenttätestireissulla on tapana myös testata, että toimiiko paikallisen operaattorin kotisivujen selaus moitteettomasti. Pikaisesti testataan myös, että toimiiko WAP-pelit, chat-palvelut ja paikkatietopalvelut kyseisen maan verkossa mobiilipäätelaitteella. Tärkeätä on myös tarkastaa suojattujen sivujen toimivuus. Tällä tarkoitan sitä, että suojaussertifikaattien yksityiskohdat ja tarvittavat notifikaatiot näkyvät varmasti käyttäjälle.

WEB – selaimen UI-testauksessakin olennaisinta on tarkastaa, että yhteysosoite on varmasti oikea. Muita huomioonotettavia testauskohtia ovat aloitussivun osoitteen paikkansapitävyys, fontti- ja näytön koon oikeat arvot, selaimessa käytetyn oletuskoodauksen oikea tapa, automaattisten kirjanmerkkien olemassaolo kirjanmerkkilistalla, JAVA/ECMA – komentosarjan paikkansapitävyys ja suojausvaroituksien asetukset.

7.5 Osoiteluettelo

Osoiteluettelo (*Phone book*) on varsin käyttökelpoinen sovellus käyttäjille. Siihen voidaan lisätä tarvittavat tiedot halutusta henkilöstä. Näitä tietoja voi olla esimerkiksi puhelin-, sähköposti-, yritys-, osoite- ja faksitiedot. Tämän applikaation kautta on helppoa siirtää SIM:illä sijaitsevia tietoja puhelimesta puhelimeen.

7.5.1 Osoiteluettelo-applikaation mustalaatikkotestaus

Osoiteluettelon funktionaalisen testauksen testisettiin liittyy esimerkiksi uuden kontaktin luominen ja siihen liittyvät operaatiot, kuten lisäinformaation lisäys, poisto ja muokkaaminen. Yleisesti katsotaan myös, että viimeksi soitettujen listalla oleva puhelinnumero voidaan lisätä osoiteluetteloon ja lisätä sen yhteyteen tietoja soittajasta. Synkronointi on myös tärkeä osa osoiteluettelon toimintaperiaatteita. Synkronoimalla saadaan esimerkiksi kotitietokoneen ja puhelimen kontaktit identtisiksi. Synkronointi tapahtuu esimerkiksi bluetoothin tai USB-kaapelin (*Universal Serial Bus*) avulla.

Kenttätestauksen yhteydessä on hyvä tarkastaa vieraan maan operaattori-SIMillä sijaitsevien kiinteiden kontaktien toiminta ja niiden aktivointi. Samalla on hyvä onnistuuko SIMillä sijaitsevien kontaktien lisääminen puhelimesta puhelimeen ja toimiiko ilmoitukset ja kyselyt aktivoinnin yhteydessä oikein.

UI-testauksessa tarkastetaan ainoastaan, että ennalta määrätyt kontaktit sijaitsevat päätelaitteessa ja se, että kontakteilla on oikeat nimi-, numero-, osoite- ja sähköpostitiedot.

7.6 Suoratoisto-applikaatio

Suoratoistoa (*Streaming*) päästetään toistamaan mediasoittimen avulla, jonka merkki vaihtelee mobiilipäätelaitteesta riippuen. Suoratoisto tukee useimmiten mediatiedostotyyppejä, kuten *RealAudiota, RealVideota, MP3, MPEG-4, QuickTime ja Windows*-formaatteja. Soittimessa voidaan myös muun muassa luoda mediakirjastoja.

Suoratoisto-applikaation avulla on mahdollista toistaa mobiilipäätelaitteeseen tallennettuja videoleikkeitä tai suoratoistaa videoleikkeitä internetistä. Suoratoisto on tiedonsiirtotapa, jonka avulla voidaan toistaa musiikkia tai videota ennen kuin koko tiedosto on tullut perille päätelaitteeseen. /18/

7.6.1 Suoratoisto-applikaation mustalaatikkotestaus

Suoratoistonkin yhteydessä testataan applikaation toimintoa muiden pakettidataa hyödyntävien toimintojen kanssa. Näitä ovat dial-up, äänipuhelu, videopuhelu, SMS ja MMS. Suoratoisto-testeissä on usein tapana tehdä myös rasitustestejä, joissa avataan tietty suoratoistolinkki vaikkapa 10 kertaa ja jokaisella kerralla yhteyden täytyy avautua, toimia ja sulkeutua moiteettomasti.

Kenttätestaustapauksissa voidaan vaatia testaajan etsimään ulkomaisen operaattorin sivuilta suoratoistolinkki ja tarkistamaan toimiiko kyseinen linkki testattavalla mobiilipäätelaitteella. Liikkuvana testitapauksena on suoratoistotiedoston toistaminen liikkeessä eri verkonvaihtojen ja location area – muutosten yhteydessä. Kuva ja ääni pitää olla suoratoistopätkässä puhtaita ja tunnistettavia.

Suoratoisto – applikaatiossa määritellään videoon ja suoratoistoon liittyvät asetukset. Videoon liittyvissä asetuksissa käsitellään toiston jatkuvuuteen liittyvää seikkaa. Suoratoistoon liittyvissä asetuksissa käsitellään proksin olemassaoloa ja sen palvelimen osoitetta sekä portin numeroa. Suoratoiston oletusyhteysosoite on myös elintärkeää tarkistaa ja kuten myös yhteysajan pituus. UDP-porttien arvot voidaan myös määrätä tässä yhteydessä sekä kaistanleveydet GPRS-, EGPRS- (*Enhanced GPRS*), UMTS- (*Universal Mobile Telecommunications System*) ja WLAN (*Wireless Local Area Network*)-yhteyksille.

7.7 Äänipuhelu

Äänipuhelut (*Voice calls*) ovat alun perin olleet puhelimen primäärinen tarkoitus ja on se sitä vielä nykyäänkin, mutta enää puhelimen toiminta ei keskity niin olennaisesti äänipuheluiden ympärille. Puhelutiedoista voidaan nähdä viimeksi soitetut,

vastaamatta jääneet ja vastatut puhelut ja niiden soittoajat. Mahdollisuus on lähettää myös DTMF-viestejä (*Dual Tone Multi-Frequency*) puhelun aikana ja määritellä muun muassa soitonsiirtoja, puhelunestoja, koputuspalvelun käytöstä ja pikanumeroasetuksista eri puhelutyypeille.

7.7.1 Äänipuhelu-applikaation mustalaatikkotestaus

Voidaan sanoa rehellisesti, että funktionaalisen testauksen ja kenttätestauksen tärkeimmät asiat ovat varmistaa äänipuhelun toimivuus. Äänipuhelun testaukseen kuuluu olennaisena osana hätäpuheluiden testaus. Se testataan funktionaalisessa testauksessa tyypillisesti lukitulla SIMillä tai ilman SIMmiä. Perusteltua on myös testata äänipuheluihin liittyviä erillisominaisuuksia, kuten kaiuttimen toimintaa ja värinähälytyksen toimintaa puhelua vastaanotettaessa. Olennaista on myös katsoa edellisten puheluiden listaa ja tarkistaa sieltä, että viimeksi soitetut, vastaamatta jääneet ja vastatut puhelut näkyvät oikein ja soittoaika edellä mainittujen kohdalla on kunnossa. Edge-teknologiaa käyttävien puheluiden kohdalla on jotain erityistapauksia äänipuheluiden kohdalla. Yksi esimerkki on se, että Edge-puhelimien valikoissa ei saa esiintyä mitään videopuheluihin viittaavaa.

Kenttätestauksessa lähes jokainen testitapaus liittyy jollain tapaa äänipuheluihin. Testinä käytetään muun muassa pitkää äänipuhelua, jossa soitetaan vaikkapa yleisesti käytettyyn testinumeroon puolen tunnin ajan ja tarkastetaan, että puhelu toimii moitteettomasti. Äänipuheluidenkin yhteydessä tarkastetaan toimivuus Location Arean ja verkonvaihdon yhteydessä. Muut äänipuhelutestitapaukset liittyvät esimerkiksi puhelunestojen ja soitonsiirtojen toimivuuteen. Myös konferenssipuheluiden perustoiminnot käydään läpi kenttätestireissuilla. Konferenssipuheluissa puheluun osallistuu yli 2 osallistujaa. Hätäpuheluiden toimivuus ulkomaan hätäkeskuksiin testataan myös läpi. Testaus suoritetaan yleensä puhelin lukittuna tai PIN-koodin kyselyn aikana.

Äänipuhelun UI-testauksessa tarkastetaan, että puhelun jälkeiset tiedot ovat oikein ja se, että puhelusuoritteiden historia säilytetään oikea aika puhelimen muistissa.

7.8 Videopuhelu

Videopuhelut (*Video calls*)ovat tuoneet uuden ulottuvuuden puhelinviestintään. Videopuhelu voidaan suorittaa kahden videopuheluita tukevan puhelinmallin välillä, jos molemmat sijaitsevan 3G-verkon kattavuusalueella. Videopuheluissa siis kuullaan lähes reaaliaikaisesti molemmista suunnista ääntä ja liikkuvaa kuvaa. Erityisesti kuvan sujuvuus riippuu suuresti yhteyden laadusta.

7.8.1 Videopuheluapplikaation mustalaatikkotestaus

Videopuheluiden toimivuutta testataan samoin tavoin kuin kenttätestireissuilla, mutta lähinnä ainoastaan toimisto-olosuhteissa stationäärisissä testitapauksissa.

Kenttätestauksessa testataan videopuheluiden toimivuutta liikkuvissa testitapauksissa ja pitkän ajan videopuhelutestitapauksissa. Tarkoitus on myös tehdä monta videopuhelutoistoa ja varmistaa, että puhelun laatu on moitteeton.

UI-testauksessa testataan videokuvan laadun määrittelyn oikeellisuus. Kameraan yleisesti liittyviäkin tapauksia UI-testauksesta löytyy, kuten kameran laukaisuäänen olemassaolon tarkistus.

7.9 Bluetooth

Bluetooth-tekniikan (*Bluetooth-technology*) avulla voidaan ottaa yhteyttä muihin Bluetooth-laitteisiin, kuten yhteensopiviin PC-tietokoneisiin (*Personal Computer*) ja puhelimiin. Erityisen hyödyllinen Bluetooth-laite puheluiden kannalta on Bluetoothilla toimivan Handsfree-laitteen parittaminen puhelimen kanssa. Tiedonsiirtäminen on mahdollista 10 metrin kantamalla.

7.9.1 Bluetooth-applikaation mustalaatikkotestaus

Bluetoothia sovelletaan moneen asiaan funktionaalisessa testauksessa. Esimerkiksi musiikkileike tai käyntikortti voidaan lähettää toiseen Bluetooth-puhelimeen

Bluetoothin välityksellä. Samoin voidaan testata Bluetoothilla toimivan Handsfree-laitteen toimintaa tai Bluetoothilla toimivan printterin toimintoja. Stressitestausta harrastetaan Bluetoothin osalta siten, että kesken Bluetooth-lähetyksen yhteys katkaistaan ja katsotaan miten matkapuhelin toipuu kyseisestä operaatiosta. Mitään laitteen ”jäätymistä” tai väärää toimintoa ei sallita kyseisen stressitestauksen yhteydessä.

Bluetoothia ei testata kenttätestireissulla muuten kuin dial-uppiin liittyvissä testitapauksissa, jossa muodostetaan dial-up-yhteys Bluetoothia hyväksikäyttäen.

Bluetooth-applikaation liittyvissä UI-testauksissa määritellään Bluetoothin aktivointi, puhelimen näkyvyys muille käyttäjille Bluetoothin kautta sekä oman puhelimen Bluetooth-yhteyksiä varten muodostettu nimi.

7.10 Kello

Kello-applikaation (*Clock-application*) kautta pääsee säätelemään herätyskellon ja kellon yleisiä asetuksia. Tämän myötä kello-applikaatio on hyvin yleinen applikaatio jokaiselle käyttäjälle arkipäivän käytössä.

7.10.1 Kello-applikaation mustalaatikkotestaus

Funktionaalisessa testauksessa tarkastetaan vaikkapa se, että radion ollessa päällä herätyskellon hälytys soi oikeassa ajassa ja radio hiljenee väliaikaisesti.

Kenttätestauksessa voidaan tarkastaa, että toimiiko NITZ (*Network Identity & Time Zone*) oikealla tavalla vieraan maan verkossa. NITZ:in ollessa päällä kellon pitäisi asettua verkon mukaiseen aikaan ilman käyttäjän erillistä asettamista.

UI-testauksen asetuksissa määritellään ensinnäkin se, että onko NITZ aktiivinen vai ei. Määrittelyssä voi olla myös se, että onko kellon tyyppi analoginen vai digitaalinen ja myöskin herätyskellon hälytysääni voi olla päätettävissä.

8 YHTEENVETO

Tutkintotyöni käsittelee mustalaatikkotestauksen roolia nykypäivän ohjelmistotestauksessa. Työn tarkoituksena on toimia eräänlaisena oppimateriaalina mustalaatikkotestauksen parissa työskenteleville testausinsinööreille. Aiheeseen olen tutustunut vajaa kaksi vuotta tähän asti kestäneellä työurallani, joka on sisältänyt vain ja ainoastaan ohjelmistotestaukseen liittyviä työprojekteja. Työurani alkoi koulutukseeni liittyvällä harjoittelulla, joka tavallaan antoi alkusysäyksen opinnäytetyöni suunnitteluun.

Työni osoittaa sen, että mustalaatikkotestaus on varsin tehokasta siinä tapauksessa, kun testitapaukset ja – spesifikaatiot on määritelty tarpeeksi tehokkaasti. Testauksen johdonmukainen ja järkiperäinen suunnittelu ennen testausprojektia on osoittautunut varsin paljon tehokkaammaksi tavaksi suorittaa ohjelmistoa koskevat testit kuin sattumanvarainen testaus ilman mitään logiikkaa. Tiettyä testaustekniikkaa käyttäen virheitä pystytään löytämään hallitusti ja pienempiä resursseja käyttäen. Se takaa kestävän kehityksen ohjelmistoprojekteille ja helpottaa testauksen hallinnointia ja dokumentaatioiden luomista. Sukupolven vaihto testausinsinöörien kesken helpottuu myös huomattavasti, kun koko testaustiimillä on sama yhteinen työskentelytapa. Näin ollen kuka tahansa testaustiimin jäsenistä voi jakaa samat testauskokemuksensa uudelle tulokkaalle.

Työni osoittaa myös sen, että mustalaatikkotestaus ulottuu varsin monen tyyppisiin testausprojekteihin. Omat projektini ovat käsitelleet funktionaalista testausta, kenttätestausta ja UI-testausta, mutta varmasti olemassa on rutkasti samankaltaisia testauksen aloja, joissa työssäni mainittuja metodeja voidaan hyödyntää projektin tehokkuuden parantamiseksi.

Tutkintotyön tekeminen on edennyt johdonmukaisesti tiettyä punaista lankaa seuraillen. Työn tekeminen on ollut varsin mielenkiintoista, koska kyseessä on läheinen ja mielenkiintoinen aihe. Tästä vielä erikoiskiitokset Jussi-Pekka Peltoselle ja Flander Oy:lle.

LÄHDELUETTELO

1 ISEB Foundation Certificate in Software Testing. Kurssimateriaali. Grove Consultants. http://www.grove.co.uk

2 Yksikkö – ja integrointitestauksen menetelmät ja mittarit [www-sivu] [viitattu 8.10.2007] Saatavissa: http://ethesis.helsinki.fi/julkaisut/mat/tieto/pg/satukangas/yksikkoj.pdf

3 Regressiotestaus osana testausprosessia[www-sivu] [viitattu 8.10.2007] Saatavissa: http://www.cc.jyu.fi/~aenevala/regressiotestaus.pdf

4 Software testing A Guide to the TMap Approach – Pol, Teunissen, van Veenendaal, 2002

5 Strengths of black box testing [www-sivu] [viitattu 8.10.2007] Saatavissa: http://www.cse.fau.edu/~maria/COURSES/CEN4010-SE/C13/black1.html

6 Software business competence [www-sivu] [viitattu 8.10.2007] Saatavissa: http://www.oamk.fi/sbc/ohjelmistotuote/ohjelmistotuotanto/testaus.htm

7 Glass box testing [www-sivu] [viitattu 8.10.2007] Saatavissa: http://en.wikipedia.org/wiki/Glass_box_testing

8 Ohjelmistotuotanto, Ilkka Haikala ja Jukka Märijärvi, 7.painos 2000

9 Testausmenetelmien käyttö sovelluksen systeemitestausvaiheessa [www-sivu][viitattu 8.10.2007] Saatavissa: http://www.mit.jyu.fi/opetus/opinnayte/gradu/systeemitestaus/Tiina_Tersa.pdf

10 Ohjelmistotestaus ja siinä käytettävät työkalut [www-sivu] [viitattu 8.10.2007] Saatavissa: http://www.mit.jyu.fi/opiskelu/seminaarit/ohjelmistotekniikka/testaus/#RTFToC7

11 Testausstrategiat [www-sivu] [viitattu 15.10.2007] Saatavissa: http://www.oamk.fi/sbc/testaus/testausstrategiat.htm

12 Ohjelmistotestaus ja siinä käytettävät työkalut [www-sivu] [viitattu 8.10.2007] Saatavissa: http://www.buzzle.com/editorials/4-10-2005-68349.asp

13 Ohjelmiston testaus [www-sivu] [viitattu:15.10.2007] Saatavissa: http://www.cc.jyu.fi/~akrats/harkat/ohjtuo/Ohjelmiston_testaus.htm

14 Cause-effect graph [www-sivu] [viitattu 8.10.2007] Saatavissa: http://en.wikipedia.org/wiki/Cause-effect_graph

15 Tekstiviesti [www-sivu] [viitattu 8.10.2007] Saatavissa: http://fi.wikipedia.org/wiki/Tekstiviesti

16 Multimedia Messaging Service [www-sivu] [viitattu 8.10.2007] Saatavissa: http://fi.wikipedia.org/wiki/MMS

17 Testattua [www-sivu] [viitattu 8.10.2007] Saatavissa: http://www.puhelimeni.fi/uutiset/index.php?itemid=494

18 Suoratoisto www-sivu] [viitattu 8.10.2007] Saatavissa: http://fi.wikipedia.org/wiki/Streaming

19 Flander Oy. [www-sivu]. [viitattu 17.10.2007] Saatavissa: http://www.flander.com