TAMPEREEN
AMMATTIKORKEAKOULU

LIIKETALOUS

TUTKINTOTYÖRAPORTTI

TESTDIRECTOR TESTAUKSENHALLINTAJÄRJESTELMÄNÄ SERIES 60 -OHJELMISTOALUSTAN TESTAUKSESSA

Ongelmat ja parannukset testaajan näkökulmasta

Susanna Hakala

Tietojenkäsittelyn koulutusohjelma
Joulukuu 2005
Työn ohjaaja: Jyrki Vehmas

TAMPERE 2005

TAMPEREEN AMMATTIKORKEAKOULU

LIIKETALOUS

Tekijä(t): Susanna Hakala

Koulutusohjelma(t): Tietojenkäsittely

Tutkintotyön nimi: TestDirector testauksenhallintajärjestelmänä Series 60 -ohjelmistoalustan testauksessa -Ongelmat ja parannukset testaajan näkökulmasta

Title in English: TestDirector as the test management tool in the testing of Series 60 Platform – Problems and improvement suggestions from the test engineer's point of view.

Työn valmistumis-kuukausi ja -vuosi: Joulukuu, 2005

Työn ohjaaja: Jyrki Vehmas **Sivumäärä:** 56 + 4

TIIVISTELMÄ

Series 60 -ohjelmistoalusta on tällä hetkellä maailman kehittynein älypuhelinalusta. Se tarjoaa muun muassa helppokäyttöisen graafisen käyttöliittymän sekä suuren määrän erilaisia sovelluksia. Alustan ovat lisensoineet matkapuhelinvalmistajat LG Electronics, Lenovo, Nokia, Panasonic, Samsung, Sendo ja Siemens.

Testausta tehdään monessa ohjelmistoprojektin vaiheessa, jotta virheitä löydettäisiin mahdollisimman paljon. Kuitenkaan ohjelmistosta on mahdotonta löytää ja poistaa kaikkia virheitä.

Series 60 -ohjelmistoalustan testauksen tukena on testauksenhallintajärjestelmä TestDirector. Sen avulla luodaan testit, suoritetaan ne sekä seurataan (testien) tuloksia ja virheitä. Järjestelmässä on ilmennyt erilaisia ongelmia sekä puutteita. Tutkintotyöni tavoitteena olikin tutkia niitä testaajan näkökulmasta.

Suoritin aiheesta kyselyn sähköpostitse. Kyselyn tuloksena sain vastauksia, joista kävivät ilmi vastaajien kokemat ongelmat sekä heidän parannusehdotuksensa TestDirectoriin. Vastaukset vahvistivat omia näkemyksiäni järjestelmästä. Olen käyttänyt kyseistä järjestelmään testaajan työssäni.

Työhöni olen koonnut yhteenvedon kyselyn vastauksista. Sen perusteella voidaan arvioida, miten Series 60:n ongelmat pitäisi ratkaista. TestDirectoriin on mahdollista lisätä itsenäisesti, ilman järjestelmän valmistajaa, uusia toiminnallisuuksia sen API-rajapinnan kautta.

Useat TestDirectorin ongelmat hidastavat työntekoa. Jos nämä olisivat kunnossa, testaus sujuisi huomattavasti joustavammin ja nopeammin. TestDirectorin epävakaus sekä kaatuilu aiheuttavat liian usein sen, että testaajien työt keskeytyvät. Epävakauden lisäksi järjestelmän muusta toiminnallisuudesta löytyi laajuudeltaan pienempiä, lähinnä käyttöliittymään, liittyviä ongelmia.

Avainsanat: ohjelmistotestaus testauksenhallinta testausprosessi ohjelmistoalusta

Sisällysluettelo:

1. Johdanto

Symbian–käyttöjärjestelmän päälle rakennettu Series 60 -ohjelmistoalusta on tällä hetkellä maailman johtava älypuhelinohjelmistoalusta. Se koostuu käyttöliittymästä, joka palvelee vuorovaikutuksellaan laitteen dataa ja ohjelmistoa sekä sovelluksia.

Ohjelmiston laadulla ja virheettömyydellä on suuri merkitys ohjelmiston käyttäjän kannalta. Testauksen avulla pyritään löytämään ohjelmistosta virheitä. Testaus käsittää viisi toimenpidettä: testauksen suunnittelun, testien määrittelyn, suorittamisen sekä suoritteiden tallentamisen ja testin toteutumisen tarkastelun. Testausprosessi alkaa aina testien suunnittelulla ja päättyy toteutumisen tarkasteluun. Kaikkia toimenpiteitä saatetaan joutua toistamaan useampaankin kertaan ennen kuin testauksen toteutumiskriteerit saavutetaan.

Testauksen apuna Series 60 -ohjelmistoalustassa on käytössä Mercury Interactiven kehittämä testauksenhallintajärjestelmä, TestDirector. TestDirector määrittää ja organisoi testauksen hallintaa antamalla järjestelmällisen ohjauksen testausprosessin läpi. TestDirector luo puitteet ja alustan testaukselle.

Opinnäytetyöni tavoitteena oli tutkia, miten Series 60 -ohjelmistoalustan testaajat kokevat käytössä olevan testauksenhallintajärjestelmän: mitä ongelmia TestDirectorissa ilmenee sekä mitä mahdollisia parannusehdotuksia testaajilla on järjestelmän toiminnallisuutta silmällä pitäen. Saadakseni selville ongelmat ja parannusehdotukset mahdollisimman monelta testaajalta päätin laatia kyselylomakkeen TestDirectorin käyttäjäoppaan ja oman TestDirector kokemukseni pohjalta. Lähetin lomakkeen sähköpostitse osalle eli noin 60 Series 60:n testauksen työtekijälle. Vastausprosentti oli noin 33 eli vastauksia tuli 22 kappaletta. Vastausprosentin pienuudesta huolimatta pystyin tekemään johtopäätöksiä järjestelmästä.

2. Series 60 -ohjelmistoalusta

Series 60 on alkujaan Nokian kehittämä ohjelmistoympäristö, joka lanseerattiin marraskuussa vuonna 2001. Se on maailman johtava älypuhelinohjelmistoalusta, joka tarjoaa puhelimille kattavan ohjelmistopohjan kehittyneillä tietoresursseilla. Series 60 -ohjelmistoalusta rakennetaan avoimen Symbian-käyttöjärjestelmän päälle. Symbian-käyttöjärjestelmä pohjautuu asiakas-palvelin –arkkitehtuuriin. Asiakas-palvelin –arkkitehtuuri on laajasti tunnustettu tehokkaaksi järjestelmäksi ohjelmistoyhteisössä. (Series 60…2004)

Series 60 –ohjelmistoalusta on suunniteltu yhdellä kädellä käytettäville matkapuhelimille. Ohjelmistoalusta rakentuu käyttöliittymästä, puhelimesta, viestitoiminnoista sekä verkossa selattavista palveluista. Alusta tarjoaa helppokäyttöisen graafisen käyttöliittymän, suuren määrän edistyksellisiä sovelluksia (kuten kalenterin, kameran, internet-selaimen), mahdollisuuden käyttää useampaa sovellusta samanaikaisesti sekä tallettaa tietoa muun muassa muistikortille. (Series 60…2005)

Series 60 -ohjelmistoalustan lisenssi tarjoaa kattavan ohjelmistopaketin, joka sisältää kaikki vaadittavat palvelun ”mahdollistajat”, eheän käyttöliittymän sekä laajan valikoiman sovelluksia. Series 60 -ohjelmistoalusta, joka lisensoidaan lähdekoodeineen, tarjoaa laitevalmistajille mahdollisuuden erottaa puhelimen ja käyttöliittymän suunnittelu sekä erinomaisen mahdollisuuden muokata alustaa omiin tarpeisiinsa. Alustan ovat lisensoineet muun muassa LG Electronics, Lenovo, Nokia, Panasonic, Samsung, Sendo ja Siemens. (Series 60…2005)

Älypuhelimet ovat avanneet uusia mahdollisuuksia ohjelmiston kehittäjille. Tavallisen GSM-puhelimen ominaisuuksien lisäksi Series 60 -älypuhelimet mahdollistavat ohjelmistosovellusten lataamisen, internetin selaamisen ja viestiliikenteen hallinnan. Käyttäjä pystyy hallitsemaan henkilökohtaisia tietojaan, synkronisoimaan puhelimen tietokoneen kanssa sekä luomaan omaa sisältöä puhelimeen, kuten valokuvia. Ohjelmistoalusta tukee myös uusia mobiilipalveluja,

kuten multimediaviestejä sekä sähköpostia. Ohjelmistoalusta antaa mahdollisuuden suunnitella innovatiivista sisältöä puhelimeen nopeasti kasvavilla markkinoilla. Matkapuhelinoperaattorit ja palveluntarjoajat tarjoavat Series 60 -laitekäyttäjille satoja sovelluksia ja myös muuta sisältöä, kuten teemoja ja soittoääniä. (Series 60…2005)

Series 60 -käyttöliittymä on kaikkein laajimmin tutkittu ja perusteellisimmin kehitetty graafinen käyttöliittymä, minkä Nokia on koskaan kehittänyt. Sen sisällyttäminen Series 60 -ohjelmistoalustaan varmistaa käyttöliittymän yhtenäisyyden kaikissa Series 60 -pohjaisissa laitteissa. (Series 60…2004) Sovellukset ovat huolellisesti integroituja keskenään. Käyttäjä voi toteuttaa helposti erilaisia toimintoja, kuten luoda ja lähettää multimediaviestin minimaalisella määrällä näppäimen painalluksia. Series 60 -käyttöliittymä varmistaa mukavan ja johdonmukaisen käyttäjäkokemuksen lisenssituotteiden kautta. (Series 60…2005)

3. Ohjelmistotestaus

Kaikki ohjelmistokehityksen tuotteet ovat ihmisten käsialaa. Ihmiset tekevät virheitä huolimatta siitä, kuinka paljon heiltä löytyy kokemusta sekä taitoa. Virheet ohjelmistossa ovat väistämättömiä.

Aikaisemmin testauksen huomio kiinnittyi lähinnä havainnoimaan ohjelmiston virheettömyyttä. Tuolloin testaus keskittyi suurimmaksi osaksi projektin loppuun. Testausta ei mitattu lainkaan, joten se oli täysin hallitsematonta. Nykyään testauksella pyritään saamaan testattavaan kohteeseen lisää arvoa eli poistamaan siellä olevia virheitä. Testauksen yksi määritelmistä kuuluu seuraavasti: ”Testaus on ohjelman suorittamista siinä olevien virheiden löytämiseksi.” Määritelmä asettaa testauksen tavoitteet korkealle. Sitä parempi, mitä enemmän virheitä löydetään testauksen aikana ja tällä tavoin saavutetaan parempi lopputulos. Käytännössä on todella vaikeaa todistaa ohjelman 100 prosenttinen oikeellisuus. (Ohjelmistotestaus… 2004)

Ohjelmistotestaus kehittyy jatkuvasti kohti monivaiheista prosessia erilaisine menetelmineen, rakenteineen, organisaatioineen ja dokumentointeineen. Usein testaus onkin ainutlaatuinen jokaisessa organisaatiossa. Kuitenkin monet testausryhmät seuraavat yksinkertaista toimintatapaa, joka samaistaa toiminnallisuusvaatimukset, luo testisuunnitelman, suorittaa testit ja jäljittää ohjelmiston virheet. (Mercury Interactive, Implementing…2003.)

Testaus voidaan jakaa kahteen keskeiseen näkökulmaan: mustalaatikkotestaukseen ja lasilaatikkotestaukseen.

3.1. Mustalaatikkotestaus

Mustalaatikkotestaus kohdistuu ohjelman ulkoisiin piirteisiin ja toimintoihin. Mustalaatikkotestaus on testausstrategia, jossa testattavaa ohjelmistoa tai sen moduulia suoritetaan ja ohjataan vain sen käyttörajapinnan kautta. Testaus suunnitellaan vaatimusten ja muun spesifikaation perusteella, ilman että testaajalla on mitään tietoa ohjelmiston tai moduulin sisäisestä rakenteesta. Tästä syystä mustalaatikkotestausta toteuttaa muu kuin ohjelmiston kehittäjä. (Ohjelmistotestaus…2004; Nguyen 2003: 7.)

Mustalaatikkotestauksessa testitapausten kattavuutta tarkastellaan moduulin saamien syötteiden ja sen antamien tulosteiden perusteella. Tällainen testaus tarkastelee sovelluksen odotettua käyttäytymistä käyttäjän näkökulmasta ja painopisteenä on toiminnallisuus. Mustalaatikkotestaus voi olla tehoton paljastamaan varmoja virhetyyppejä, kuten tietovuotovirheitä tai rajapintavirheitä kooditasolla. (Ohjelmistotestaus…2004; Nguyen 2003: 7.)

3.2. Lasilaatikkotestaus

Lasilaatikkotestaus on vastakohta mustalaatikkotestaukselle. Lasilaatikkotestauksessa testaaja on tietoinen ohjelman toteutuksesta. Testauksen suunnittelussa voidaan käyttää spesifikaation lisäksi myös ohjelman sisäistä rakennetta. Lasilaatikkotestausta käyttää yleensä ohjelman kehittäjä.

Lasilaatikkotestauksen painopiste on testien kattavuudessa ja suorituskyvyssä. Testitapausten kattavuutta tarkastellaan sen perusteella, kuinka kattavasti ohjelma on testattu eli onko kaikki koodirivit käyty läpi. Lasilaatikkotestaus ei nosta helposti esille makrotason laaturiskejä käyttöympäristössä, yhteensopivuusongelmia, aika-relatiivisia eikä käytettävyydessä ilmenneitä virheitä. (Nguyen 2003: 7)

3.3. Ohjelmistotestauksen käsitteitä

Testauksesta puhuttaessa tulee vastaan erilaisia käsitteitä. Seuraavassa on selitetty muutamia yleisimmin käytettyjä käsitteitä.

Testauksessa tulee väkisin vastaan *testitapaus*. Testitapaus on testauksen perusyksikkö. Testitapaus koostuu yhden testin lähtökohdista eli mistä tilanteesta testi lähtee liikkeelle, syötteistä eli esimerkiksi annettavista näppäinten painalluksista, askelten suoritteista eli mitä tapahtuu, kun näppäimiä painetaan sekä oikeasta lopputuloksesta eli mitä todella tapahtuu näppäinten painalluksista.

*Testispesifikaati*o eli testimäärittely sisältää yleensä tietyn tyyppisiä testitapauksia, esimerkiksi kaikki tietyn sovelluksen testit.

Testityyppi kuvaa tavoitteen, sisällön, tuntomerkit ja/tai testauksen painopisteen.

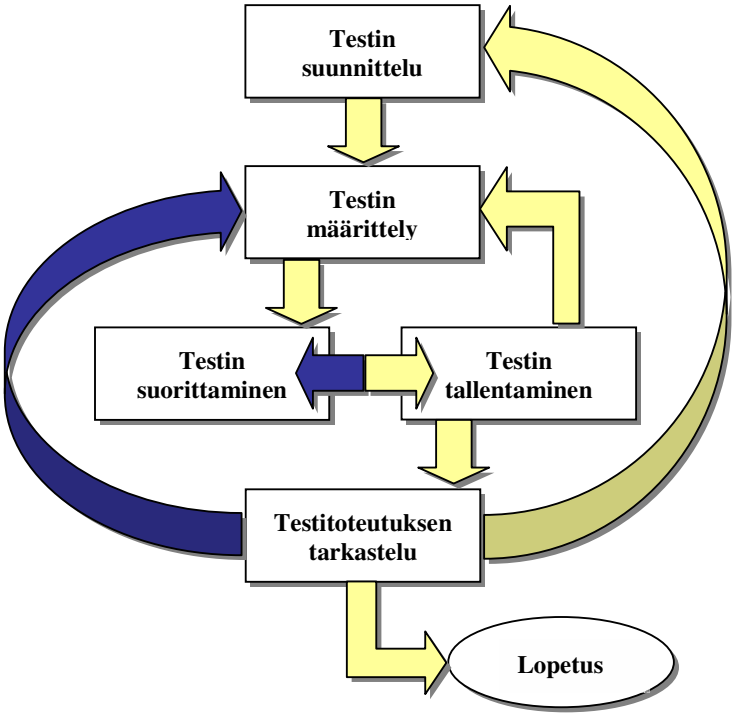
Testaussuunnitelma pitää sisällään suunnitelman testauksen aikataulusta, resursseista, työkaluista sekä testausympäristöstä.

Testausraportissa on tuloksia ja lukuja sekä diagrammeja testien suorituksista. (Series 60 Testing…2005)

Virhe on poikkeama spesifikaatiosta. Testausta on mahdotonta tehdä ilman spesifikaatiota, sillä lopputuloksen oikeellisuutta on tällöin vaikea todeta. (Haikala ym. 2004: 287.)

4. Testausprosessi

Testausprosessi (kuva 1) voidaan jakaa eri vaiheisiin samaan tapaan kuin ohjelmistokehitysprosessi. Perusteellinen testausprosessi käsittää viisi toimenpidettä: suunnittelun, määrittelyn, suorittamisen, tallentamisen sekä toteutumisen tarkastelun.



Kuva 1. Testausprosessi. (Software Testing…2003)

Testausprosessi alkaa aina testien suunnittelulla ja päättyy testien toteutumisen tarkasteluun. Kaikkia toimenpiteitä voidaan joutua toistamaan useampaan ker-

taan ennen toteutumiskriteerien saavuttamista. Mikään toimenpide ei voi alkaa ennen kuin edellinen toimenpide on päättynyt. Testausprosessin läpivienti on aina dokumentoitava. (Software Testing…2003)

Testauksen työvaiheisiin ja niihin läheisesti liittyvään virheiden jäljitykseen ja korjaukseen kuuluu tyypillisesti yli puolet ohjelmistoprojektin resursseista, joten testauksen läpivientiin parhaalla mahdollisella tavalla kannattaa kiinnittää huomiota. (Haikala ym. 2004: 283.)

Hyvän testauksen pohjana on hyvä testisuunnittelu. Testisuunnittelu tuottaa testisuunnitelman spesifioituna kyseiselle testauksen tasolle. Kun tehdään testisuunnitelmaa, on selkeästi määriteltävä testauksen laajuus ja mainittava kaikki olettamukset, jotka on oltava tehtynä ennen testin määrittelyä sekä suorittamista. Suunnitelmaan tulee myös hahmottaa kaikki muut ohjelmistovaatimukset ennen kuin testaus voidaan aloittaa sekä määriteltävä testauksen toteutumisen kriteerit. Testaussuunnitelmasta selviää muun muassa mitä testejä tehdään ja milloin, miten ne järjestetään ja millaisia tuloksia odotetaan. (Software Testing…2003.)

Määrittelyssä käytetään testitapausten suunnittelussa valittuja tekniikoita. Jokaiselle testitapaukselle määritellään tavoite, ohjelmiston lähtötilanne, syötteet sekä odotettu lopputulos. Määrittely voidaan jakaa kolmeen erilliseen osaan: testiolojen hahmottamiseen, testitapausten suunnitteluun ja testitapausten toteutukseen. (Software Testing…2003)

Testien suorittamisessa on tarkoituksena ajaa kaikki testitapaukset läpi. Testien suorittaminen voidaan tehdä joko manuaalisesti tai käyttäen automatisointityökalua.

Manuaalisesti ajettavat testit ajetaan määrittelyn mukaisilla syötteillä. Testin suorituksessa saatuja todellisia tuloksia verrataan odotettuihin lopputuloksiin. Testit saavat tuloksesta riippuen hyväksytty- tai hylätty-statuksen. Hylätty-statuksen omaavissa testitapauksissa on havaittu virhe. Yleisesti kaikkein tär-

keimpiä testitapauksia ovat ne, joiden avulla löydetään vakavia virheitä ohjelmistosta. (Software Testing…2003)

Automatisoidut testit edellyttävät, että testaaja ohjelmoi ajettavat testit koneellisesti luettavaan muotoon, kuten skripteiksi. Automatisointityökalu muodostaa ajoympäristön, joka ottaa sovelluksen käyttöön, lähettää dataa kuten luotuja skriptejä sovellukselle, tallettaa tulosteen, lähettää tulosteen vertailijalle arvioitavaksi sekä kirjaa tulokset. (Tamres 2002: 226)

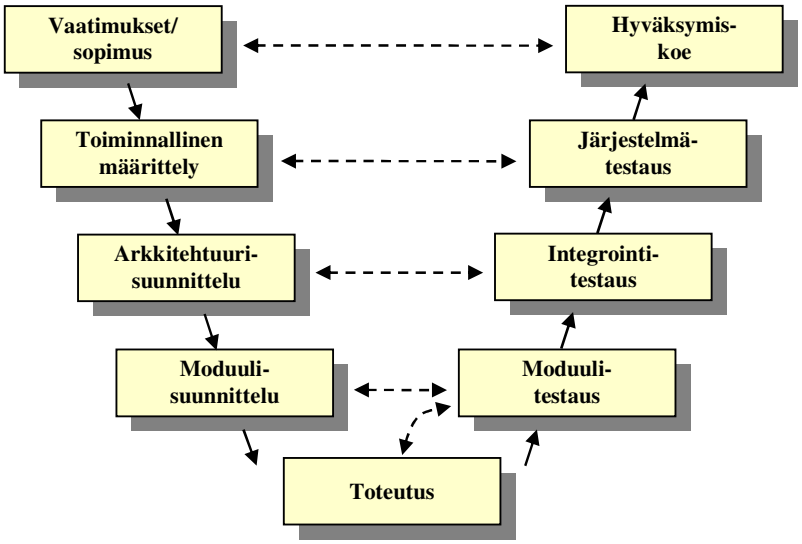
Testien tallentaminen on sidoksissa testien suorittamisen kanssa. Aluksi tarvitsee tallentaa ohjelmistoversiot ja mitä testimääritystä käytetään. Jokaisesta testitapauksesta tallennetaan todellinen lopputulos ja suoritettujen testien lopputulosten perusteella arvioidaan, täyttävätkö testien tulokset vaaditut kriteerit. Todellista lopputulosta verrataan odotettuun lopputulokseen. Kun ristiriitaa todellisen ja odotetun lopputuloksen välillä ilmenee, se kirjataan ja analysoidaan, missä virhe sijaitsee. (Software Testing…2003)

Toteutumisen tarkastelussa tutkitaan testien tallettamia tietoja toteutumiskriteerejä vastaan. Jos kriteerit eivät täyty, on palattava takaisin määrittelytasolle täydentääkseen testitapausten määrää toteutuskriteerien tasolle. (Software Testing…2003)

Riittävää testauksen määrää on vaikea arvioida. Tuotekehitystyössä testauksen lopettaminen on kompromissi tuotteessa olevien virheiden aiheuttamien kustannusten ja markkinoilta myöhästymisen aiheuttaman tuoton menetyksen välillä. (Haikala ym. 2004: 293.)

5. Testauksen tasot

Testaus voidaan jakaa useaan eri tasoon esimerkiksi V-mallia (kuva 2) apuna käyttäen.



Kuva 2. V-mallin testaustasot. (Ohjelmistotestaus…2004)

Testausta tehdään kehitettävän ohjelman eri rakennetasoilla. Siirryttäessä V-mallin alimmalta tasolta ylöspäin testauksen luonne muuttuu lasilaatikkotestauksesta yhä enemmän mustalaatikkotestaukseksi. Testaus voidaankin jakaa kolmeen päätasoon: yksikkö- eli moduulitestaukseen, integrointitestaukseen sekä järjestelmätestaukseen. V-mallin mukaisesti testauksen suunnittelu tapahtuu testaustasoa vastaavalla suunnittelutasolla. Mitä korkeammalla V-mallin testaustasolla ollaan, sitä kalliimmaksi virheiden korjaus tulee. (Haikala ym. 2004: 290, 292.)

5.1. Moduulitestaus

Moduulitestauksessa testattavana on yksittäinen moduuli. Moduulitestaus on ohjelmiston ensimmäinen testausvaihe. Testauksen tavoitteena on löytää moduuliin jääneet virheet. Moduulin toimintaa verrataan moduulisuunnittelun ja arkkitehtuurisuunnittelun tuloksiin, tavallisimmillaan tekniseen määrittelydokumenttiin. Testauksen lähtökohtana on kuvaava spesifikaatio, jonka mukaisesti normaalisti moduulin ohjelmoija suorittaa testausta. (Haikala ym. 2004: 289.)

Painopisteenä testitapausten suunnittelussa on lasilaatikkotestaus. Moduulitestaus on viimeinen testauksen vaihe, jossa suunnittelija hallitsee kokonaan tuottamansa ohjelmakoodin sisällön ja toiminnan. Moduulitestaus on erittäin keskeinen testausvaihe koko ohjelmiston testauksessa. Tässä testauksen vaiheessa löytyneet virheet on vielä helppo korjata. (Ohjelmistotestaus…2004)

Mikäli moduulitestauksessa ei löydy virheitä, voidaan siirtyä integrointitestaukseen. Mikäli myöhäisemmässä vaiheessa ohjelmistosta löytyy virhe, on palattava takaisin moduulitestausvaiheeseen, korjattava virhe ja moduulitestattava uudelleen. (Ohjelmistotestaus...2004)

5.2. Integrointitestaus

Järjestelmät koostuvat useista moduuleista, jotka on koottava yhteen. Integrointitestauksessa testataan järjestelmän osien vuorovaikutuksen, yhteistoiminnan ja osien muodostamaa kokonaisuutta. (Tevanlinna 2004)

Integrointitestauksen painopisteenä on moduulien välisten rajapintojen toimivuuden tutkimisessa. Testauksen tavoitteena on integroida moduulit toisiinsa, löytää virheitä moduulien välisistä rajapinnoista ja varmistaa järjestelmän vakaus järjestelmätestauksen aloittamista varten. Integrointitestaus etenee usein rinnan moduulitestauksen kanssa. Sitä onkin usein tarpeetonta tarkastella erillään moduulitestauksesta. (Haikala ym. 2004: 290.)

5.3. Järjestelmätestaus

Järjestelmätestauksessa testataan koko järjestelmän toiminnallisuus ja suorituskyky. Järjestelmätestaus on tyypillisesti mustalaatikkotestausta ja testauksen lähtökohtana on toiminnallinen määrittely. Tuloksia verrataan määrittelydokumentaatioon ja asiakasdokumentaatioon. (Ohjelmistotestaus…2004)

Järjestelmätestaus testaa sovelluksen toimintaa eri olosuhteissa. Olosuhteita ovat muun muassa laitteistokonfiguraatiot, käyttöjärjestelmät ja käyttöjärjestelmäversiot, tietoliikenne, verkot ja protokollat, yhteistoiminta muiden sovellusten kanssa sekä resurssi- että aikakuormitus. (Ohjelmistotestauksen…2002)

Järjestelmätestauksen tarkoituksena on vähentää virheellisen järjestelmän toimittamisen riskiä. Järjestelmätestaukseen voidaan lisäksi liittää kenttätestaus sekä hyväksymistestaus. Järjestelmätestauksen suorittaa henkilöt, jotka ovat kehitystyöstä mahdollisimman riippumattomia. (Haikala ym. 2004: 290.)

5.4. Series 60 –ohjelmistoalustan testaustyypit

Series 60 –ohjelmistoalustassa testit luodaan käyttöliittymämäärittelyjen, tuotevaatimusten sekä hankkeiden ja korjausten mukaisesti sekä testataan dokumentoitu toiminnallisuus. Seuraavassa Series 60 -ohjelmistoalustan eri testityypit suoritusjärjestyksessä:

Moduulitestaus

Moduulitestaus suoritetaan aina, kun uutta koodia kuten virhekorjaus tai uusi toiminnallisuus lisätään olemassa olevaan sovellukseen. Moduulitestien suorittajina ovat useimmiten koodin kehittäjät eli koodaajat. Kun moduulitestin tulokset täyttävät vaaditut kriteerit, voidaan ominaisuus lisätä ohjelmistoon. (Series 60 Testing…2005)

BAT-testaus

Kun ohjelmistosta on tulossa uusi versio, suoritetaan versioehdokkaalla ensimmäisenä BAT -testaus (Basic Acceptance Test) eli perushyväksymistestaus. BAT-testauksessa testataan Series 60 –ohjelmistoversioiden perustoiminnallisuus ja näin varmistetaan, että ohjelmisto toimii riittävän hyvin julkaisemista varten. Testauksen suorittaa BAT-testauksesta vastaava testaaja. Testauksen jälkeen voidaan suorittaa muu testaus. Tällainen testaus on osa järjestelmätestausta. (Series 60 Testing…2005)

Toiminnallisuustestaus

Toiminnallisuustestauksen (functional testing) tarkoituksena on testata uusia toiminnallisuuksia jokaisessa alustan uudessa ohjelmistoversiossa. Vaatimusmäärittelyjen sekä käyttöliittymämäärittelyjen perusteella testaaja laatii tai päivittää toiminnallisuustestauksen spesifikaatiot.

Toiminnallisuustestauksessa käytetään joko aivan uusia testejä tai testataan uudelleen vanhoja testejä, josta on löytynyt virheitä. Uudelleen testauksella saadaan selville, onko virhe korjaantunut sovelluksessa vai onko se aiheuttanut lisää virheitä. Toiminnallisuustestaus on yksi Series 60:n järjestelmätestauksen osa. (Series 60 Testing…2005)

Regressiotestaus

Regressiotestauksella pyritään löytämään ohjelmistosta mahdollista taantumista. Regressiotestauksessa käytetään osaa tavallisimmista toiminnallisuustestauksen testitapauksista. Alustan regressiotestausta tehdään niin monta testikierrosta kuin on tarpeen. Myös regressiotestit kuuluvat osaksi järjestelmätestausta. (Series 60 Testing…2005)

Ei-toiminnallisuustestaus

Yksi järjestelmätestauksen osa on ei-toiminnallisuustestaus (non-functional testing). Ei-toiminnallisuustestauksen avulla varmistetaan, että laite toimii puhtaasti eri testitapauksissa. Testauksen tarkoituksena ei ole testata uusia toiminnallisuuksia vaan testaus sisältää käyttövarmuustestejä, resurssitestejä sekä suorituskykytestejä.

Käyttövarmuustestaus sisältää pitkäjaksoista testausta, kestävyys- ja elpymistestejä. Resurssitestaus sisältää rasitus-, muisti-, tiedonlataus- sekä energianhallintatestejä. Suorituskykytestit ovat sovelluksen suorituskyky-, tiedonläpivienti- sekä vertailutestejä. Testisuunnitelmat kirjoitetaan ja testit ajetaan ei-toiminnallisten vaatimusten mukaisesti. (Series 60 Testing…2005)

Kielivarianttitestaus

Kielivarianttitestauksessa testataan ohjelmiston kielivarianttiversioita, joissa huomio kiinnittyy toiminnallisuuteen sekä kielitieteeseen. Kielellisesti tekstien käännöksien on oltava järkeviä ja tilanteeseen sopivia. Testien suunnittelussa ovat apuna toiminnallisuustestit, päätestisuunnitelmat sekä niihin liittyvä data sekä tulokset ja kielivarianttisuunnitelma.

Kielivarianttitestauksessa tehdään kyseiselle versiolle ensin BAT-testit, joiden jälkeen ajetaan normaalit kielivarianttitoiminnallisuustestit sekä kielispesifisiä ominaisuuksia sisältäviä toiminnallisuustestejä. Ja jos on vielä tarpeellista, kielivarianttiversiolla ajetaan myös ei-toiminnallisuustestejä. Kielivarianttitestaus on myös osa järjestelmätestausta. (Series 60 Testing…2005)

True-testaus

True-testauksen eli käyttöönottotestauksen tarkoituksena on vahvistaa, että ohjelmisto toimii kuluttajan ennakko-odotusten ja toiveiden mukaisesti. Käyttöön-

ottotestauksella pyritään määrittelemään ohjelmiston ongelma-alueet. Testaajana toimii ohjelmiston loppukäyttäjä.

Testauksen tuloksena halutaan loppukäyttäjän reaktioita tuotteen yleislaadusta. Testaaja laatii viikoittain raportin, jossa selviää tuotteen maturiteetti eli niin sanottu kypsyys. Testauksen kautta kerätään käyttäjiltä ylimääräistä palautetta sekä parannusehdotuksia. True-testaus on osa hyväksymistestausta. (Series 60 Testing…2005)

6. Testauksen hallinta

Testauksen hallintaan sisältyy arviointia, tarkkailua, ohjausta/hallintaa ja tapausten tallentamista ja seurantaa. On ymmärrettävä, mitä toimintoja on tehtävä, jotta testaus pysyy vaadituissa tavoitteissa. Testausprosessiin kuuluu monia osatekijöitä. Testauksen hallinta saattaa muodostua ongelmaksi, ellei hallinnan apuna käytetä jotain työkalua.

Kuten ohjelmiston kehitysprosessi myös testausprosessi tarvitsee järjestelmällistä rakennuspalikkaa, joka sisältää vaatimusten määrittelyn, suunnittelun, toteutuksen, suorittamisen ja analysoinnin, varmistaakseen kattavuuden, yhtenäisyyden ja testauksen varojen uudelleenkäytön. (Mercury Interactive, Implementing…2003.)

Testauksen hallinta on menetelmä organisoida ohjelmistotestausta. Testauksen hallinta mahdollistaa testien helpon saavutettavuuden ja uudelleen käytettävyyden. Testauksen hallintaa yritetään juurruttaa tiukasti helpomman organisoinnin, yhteistyön ja informaation jaon käsitteisiin. (Mercury Interactive, Implementing…2003.)

Eheä vaatimuspohjainen testaus varmistaa, että viimeistelty järjestelmä kohtaa käyttäjän tarpeet. Ilman testauksen hallintaa, joka sitoo testisuunnitelmat ohjel-

miston toiminnallisiin vaatimuksiin, sallii organisaatioiden jäljitellä vaatimuksia ja muuttaa testitapauksia, on lähes mahdotonta toteuttaa testejä varmistaakseen, että järjestelmä sisältää tietyn toiminnallisuuden. (Mercury Interactive, Implementing…2003.)

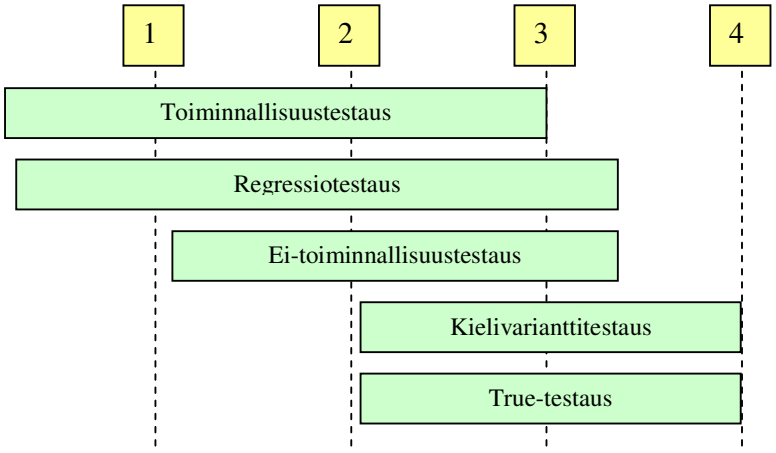
6.1. Series 60 -ohjelmistoalustan testauksenhallintaprosessi

Series 60 -ohjelmistoalustan testaus etenee pitkälti saman tapaan kuin miten olen aikaisemmin tässä työssäni kuvannut testausprosessin etenemistä. Asiakaspalautteen, ohjelmointisuunnitelman, inkrementtiaikataulujen ja tuotevaatimusten pohjalta luodaan testausstrategia, version päätestaussuunnitelma sekä alueellinen eli ohjelmiston sovellusten oma inkrementtitestisuunnitelma. (Series 60 Testing…2005)

Version päätetestisuunnitelman sekä inkrementtisuunnitelman perusteella toteutetaan Series 60:n varsinainen testaus. Testaustulosten perusteella laaditaan prosessimetriikat eli raportit, joissa luvuin ja diagrammein kuvataan testauksen tuloksia. Prosessimetriikoiden avulla tehdään testausprosessin uudistusta. Muutostoiminnat vaikuttavat uudelleen suunniteltavaan testausstrategiaan. (Series 60 Testing…2005)

6.2. Series 60 -testauksen virstanpylväät

Virstanpylväitä (milestone) käytetään kuvaamaan projektin erilaisia tavoitteita sen elinkaaressa. Virstanpylväskatselmuksissa katsotaan, onko projekti saavuttanut ennalta määritellyt tavoitteet. Jos näitä tavoitteita ei ole saavutettu, projekti ei saavuta virstanpylvästä ja joutuu pitämään uuden katselmuksen, kunnes tavoitteet on tyydyttävästi saavutettu. Virstanpylväitä käytetään aikataulujen suunnittelussa ja seuraamisessa. (Series 60 Testing…2005)



Kuva 10. Virstanpylväät sekä eri testityyppien sijoittuminen (Series 60 Testing…2005)

Virstanpylväs 1

Virstanpylvään ajankohtana suurin osa testaussuunnitelma pitäisi olla tehtynä.

Virstanpylväs 2

Ennen toista virstanpylvästä pitäisi olla testisuunnitelman hyväksytty sekä lähes kaikki testit ajettu.

Virstanpylväs 3

Jotta kolmas virstanpylväs saavutettaisiin, on suurin osa testeistä ajettu hyväksytysti ja toiminnallisuustestit lopetettu.

Virstanpylväs 4

Ohjelmisto pitäisi olla tässä vaiheessa testattu kokonaan ja valmis käyttäjille.

7. Mercury Interactive TestDirector - testauksenhallintajärjestelmä

Testauksenhallintajärjestelmä auttaa ohjelmistokehityksen elinkaaren alusta loppuun asti. Tämän kategorian järjestelmällä on tuki testauksen suunnittelulle ja seurannalle, mutta myös virheiden seurannalle.

Jotkut testauksenhallintatyökalut auttavat järjestelmän toiminnallisuuden hajottamisessa testitapauksiin ja kykenevät jäljittämään ja viittaamaan vaatimuksista testitapauksiin ja taas takaisin. Tällä tavoin, jos vaatimus muuttuu, testauksenhallintajärjestelmän avulla on mahdollista korostaa uudelleenajettavat sekä päivitettävät testitapaukset. Samalla tavoin jos testitapaus ei mene läpi testiä ajaessa, järjestelmä kykenee nostamaan ylös ne vaatimukset, joita testien hylkääminen koskee. (Software Testing…2003)

Mercury Interactiven kehittämä TestDirector on maailmanlaajuinen testauksenhallintajärjestelmä. Olemalla täysin internet-pohjainen, TestDirector tukee hajautetun testausryhmän kommunikointia sekä yhteistyötä ryhmien kesken. Tämän päivän organisaatiossa, erilaiset ryhmät on kytkettynä testausorganisaatioon: esimiehet, kehittäjät, asiakastuki sekä jopa asiakkaat. (Mercury Interactive, Implementing…2003.)

TestDirector on käytössä Series 60 –ohjelmistoalustan testauksen apuna. Series 60:n sisällä TestDirectorin käyttäjiä löytyy Suomen lisäksi muun muassa Amerikasta sekä Kiinasta.

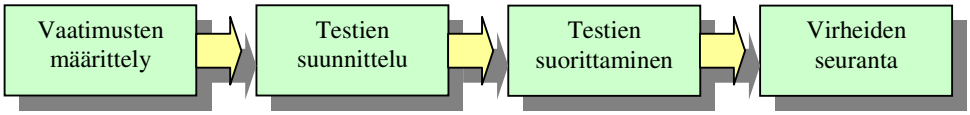
TestDirector sallii ryhmitellä korkeatasoisia sovelluksia nopeasti ja tehokkaasti tarjoten johdonmukaisen, toistettavan prosessin joukolle vaatimuksia, testien suunnittelun ja ajoittamisen, tulosten analysoimisen sekä vian käsittelyn. (Software Test…2005)

TestDirector määrittelee ja organisoi testauksen hallintaa antamalla järjestelmällisen ohjauksen testausprosessin läpi. TestDirector luo testaukselle sekä puit-

teet että alustan. TestDirector ylläpitää testien projektitietokantaa, joka kattaa kaikki sovelluksen toiminnallisuuden näkökannat. (Mercury Interactive, TestDirector…2003: 3.)

TestDirectorissa on neljä osiota testauksen hallintaa varten: vaatimusten määrittely, testien suunnittelu, testien suorittaminen ja virheiden seuranta. Osiot etenevät kuten kuvassa 3, joka vastaa pitkältä kuvan 1 testausprosessia.

Vaatimusten määrittelyssä analysoidaan ja määritellään testauksen vaatimukset. Testien suunnittelussa luodaan testisuunnitelma, jonka pohjana ovat testausvaatimukset. Testien suorittamisessa luodaan testisetti eli testien joukko ja suoritetaan testien ajot. Virheiden seurannassa raportoidaan sovelluksesta löytyneet virheet ja seurataan virheiden korjausten etenemistä. (Mercury Interactive, TestDirector…2003: 4.)



Kuva 3. TestDirectorin osiot. (Mercury Interactive, TestDirector…2003: 4.)

Series 60 –ohjelmistoalustan testauksessa TestDirectoria käytetään mustalaatikkotestauksen apuna. Järjestelmän avulla suoritetaan ainoastaan järjestelmätestauksesta, mutta mikään ei estäisi toteuttamasta integrointitestausta ja myös moduulitestausta TestDirectorissa.

7.1. TestDirectorin osiot

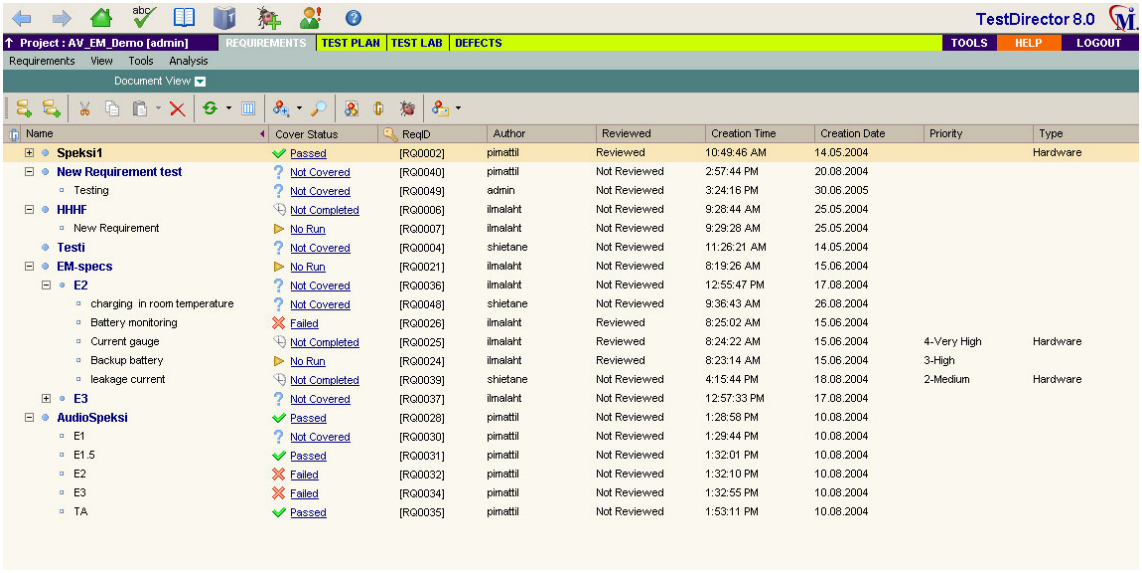
7.1.1. Vaatimusten määrittely

Testaustiimi aloittaa testausprosessin kokoamalla kaiken mahdollisen dokumentaation testattavasta sovelluksesta, kuten tuotevaatimukset, järjestelmän vaatimuskuvaukset ja suunnitteludokumentit. Tutkimalla kyseisiä dokumentteja pyritään hankkimaan perusteellista ymmärrystä sovelluksesta. Ymmärryksen avulla määritellään testaukselle puitteet: testauksen tavoitteet, päämäärät sekä strategiat. Testauksen puitteiden perusteella määritellään testauksen vaatimukset testattavaa sovellusta varten. (Mercury Interactive, TestDirector…2003: 66.)

Vaatimukset kuvaavat yksityiskohtaisesti, mitä tarpeita sovelluksen testauksella on. Ne määrittelevät testausryhmille perustan, miten testausprosessi etenee. Vaatimukset linkataan testeihin ja virheisiin, jotta saadaan järjestytettyä täydellinen jäljiteltävyys ja apu päätöksentekojärjestelmälle. (Mercury Interactive, TestDirector…2003: 65.)

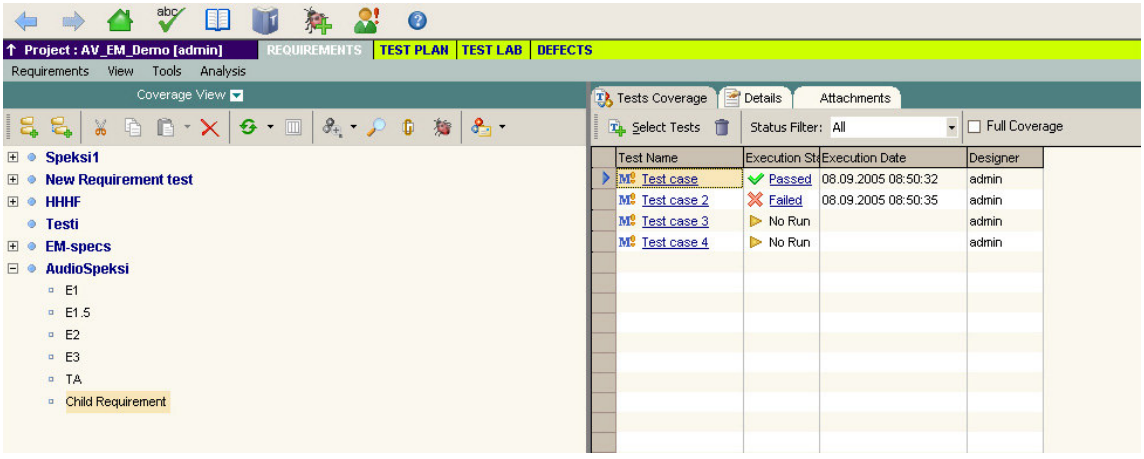
TestDirectorin Requirements-moduuli (kuva 4) tekee mahdolliseksi määritellä ja hallita testauksen vaatimuksia testausprosessin ensimmäisenä askeleena. Testausryhmät kirjaavat vaatimukset TestDirectoriin manuaalisesti. Requirements-moduulissa vaatimukset voidaan näyttää kolmella eri tavalla: Document-näkymänä, Coverage-näkymänä sekä Coverage Analysis -näkymänä. (Mercury Interactive, TestDirector…2003: 77.)

Document-näkymässä (kuva 4) vaatimukset näytetään vaatimuspuuna. Se on graafinen esitys vaatimusspesifikaatiosta, joka esittää vaatimukset hierarkkisessa järjestyksessä. Yksittäiset vaatimukset voidaan jakaa pienempiin osiin eli lapsivaatimuksiin. Lapsivaatimukseksi sanotaan vaatimusta, joka on päävaatimukseen nähden alemmalla hierarkkisella tasolla. Vaatimus voi pitää sisällään useampia lapsivaatimuksia. (Mercury Interactive, TestDirector…2003: 77.)

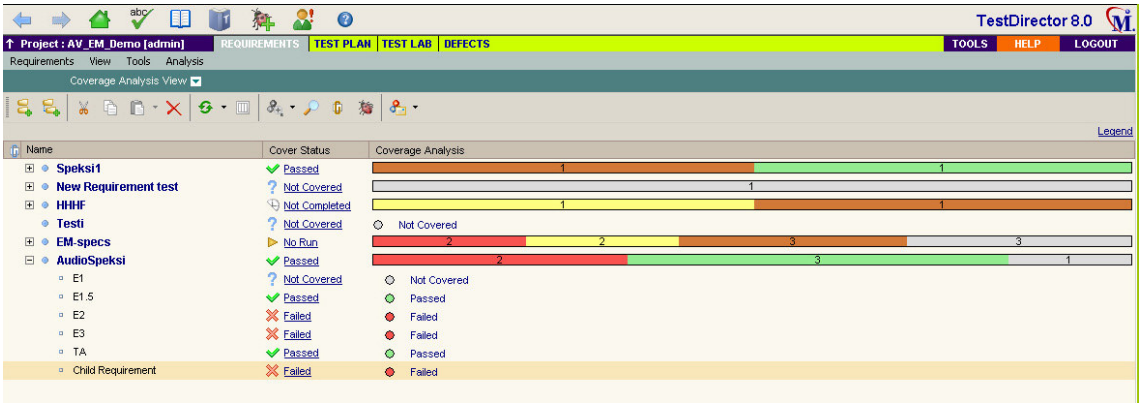


Kuva 4. TestDirectorin Requirements-moduulin Document-näkymä

Coverage- sekä Coverage Analysis –näkymissä vaatimukset näytetään testikattavuuden mukaan. Coverage-näkymä (kuva 5) mahdollistaa testien liittämisen vaatimukseen sekä testikattavuuden muuttamisen. Coverage Analysis –näkymän (kuva 6) avulla on mahdollista analysoida lapsivaatimusten erittelyjä vaatimusten kattavuusstatusten mukaan. (Mercury Interactive, TestDirector…2003: 70 - 71.)



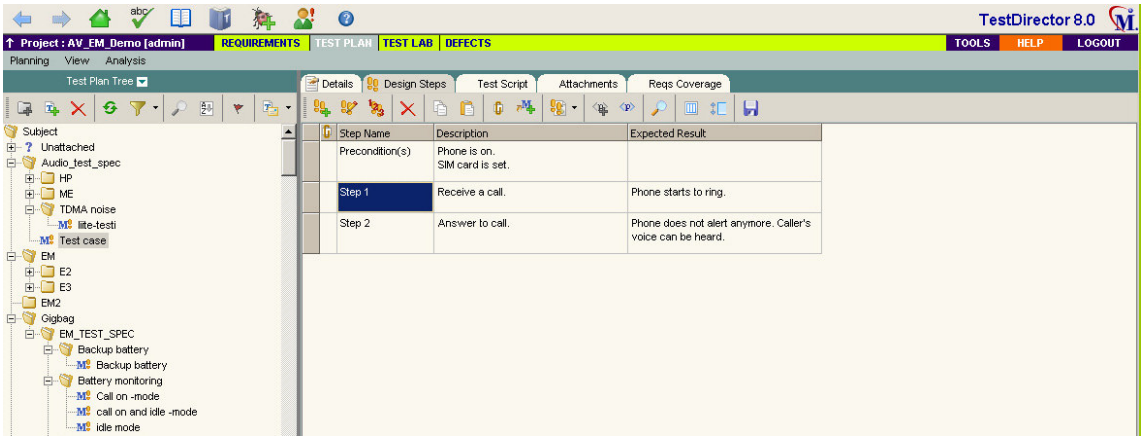
Kuva 5. TestDirectorin Requirements-moduulin Coverage-näkymä



Kuva 6. TestDirectorin Requirements-moduulin Coverage Analysis –näkymä

7.1.2. Testien suunnittelu

Tyypillinen sovellus on erittäin laaja, joten se on vaikea testata yhtenä kokonaisuutena, jos halutaan kunnon testaustuloksia. Tästä syystä testaus jaetaan usein pienempiin osiin. TestDirectorin Test Plan –moduulin (kuva 7) avulla sovellus voidaan jakaa osiin toiminnallisuutensa perusteella. TestDirector esittää sovelluksen osat testisuunnitelmapuussa. Se on graafinen esitys testisuunnitelmasta, joka näyttää hierarkkiset suhteet sovelluksen toiminnallisuudesta. (Mercury Interactive, TestDirector…2003: 115.)



Kuva 7. TestDirectorin Test Plan –moduuli ja testin askelmat.

Suunnitelmapuun voi nähdä kuvan 7 vasemmassa reunassa. Testisuunnitelmapuu on järjestelty aiheittain. Aiheet kuvaavat tiettyä toiminnallisuutta. Aiheet on jaoteltu kansioittain. Kansioita ja kansioiden alle luodaan testejä. Testiä luotaessa sille määritellään perustiedot kuten nimi, status ja suunnittelija. Testejä on hyvä luoda myös sellaiselle toiminnallisuudelle, jonka tiedetään varmasti sisältävän virheitä.

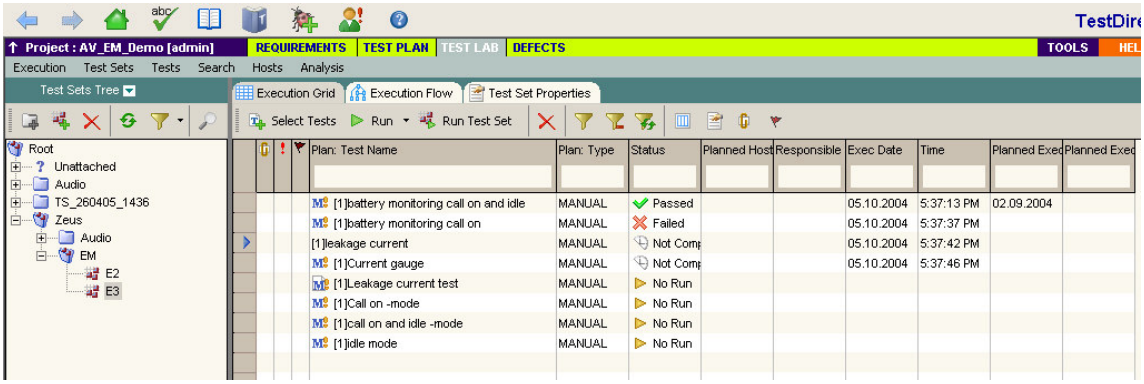
Kehiteltäessä manuaalisia testejä Test Plan –moduulissa määritellään testiaskelmat: yksityiskohtaiset askel-askeleelta esittelyt, kuinka testi ajetaan. Askelma sisältää suoritettavat toiminnot: annettavat syötteet ja odotetut tulosteet. Manuaalinen testi voidaan nähdä kuvan 7 oikean puoleisessa osiossa. Kuvan testissä näkyy askelmat numerojärjestyksessä. Description–kentässä on kuvattu jokaisen askelman annettavat syötteet ja suoritettavat toimenpiteet. Expected Result –kentät kuvaavat odotettuja tulosteita. Kuvan testitapaus on ainoastaan esimerkki testitapauksen rakenteesta TestDirectorissa. Testitapaus ei ole Series 60 -ohjelmistoalustan testitapaus.

7.1.3. Testien suorittaminen

Aina kun sovellus muuttuu esimerkiksi kun siihen on lisätty uusi ominaisuus tai kun vanhaa sovellusta on korjattu, suoritetaan projektin testit läpi. Testejä ajetaan, jotta virheet löydettäisiin ja päästäisiin arvioimaan sovelluksen laatua. Series 60:ssa testit ajetaan TestDirectorissa olevien testiaskelmien mukaisesti joko puhelimessa tai tietokoneen emulaattorilla. Testin tyypistä riippuen testi suoritetaan aina uudelleen niin kauan, kun virheitä löydetään. Samalla päästään testaamaan, ettei lisätty korjaus ole rikkonut sovellusta muualta.

Kun testit on luotu Test Plan -moduulissa, niistä kootaan testisetti Test Lab –moduuliin (kuva 8). Moduuliin luodaan testisettipuu, jonka avulla organisoidaan testausprosessia erilaisten hierarkioiden avulla. Testisetti on joukko testejä TestDirector-projektirakenteessa, jolla pyritään saavuttamaan testitavoitteita. Testisettipuu voi sisältää päätasolla erilaisia kansioita sekä alikansioita. Tiettyyn

ominaisuuteen liittyvät testisetit löytyvät omasta kansiostaan. (Mercury Interactive, TestDirector…2003: 195.)



Kuva 8. TestDirectorin Test Lab –moduuli sekä yhden testisetin testit.

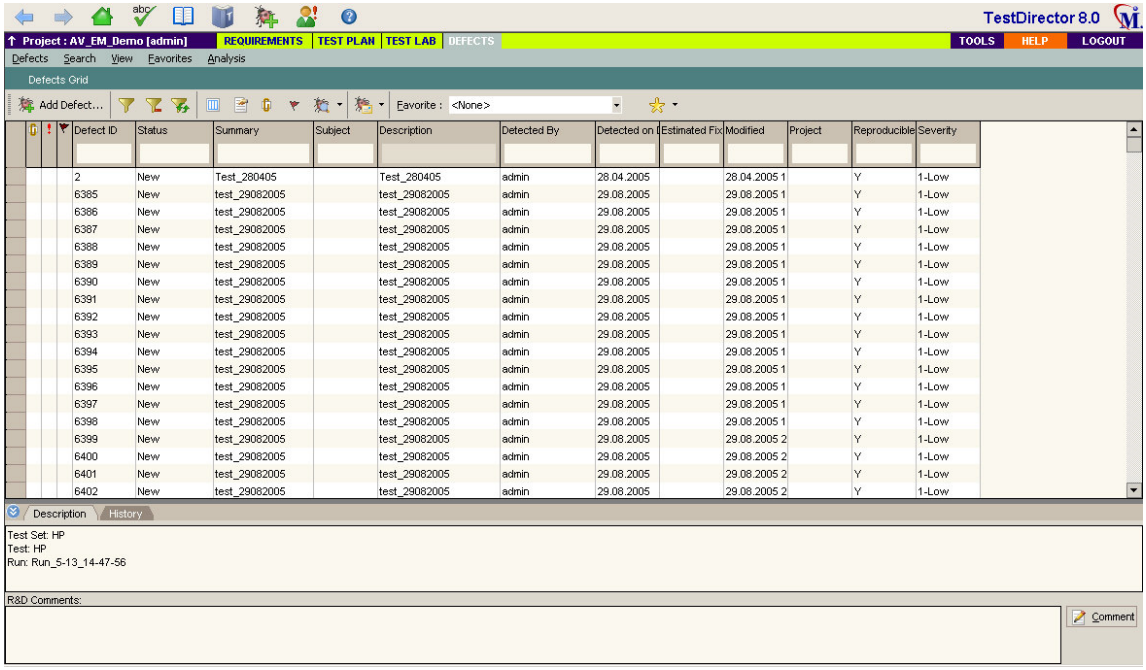
Kun testejä ajetaan manuaalisesti, seurataan testin askelmia ja sovelluksessa suoritettuja toimintoja. Jokainen askelma, joko hyväksytään tai hylätään, riippuen siitä, miten sovelluksen todelliset tulokset vastaavat odotettuja tulosteita. Automatisoituja testejä ajettaessa TestDirector avaa valitun testaustyökalun automaattisesti, ajaa testit ja vie testitulokset TestDirectoriin. (Mercury Interactive, TestDirector…2003: 236.)

Series 60 -ohjelmistoalustassa testejä ajetaan tällä hetkellä ainoastaan manuaalisesti.

Testien suorittamisen jälkeen tulokset voidaan nähdä TestDirectorissa. Testitulokset koostuvat kaikista hyväksytty/hylätty –statuksella olevista testeistä sekä niiden askelmista. Tulokset osoittavat sen, onko sovelluksesta löytynyt virheitä. Jossain tapauksissa askelmien hylkääminen voi johtua siitä, että odotetut tulokset ovat vanhentuneet, eivätkä täten ole enää kelvollisia. Tällaisessa tapauksessa testit on päivitettävä, jotta ne voidaan suorittaa uudelleen ja jotta odotettu toiminnallisuus vastaa uutta tulosta. (Mercury Interactive, TestDirector…2003: 249.)

7.1.4. Virheiden seuranta

Sovellusten virheiden paikallistaminen ja korjaaminen ovat elintärkeitä ohjelmiston kehittämisprosessille. TestDirectorin Defects-moduulin (kuva 9) avulla voidaan raportoida virhetallenteina testattavassa sovelluksessa ilmenneet virheet. (Mercury Interactive, TestDirector…2003: 265.)



Kuva 9. TestDirectorin Defects-moduuli

Aina kun testissä havaitaan virhe, on siitä luotava Defects-moduuliin uusi virhetallenne. Virheitä voidaan löytää ja liittää TestDirector-projektiin kaikissa testausprosessin vaiheissa. Kuvassa 9 voidaan nähdä moduulin virhetallenteet. Jokainen virhetallenne on omalla rivillään. Löydetty virhe on liitettävä aina ajettuun testiin, jotta nähdään missä testissä virhe on havaittu. Moduulista nähdään myös mahdollinen uudelleenajon tarve, sillä virheen korjautuessa on Defects-moduuliin muutettava virheen tilaa. Korjattu virhe johtaa siihen, että testi, jossa virhe on ilmaantunut, on ajettava uudelleen.

Virhetallenteet informoivat muita uusista löydetyistä virheistä. Virheenkorjaamisprosessi muuttuu nopeammaksi, tehokkaammaksi ja perusteellisemmaksi, kun virhetietoja jaetaan. Tieto virheen korjauksesta päivitetään TestDirector-projektiin. Näin muutkin saavat tiedon siitä, että virhe on korjaantunut. (Mercury Interactive, TestDirector…2003: 277.)

7.2. TestDirectorin käyttö Series 60 –ohjelmistoalustan testauksessa

TestDirector on käytössä osassa Series 60 –ohjelmistoalustan testaustyypeistä. Järjestelmää käytetään BAT-testauksessa, toiminnallisuustestauksessa, regressiotestauksessa, ei-toiminnallisuustestauksessa sekä kielivarianttitestauksessa.

True-testauksessa TestDirectoria ei ole mahdollista käyttää, sillä tässä testityypissä testaajina toimivat lopulliset käyttäjät. True-testauksessa ei käytetä valmiiksi suunniteltuja testejä vaan ja testauksen tuloksina ovat käyttäjän mielipiteet ohjelmiston toiminnallisuudesta.

Kehittäjien tekemässä moduulitestauksessa ei myöskään ole järjestelmä käytössä. Olisi kuitenkin mahdollista kehittää, että myös moduulitestaus tultaisiin hoitamaan TestDirectorissa. Näin moduulitestauksen tuloksetkin olisi muiden kuin kehittäjän nähtävillä ja mahdollisesti muiden ajaessa testejä, löytyisi testit järjestelmästä helposti.

7.3. TestDirectorin ylläpito Series 60 –ohjelmistoalustassa

TestDirector tarjoaa eheän ja avoimen testauksenhallintakehyksen, joka mahdollistaa kaikkien testauksen vaiheiden hallinnan ja kontrolloinnin.

TestDirectoriin on mahdollista lisätä uutta toiminnallisuutta ilman järjestelmän valmistajaa, Mercury Interactivea. Nokian sisällä on organisaatio, joka pitää yllä TestDirectoria ja tekee sinne tarvittavia muutoksia.

Mahdollisuudet Nokian sisäisiin TestDirectorin toiminnallisuusmuutoksiin ovat täten lähes rajattomat. Mercury tarjoaa API (application program interface) rajapinnan, jonka avulla TestDirectorin toiminnallisuutta voidaan laajentaa erilaisilla testauksen sekä raportoinnin ratkaisuilla. Myös Series 60 -ohjelmistoalustan oma TestDirector-tiimi pystyy lisäämään joitain muutoksia TestDirectorin toiminnallisuuteen.

TestDirectorin avoin API ryhmitellään kategorioihin ytimen moduulien mukaisesti: Requirement Management eli vaatimusten hallinta, Test Plan, Test Lab sekä Defect Tracking eli virheiden seuranta. Moduulit mahdollistavat helpon tiedonsaannin missä tahansa testausprosessin vaiheessa. Tämä API tarjoaa funktioita, jotka sallivat yhteydenoton projektitietokantaan, tuovat tietoa ulkopuolisilta sovelluksilta projektitietokantaan sekä vievät tietoa projektitietokannasta ulkopuolisiin sovelluksiin. (Mercury TestDirector 2005.)

8. Series 60 -ohjelmistoalustan testauksessa suoritettu TestDirector –kysely

8.1. Kyselyn tavoite

Sain toimeksiannon työlleni Series 60 –ohjelmistoalustan TestDirector-tiimiltä. Tutkintotyöni tavoitteena oli miettiä TestDirector-testauksenhallintajärjestelmään liittyviä käytännön ongelmia ja parannuksia testaajan näkökulmasta. Selvitykseni avulla on tarkoitus nostaa esille tekijöitä, jotka haittaavat testaajien jokapäiväistä työntekoa TestDirectorissa.

Korjausehdotukset luovutetaan Series 60 –ohjelmistoalustan TestDirector-tiimille, joka päättää mitä ongelmille sekä parannusehdotuksille tehdään.

8.2. Kyselyn toteutustapa

Olen toiminut Series 60 –ohjelmistoalustan testaajana, joten olen käyttänyt paljon kyseistä järjestelmää. Täten minulla on vahva näkemys järjestelmästä, sen ongelmista ja kaivatuista parannuksista. En kuitenkaan halunnut koota mahdollisia ongelmia ja parannusehdotuksia ainoastaan oman kokemukseni sekä näkemykseni pohjalta, halusin saada mukaan myös muiden testaajien näkemyksiä.

Toimeksiannon saatuani aloin miettimään mahdollisia toimintatapoja tavoitteen saavuttamiseksi. Mielessäni kävivät sekä lähetettävä kyselylomake että perinteinen haastattelu. Koska Series 60 -ohjelmistoalustan testaus ei työskentele yhdessä toimipisteessä, oli haastattelu suljettava pois. Päädyin laatimaan kyselylomakkeen.

Aloitin lomakkeen laatimisen oman TestDirector-käyttökokemukseni pohjalta. En ole työssäni käyttänyt kaikkia mahdollisia TestDirectorin ominaisuuksia.

Minun olikin selvitettävä, mitä kaikkea järjestelmällä voi tehdä. Hankin käsiini TestDirector-käyttäjäoppaan. Opas toimi hyvänä runkona kyselylomakkeelle.

Jaoin lomakkeen kuuteen osaan. Jokainen osa käsitteli TestDirectorin tiettyä osa-aluetta. Osa-alueet olivat TestDirectorin yleiset ominaisuudet, vaatimuksen määrittely, testauksen suunnittelu, testien ajo, virheiden seuranta sekä tulosten analysointi. Laadin asteikon 0 – 5, jonka perusteella toiminnallisuutta arvioitiin. Lomakkeessa oli myös oma tilansa kommenteille, johon vastaajat selvensivät ongelmaa sekä kertoivat toiminnallisuutta koskevia parannusehdotuksia.

Lähetin tekemäni lomakkeen Series 60:n TestDirector-tiimin vetäjälle tarkastettavaksi ja hänen korjausehdotuksiensa perusteella muutin lomaketta. Lähetin valmiin lomakkeen (liite 1) sähköpostitse postilistalliselle eli noin 60 henkilölle Series 60 testauksen työntekijöitä. Harmikseni suhteellisen pieni joukko vastasi kyselyyni, vain noin 33 prosenttia eli 22 kappaletta. Vastauksia tuli Tampereen sekä Helsingin toimipisteistä.

Huomasin vasta jälkeenpäin, että kyselylomakkeeni numeroasteikossa oli hiukan epäjohdonmukainen numerointi, mutta sain kuitenkin ongelmakohdat hyvin selville. Kyselyn avulla sain myös selvitettyä järjestelmän ominaisuuksia, joita testaajat harvoin käyttävät ja missä kokevat kehittämisen tarvetta.

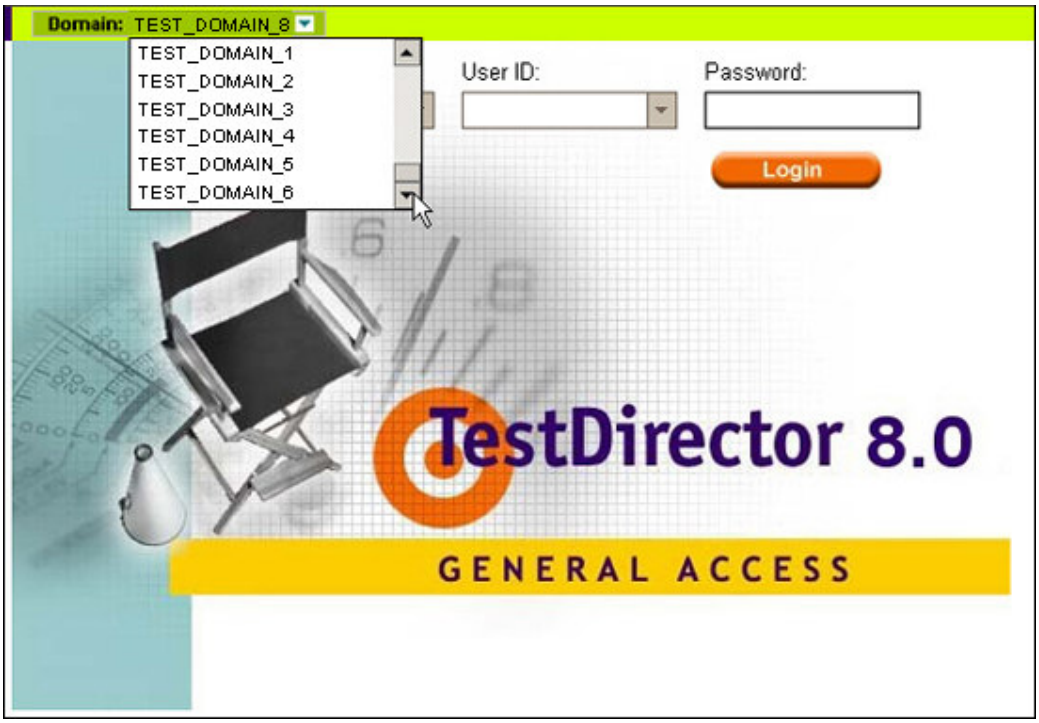
9. TestDirector-kyselystä tehdyt päätelmät

Käyn työssä lävitse kyselylomakkeen kohdat otsikoittain. Otsikoiden alle olen koonnut päätelmäni vastaajien vastauksista ja kommenteista.

9.1. TestDirectorin yleiset ominaisuudet

9.1.1. Kirjautuminen

Ensimmäinen kyselylomakkeen kohta koski TestDirectorin domainin ja projektien valintaa sekä kirjautumista järjestelmään. Ongelmaksi osoittautui domainin valintalista, jonka voi nähdä kuvassa 10.



Kuva 10. TestDirectorin kirjautumissivu

Vastaajien kommentit:

Valintalista on toteutettu ponnahdusikkunan avulla. Tällä hetkellä listasta on vaikea valita oikeaa domainia sillä, kun hiiren kursori eksyy sivuun ponnahdusikkunasta, ikkuna sulkeutuu välittömästi.

Kantojen rakenne on vastaajien mielestä vaikea. Oikeaa kantaa Series 60 – kantojen ulkopuolella saa etsiä kuin neulaa heinäsuovasta. Osaa alueista ei mahdoton osata yhdistää millään tavoin oikeaan domainiin, jos ei aivan tarkkaan tiedä missä jokin tietty projekti sijaitsee.

TestDirector-järjestelmän sisäänkirjautumisessa nähtiin muitakin ongelmia. TestDirector tarkistaa ja lataa erilaisia tiedostoja sisäänkirjautumisen yhteydessä. Tämä vie usein liian kauan aikaa. Moni käyttäjä odottaa pääsevänsä sisään järjestelmään sekuntien odotusajan jälkeen. Monesti kirjautuminen vie kymmeniä sekunteja, joskus jopa minuutteja.

Omat kommentit:

TestDirector voi heittää käyttäjän ulos järjestelmästä monesta syystä, esimerkiksi silloin, kun järjestelmä on ollut käyttämättömänä tietyn ajan. Usein tällaisissa tapauksissa käyttäjän yrittäessä kirjautua takaisin järjestelmään käyttäen samaa selainikkunaa, sisäänkirjautuminen epäonnistuu. Ainoa vaihtoehto päästä takaisin järjestelmään on sulkea kyseinen selain ja avata uusi.

Monesti käyttäjät eivät huomaa sulkea selainta yrittäessään kirjautua takaisin järjestelmään. Sisäänkirjautuminen ei onnistu kerta toisensa jälkeen. Tästä syystä käyttäjät menettävät usein malttinsa, sillä hän ei huomaa/ymmärrä sulkea selainta ja avata uutta. Monesti ongelmia kirjautumisessa aiheutuu myös silloin, kun käyttäjä yrittää kirjautua sisään järjestelmään käyttäen uutta selainikkunaa eikä huomaa sulkea vanhaa selainta taustalta.

Myös itse koen ongelmallisena Domain-ponnahdusikkunan. Hiiri on yritettävä pitää ponnahdusikkunassa, jotta ikkuna ei sulkeutuisi. Jos domain-lista on pitkä, on listaa rullattava alaspäin tai ylöspäin löytääkseen etsimänsä domainin. Tällöin hiiri eksyy helposti ikkunan ulkopuolelle ja ikkuna sulkeutuu. Ponnahdusikkunassa olisi hyvä olla muutaman sekunnin viive ennen kuin se katoaa näkyvistä. Näin hiiren eksyessä pois ikkunasta, käyttäjällä olisi mahdollisuus siirtää kursori takaisin ikkunaan ja jatkaa oikein domainin etsimistä eikä ikkunaa tarvitsisi avata uudelleen.

9.1.2. TestDirectorin datan kanssa työskentely

TestDirectorissa työskennellään erilaisten taulukoiden sekä tiedostorakenteiden parissa. Taulukoissa on mahdollista järjestää sarakkeita, suodattaa tallenteita erilaisin edellytyksin sekä tallentaa dataa tiedostoihin: Excel- ja Word-dokumentteihin, muistioon sekä HMTL-muotoon.

Datan eli lähinnä testitapausten tallentamista tekstitiedostoon ja HTML-dokumenttiin oli käyttänyt ainoastaan vajaa puolet kyselyyn vastaajista. Excel-taulukkoon sekä Word-dokumenttiin tallentamista oli käytetty tekstitiedostoon ja HTML-dokumenttiin verrattuna enemmän. Kuitenkin en saanut kommenttien avulla selville syytä toiminnallisuuden vähäiseen käyttöön.

Omat kommenttini:

Mietin mahdollisia syitä, miksi datan tallentamista eri tiedostomuotoihin käytetään niin vähän. Pidän tähän syynä sitä, että vastaajat eivät näe toiminnallisuutta tarpeellisena. Myös tiedostojen luettavuudessa olisi omasta mielestäni parantamisen varaa.

Nimittäin tallennettaessa testidataa HTML-dokumenttiin tulee dokumentista todella epäselvä. Tiedot tallentuvat taulukkomuodossa. Taulukon soluille ei piirre-

tä lainkaan ääriviivoja ja taulukosta tulee todella leveä, mikä tekee sen käytöstä hankalaa. Harvoin taulukko mahtuu kokonaisuudessaan tietokoneen näytölle.

Tekstitiedostomuodossa tekstit ovat esitetty kuin yhtenä suurena kirjainkasana. Tekstitiedostoa ei ole muutenkaan mahdollista muokata kovinkaan paljoa. HTML-dokumentti vaatisi muokkausta, jotta sitä viitsisi käyttää. HTML-dokumentin muokkaaminen on hankalaa, sillä dokumentti pitäisi koodata uudelleen.

Tallennettaessa testitapauksia Excel-taulukkoon on taulukkoa muokattava, jotta tiedot saisi luettavaan muotoon. Tekstit ovat epäselviä luettavaksi, sillä taulukon soluja ei ole suurennettu eikä levenny lainkaan. Excel-taulukkoa on kuitenkin huomattavasti helppo muokata verrattuna esimerkiksi HTML-dokumenttiin. Word-dokumenttiin tallennettaessa tiedot on esitetty valmiiksi selkeällä tavalla. Se on mielestäni ainoa tallennustapa, jota ei välttämättä tarvitse muokata tallentamisen jälkeen.

Tallenteiden suodattamisesta olen huomannut sen, että kaikissa suodattimen kohdissa ei ole valmiiksi annettua ehtoa. Tällaisessa tapauksessa on vaikeaa määritellä ehto oikein, jotta toivottu tulos saadaan näkyviin. Kun kenttä on täysin tyhjä eikä vaihtoehtoja ole, on minulle on usein epäselvää, minkälaista ehtoa kenttään kaivataan.

9.1.3. Liitteen lisääminen

Testiprosessin alusta loppuun asti on mahdollista liittää erilaisia liitteitä testattavan sovelluksen havainnoimiseen. TestDirectoriin voidaan liittää tiedosto, URL-osoite, tilannekuva, kuva leikepöydältä tai järjestelmätietoja. Kyselyn vastausten perusteella sain huomata, kuinka vähän liitteitä käytetään testauksen tukena.

Suurin osa, jotka olivat käyttäneet ominaisuutta, eivät olleet havainneet ongelmia liitteiden liittämisessä. Liitteiden näyttämisestä tuli jonkin verran kommentteja.

Vastaajien kommentit:

Vastauksissa koettiin turhauttavana myös se, että URL-osoitetta ei pysty tallentamaan. Series 60 tapauksessa lähes aina liite on URL-osoite. URL-osoite on usein myös niin pitkä, joten se ei näy liitenäkymässä kokonaan. Täydellistä osoitetta ei saa näkyviin millään muulla tavalla kuin avaamalla liitteen.

Omat kommentit:

On myönnettävä, että olen itsekin käyttänyt erittäin vähän liitteitä testien apuna. Syynä tähän on varmasti se, että liitteistä puhutaan sisäisissä ohjeissa hirveän vähän eikä ole tullut toivetta liitteiden käytöstä. Olen käyttänyt liitteenä ainoastaan URL-osoitetta. Havaitsin asiasta aivan samaa kuin vastaajat. Kun osoite on niin pitkä, ettei se mahdu näkymään, osoitetta ei saa näkyviin muulla tapaa kuin klikkaamalla liite auki ja lukemalla sen selaimen osoiteriviltä.

Olisi hyvä, jos liitenäkymässä liitteen osoite näytettäisiin kokonaan muun muassa silloin kun kyseinen osoite on valittuna. TestDirectorissa käytetään jo vastaavaa toiminnallisuutta muun muassa TestPlan-moduulissa, kun testitapauksen nimi ei näy kokonaan.

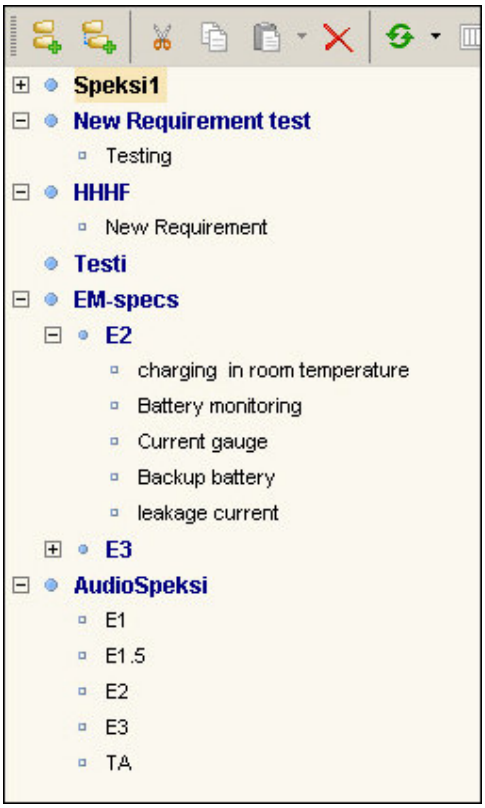
9.2. Vaatimusten määrittely

Vaatimukset kuvaavat yksityiskohtaisesti, mitä tarpeita testauksella on, ja määräävät perustan testaustiimille, johon eheä testausprosessi nojautuu. Vaatimusten kanssa ei näyttänyt vastausten perusteella olevan suurempia ongelmia. Testaajat eivät itse lisää eivätkä muokkaa vaatimuksia.

Vastaajien kommentit:

Requirements-moduulissa vaatimusten määrä on suuri ja ne on jaettu eri kansioiden alle. Tästä syystä vaatimusten etsiminen manuaalisesti voi olla aikaa vievää. Käytössä on onneksi Etsi-toiminto, minkä avulla vaatimuksen etsiminen käy huomattavasti helpommin.

Vaatimusten näyttämisessä Requirements-moduulissa olisi parannettavaa. Vaatimukset, joilla ei ole lapsi-vaatimusta, näytetään perustekstinä. Kun taas lapsivaatimuksen sisältävät vaatimukset sekä kansiot näytetään lihavoidulla tekstillä. Tämän voi nähdä kuvasta 11. Lihavoitu teksti eroaa liian paljon perustekstistä. Saman tason vaatimukset pitäisi näyttää samalla tavalla, joko perustekstinä tai lihavoituna.



Kuva 11. Vaatimusten esittämistapa

Vaatimuksiin liittyvien virheiden näyttämiseen ei oltu tyytyväisiä. Ongelmana oli virhedialogin tekstit tai lähinnä se, että tekstejä ei pysty valitsemaan dialogista. Series 60 -ympäristössä olisi lähes välttämätöntä, että tekstejä tai lähinnä virheen tunnisteen pystyisi kopioimaan listasta. Näin käyttäjät pystyisivät paremmin hyödyntämään kyseistä näkymää. He pystyisivät muun muassa helpommin etsimään ja tarkastelemaan virheitä erillisessä virhetietokannassa.

Omat kommentit:

Kun yllämainitusta virhenäkymästä ei pysty kopioimaan tietoja tietokoneen leikepöydälle, on käytettävä kynää ja paperia, jotta saisi virheen tunnisteen talteen ja siirrettyä sen muun muassa erilliseen virhetietokantaan tai Defects-moduuliin. Näkymässä ei myöskään pysty muokkaamaan näytettäviä kolumneja eikä muutenkaan tekemään mitään muuta kuin katselemaan tietoja. Olisi hyödyllistä, jos virheessä olisi esimerkiksi linkki, josta pääsisi siirtymään Defects-moduuliin kyseisen virheen tietoihin.

9.3. Testaussuunnitelma

9.3.1. Testaussuunnitelmapuun kokoaminen

Tyypillinen ohjelma on liian laaja testattavaksi yhtenä kokonaisuutena. Test Plan –moduuli mahdollistaa sovelluksen jakamisen toiminnallisuuden perusteella. Testaussuunnitelmapuun luomisessa ja uuden testitapauksen lisäämisessä ei ilmennyt ongelmia. Ihmetyksen aiheeksi nousi se, miksi testaajan ominaisuudet omaavalla käyttäjällä ei ole oikeuksia poistaa kansioita.

Vastaajien kommentit:

Testaussuunnitelmapuun muokkaamiseen tai lähinnä suodattamiseen tuli vastaajilta parannusehdotus. Puuta olisi hyvä saada suodatettua niin, että näkyvissä olisi ainoastaan ne kansiot, mitä käyttäjä haluaa ja mitä eniten käyttää. Testisuun-

nitelmapuu on nimittäin niin suuri ja laaja, eikä läheskään kaikkia kansioita tarvitse koskaan. Tällä tavoin saataisiin pois näkyvistä itselle tarpeettomat kansiot. Suodattimellahan saa näkyviin ainoastaan halutut testitapaukset. Silloinkin jää näkyviin näytetään kaikki kansiot, vaikka ne eivät sisältäisi suodattamisen jälkeen ainuttakaan testitapausta.

Myös oikeuksien puuttuminen on todella turhauttavaa. Ne, joilla on oikeuksia poistaa kansioita, kokivat, että kansioiden poistossa on ongelmia. Kun kansion poistaa ja näkymä päivitetään, kansio palaa takaisin. Kansion voi joutua poistamaan useamman kerran ennen kuin se todella poistuu.

Omat kommentit:

Itse en omista oikeuksia kansioiden poistamiseen. Monesti olen luonut kansion ja todennutkin sen turhaksi. Olisin halunnut poistaa kansion, mutta siihen ei ole ollut mahdollisuutta. Kansio on ajelehtinut siellä täällä ja odottanut aikaa, jolloin sitä voisi hyödyntää. Tietysti voisi pyytää riittävät oikeudet omaavaa käyttäjää poistamaan ylimääräisen kansion. Kuitenkin kansion poistaminen sujuisi paljon yksinkertaisemmin, jos poistamisen voisi tehdä itse.

Testitapausta suunniteltaessa Test Plan –moduulissa testi voidaan merkitä eri suunnitteluvaiheiden mukaan joko Draftiksi, Proposaliksi, Approvediksi tai Designiksi. Minua on ihmetyttänyt, miksi testaajan oikeudet omaavalla käyttäjällä ei ole oikeuksia poistaa muita kuin Draft-tilassa olevia testitapauksiaan. Kuitenkin Design-tilassa sekä Proposal-tilassa olevat testitapaukset pystyy muuttamaan Draft-tilaan ja poistamaan tätä kautta. Miksi ei siis olisi oikeuksia poistaa Design- sekä Proposal-tilassa olevia testitapauksia suoraan.

9.3.2. Testien linkittäminen vaatimuksiin

Testausprosessi alkaa vaatimusten määrittelyllä. Test Plan –moduulissa kootaan testaussuunnitelmapuu, johon luodaan testitapaukset. Pystyäkseen seuraamaan

vaatimusten ja testien suhdetta, ne on linkitettävä toisiinsa. Vaatimusten linkittämisessä testitapaukseen ja testitapauksen linkittämisessä vaatimukseen ei ilmennyt ongelmia.

Vastaajien kommentit:

Linkityksen poistamisessa kummasteltiin sitä, miksi usean eri vaatimuksen tai usean testitapauksen linkitystä ei voida poistaa samanaikaisesti vaan jokainen linkki on poistettava yksitellen.

9.3.3. Testien laatiminen

Testejä laadittaessa testiin määritellään testiaskelmat. Askelmiin tulee yksityiskohtainen askel-askeleelta selvitys, kuinka testi suoritetaan. Askelmat sisältävät toiminnan kuvauksen, annettavat syötteet sekä odotetut tulosteet. Testiaskelman luomisessa ei kyselyn mukaan ollut ongelmia. Testiaskelman muokkaamisessa ilmeni muutama ongelmallinen toiminto. Noin puolet vastaajista oli käyttänyt Etsi-toimintoa etsiäkseen tekstejä testiaskelmista.

Vastaajien kommentit:

Nyt TestDirectorissa testiaskelmia luotaessa ensimmäinen askelma on Step 1, toinen askelma Step 2, jne. Series 60:n testauksessa ensimmäisenä askeleena käytetään esitieto- eli Precondition(s)-kenttää, toisena askelmana on Step 1, kolmantena Step 2, jne. Askelmien luominen kävisi helpommin, jos TestDirectorissa askelmien nimet olisivat oletuksena, kuten Series 60 -ohjelmistoalustan testauksessa on tapana nimetä testiaskelmat. Tällä hetkellä Precondition(s)-kenttä on luotava manuaalisesti ja aina kun uuden askelman lisää, on kentän nimeä muutettava.

Joskus testiaskelmaan tehdyt muutokset eivät tallennu. Aluksi muutos on näkyvissä, mutta jos kirjautuu ulos järjestelmästä ja taas takaisin järjestelmään, saat-

taa olla, että tehty muutos katoaa ja TestDirector palauttaa automaattisesti vanhan tekstin askelmaan.

Testiaskelmien uudelleenjärjestäminen tuntuisi toimivan välillä hiukan omavaltaisesti. Raahatessa testiaskelmaa toisen askelman yli, käyttäjä ei voi koskaan tietää, sijoittuuko raahattu askelma ennen vai jälkeen toista askelmaan.

Testiaskelmien uudelleennumerointi ei toimi aivan halutulla tavalla johtuen edellä mainitusta Precondition(s)-kentästä. Uudelleennumerointi ei huomio Precondition(s)-kenttää lainkaan vaan järjestää sen ensimmäiseksi askelmaksi, Step 1:ksi. Tällaisen toiminnallisuuden takia uudelleennumerointi on hyödytön, sillä tämän jälkeen on muutettava jokaisen askelman numerointi manuaalisesti. Eräs vastaajista oli toiminut uudelleennumeroinnin kanssa niin, että hän oli siirtänyt Precondition(s)-kentän aivan viimeiseksi askelmaksi ennen uudelleennumerointia. Täten hänen täytyi muuttaa manuaalisesti ainoastaan viimeisen kentän tieto takaisin Precondition(s)-nimiseksi ja siirtää se oikealle paikalleen.

Testiaskelman koon muokkaaminen ei näytä tallentuvan niin, että se säilyisi uudelleenkirjautumisessa. Kaikkia askelman tekstejä ei aina näytetä koon muokkaamisen jälkeen. Joskus kun askelman kaikki tekstit eivät ole näkyvissä ja askelman kokoa yrittää muuttaa, ei muutos astu voimaan vaan muutosta edeltänyt tila näytetään.

Tekstin etsiminen testiaskelmista koettiin turhana, sillä haulla pystyy etsimään tekstejä ainoastaan yhdestä ainoasta testitapauksesta kerrallaan. Samalla vaivalla silmäilee kyseisen testitapauksen läpi löytääkseen hakemansa.

Tekstin Korvaa-toimintaa pidetään riskialttiina, sillä TestDirector ei osoita millään tavalla, mihin muutokset on tehty. Etsi ja korvaa –toiminnallisuus tarvitsisi toteuttaa samalla tavalla kuin Microsoft Wordissa. Jokainen tehty muutoskohta näytettäisiin ja varmistettaisiin, halutaanko kyseisen muutoksen tehdä. Sillä tavoin Korvaa-toiminto toimisi parhaiten.

Omat kommentit:

Monesti itsestäkin tuntuu turhauttavalta muokata testiaskelmia. Saa miettiä, tallentuuko muutos vai ei, joutuuko sen tekemään jossain vaiheessa uudelleen.

Olen törmännyt useamman kerran ongelmaan, että testitapaus on lukittuna omalle käyttäjätunnukselle. Tällaisessa tapauksessa ei pysty muokkaamaan omaa testitapaustani. Virheilmoitus objektin lukituksesta tulee aina näkyviin, kun yritän tehdä muutoksia. Tällöin on kirjauduttava ulos TestDirectorista. Sekään ei ole aina auttanut vaan jossain tapauksissa minun on kirjauduttava kokonaan ulos koneelta tai pahimmassa tapauksessa käynnistettävä tietokone uudelleen.

9.4. Testien suorittaminen

9.4.1. Testisetin luominen

Testisetti on joukko testejä, joiden tulisi tavoittaa tietyt tavoitteet. Testisetin luomisessa, testien liittämisessä testisettiin, testien poistossa testisetistä, testisetin kopioimisessa ja uudelleen nimeämisessä yli puolissa vastauksissa ei ilmennyt ongelmia suorittaa toimintoa. Kansiota sekä testisettiä ei testaajan oikeudet omaava pysty poistamaan. Tämä harmitti useaa vastaajaa.

Testisetin nollaus muuttaa kaikki setin testit No run –tilaan eli tilaan, jossa testejä ei ole ajettu. Kyseistä toimintoa oli käyttänyt ainoastaan neljäsosa käyttäjistä.

Omat kommentit:

Syynä testisetin nollauksen käyttämättömyyteen on varmastikin se, että jokaisella ajokerralla on Series 60:n ohjeiden mukaisesti luotava uusi testisetti eikä samaa testisettiä voi ajaa uudelleen. Toiminnon käyttö varmasti lisääntyy toimintaohjeiden muuttumisen myötä.

Monesti kopioin toisen testisetin, jota lähden muokkaamaan. Tällöinhän testisetin tietoihin jää alkuperäisen setin yksityiskohdat. Testaajan oikeuksilla minulla ei edes ole mahdollisuutta vaihtaa setin tietoja. Tästä syystä minun pitää jokaisen ajamani testitapauksen kohdalla muuttaa testin ajon tietoja. Kaipaisin oikeuksia setin tietojen muuttamiselle sekä toimintoa, joka päivittää kaikkiin kopioidun testisetin testitapauksiin päivitetyt tiedot.

9.4.2. Testien ajaminen manuaalisesti

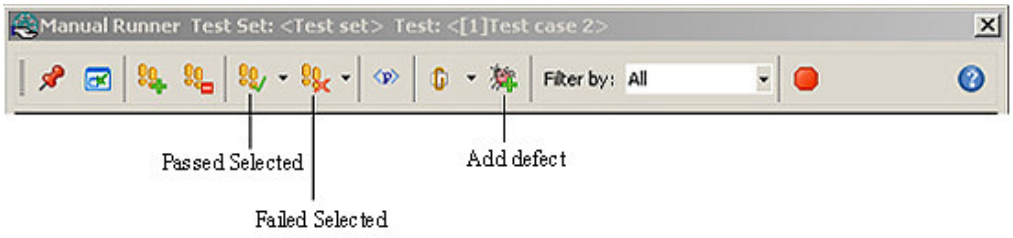
Kun testi ajetaan manuaalisesti, seurataan testiaskelmia ja suoritetaan toimintoja sovelluksessa testin ajon aikana. Odotettuja tuloksia verrataan todelliseen lopputulokseen ja tulokset kirjataan ylös.

Yksittäisen manuaalisen testin ajossa vastaajat olivat sitä mieltä, että testien ajossa ei ole oikeastaan minkäänlaisia ongelmia. Joitain kommentoitavia toiminnallisuuksia manuaalisesta ajosta kuitenkin löytyi.

Vastaajien kommentit:

Ennen testin ajoa on päivitettävä manuaalisesti testitapausten tyhjät kentät testisetin tiedoilla nappia painamalla. Osa vastaajista on sitä mieltä, että toiminnon pitäisi olla automatisoitu, sillä testiä ei pysty ajamaan ennen kuin testitapausten tyhjät kentät on täytetty.

Kuvassa 12 on Manual Runner –ikkunan yläreunassa olevat painikkeet. Joku vastaajista oli sitä mieltä, että Passed Selected – , Failed Selected – sekä Add defect –painikkeet pitäisi olla paljon suuremmat syystä, että niitä käytetään kaikkein eniten tai erotettuna jollain muulla tavalla omaksi ryhmäkseen.



Kuva 12. Manual Runner –ikkunan painikkeet

Yksi syy testiaskelmien lisäämättömyyteen testin ajon aikana on se, että uusi askelma luodaan aina viimeiseksi askelmaksi eikä ajon aikana askelmien järjestystä ei voi muuttaa. Uuden askelman lisääminen on lähes tarpeetonta siinä tapauksessa, jos uusi askelma haluttaisiin lisätä muualle kuin viimeiseksi askelmaksi. Testaajan oikeudet omaavalle käyttäjälle ei ole kuitenkaan oikeutta poistaa askelmia ajon aikana.

Testin ajon aikana on mahdollista tehdä muutoksia testiaskelmien sisältöihin. Testiin tehdyt muutokset voidaan tallentaa vasta kun testin ajo lopetetaan. Jos käyttäjä ei omista riittäviä oikeuksia muutosten tallentamiseen, kaikki tehdyt muutokset katoavat, kun testin ajo lopetetaan. Jos muutoksia ei olekaan mahdollista tallettaa, on käyttäjä nähnyt turhaa vaivaa muutosten tekemiseen. Joskus testiaskelmiin ajon aikana tehtyjä muutoksia ei ole ollut oikeuksia tallettaa. Oikeudet testien muokkaamiselle tulisi säilyttää, sillä se on monesti paras tapa pitää testit ajantasalla.

Kun testattavassa sovelluksessa testin ajon aikana havaitaan epäkohta, luodaan siitä virheraportit erilliseen virhetietokantaan ja TestDirectoriin. Virheiden raportointi pitäisi saada hoidettua yksikertaisemmalla tavalla. Lisää kommentteja virheiden raportoinnista sekä päivittämisestä löytyy kohdasta 9.5. Virheiden seuranta.

Ajettaessa testejä asetetaan yksittäin testiaskelma tai koko testitapaus tuloksen mukaan joko Passed, Failed, Not Completed tai N/A eli testiä ei voi käyttää kyseissä tilanteessa. Tällä hetkellä testaaja ei pysty valitsemaan tilaa N/A. Testaaja

ei myöskään pysty muuttamaan testitapauksen statusta muuten kuin ajamalla testitapauksen. On kuitenkin tapauksia, jolloin status olisi järkevää muuttaa testisetin testitapauslistalta.

9.4.3. Testitulosten näyttäminen

Testien suorittamisen jälkeen, tulokset voidaan nähdä TestDirectorissa. Testitulokset sisältävät kaikki hyväksytty/hylätty-statuksella olevat testit sekä testiaskelmat. Test Run Properties –valintaikkunassa näytetään testin suorittamisen yksityiskohdat, verrataan uusinta testisuoritusta edellisiin suorituksiin, hallinnoidaan liitteitä ja näytetään testin suorittamisen muutoshistorian.

Lähes jokainen, muutaman vastaajaa lukuun ottamatta, oli käyttänyt Test Run Properties –valintaikkunaa. He eivät kokeneet valintaikkunaa ongelmallisena. Kuitenkin tuli parannusehdotus ikkunan Details of the Test Run –kohtaan. Kyseisestä näkymästä ei pysty lisäämään uutta virhettä TestDirectoriin, mikä olisi tarpeen.

9.5. Virheiden seuranta

Kun testaaja löytää virheen sovelluksesta suorittaessaan testiä, hän tekee virheraportoinnin TestDirectorin projektille. Virhetallenteet informoivat muita testaajia sekä kehittäjiä löydetyistä virheistä.

Virheraportointiin vastaajat eivät olleet kovinkaan tyytyväisiä. Tämän hetkinen toiminnallisuus toimii sellaisenaan hyvin, mutta siinä olisi parantamisen varaa.

<u>Vastaajien kommentit:</u>

Vastaajat ovat turhautuneet siihen, että virheraportointi pitää tehdä kahteen eri paikkaan: erilliseen virhetietokantaan sekä TestDirectorin Defects-moduuliin. Kun virheen tila muuttuu erillisessä virhetietokannassa, on tieto päivitettävä ma-

nuaalisesti TestDirectorissa. Muuten TestDirectorin virhetiedot eivät vastaa todellista virheen tilaa.

Kun virhe on korjattu, testit, joissa virhe ilmeni, on ajettava uudelleen ja luotava uusi virheraportti, jos virheitä vielä löytyy. Olisi hyvä, jos nykyistä virheraporttia pystyisi muokkaamaan ja muuttamaan virheen tiedot uusinta tietoa vastaavaksi. Sellaiseen ei kuitenkaan normaalit testaajanoikeudet omaavalla käyttäjällä ole oikeuksia. Pieni muutos saa aikaan hirveän määrän työtä, joka vie turhaa aikaa.

Hankalaa on myös se, että jokainen löydetty virhe on raportoitava jokaisessa testitapauksessa erikseen. Olisi todella hyödyllistä, jos yhteen virheeseen pystyisi linkittämään useamman testitapauksen.

Omat kommentit:

Virheraportin poistoonkaan ei ole oikeuksia, mikä on inhottavaa. Itse olen tehnyt virheraportoinnin täysin väärään testitapaukseen tai liittänyt testitapaukseen täysin väärän virheen. Tällaisissa tapauksissa toivoisi, että virheellisen virheen voisi poistaa.

9.6. TestDirector analyysi

9.6.1. Raporttien luominen

TestDirectorin avulla luotavat raportit auttavat arvioimaan määriteltyjä vaatimuksia, niiden testikattavuutta, testisuunnitelmia, testien suorittamista sekä virheiden seurantaa. Raportin voi luoda jokaisesta TestDirectorin moduulista.

Vastaajat kokivat raportin luonnin liian sekavana. Varmasti tämä on syynä sille, että ainoastaan vajaa puolet olivat luoneet raportteja. Ne, jotka niitä olivat luo-

neet, eivät maininneet mitään erityisiä ongelmia raportin luonnissa tai raportin muokkaamisessa. Aliraportteja olivat luoneet ainoastaan kolme vastaajaa.

Omat kommentit:

Vastausten perusteella voisin päätellä, että testaajien ei tarvitse luoda eikä käyttää kyseisenlaisia raportteja, sillä niin pieni osa vastaajista oli niitä käyttäneet. Raportteja luovat huomattavasti enemmän leaderit ja managerit. Itselläkään ei ole 1,5 vuoden TestDirectorin käytön aikana ollut tarvetta käyttää kyseistä ominaisuutta.

9.6.2. Diagrammien luominen

TestDirectorin diagrammit auttavat tekemään johtopäätöksiä nopeasti ja niiden avulla näkee erilaisten tietojen suhteet projektissa. Diagrammin voi luoda missä testausprosessia vaiheessa tahansa. Vastaajat olivat käyttäneet diagrammien luontia yhtä vähän kuin raporttien luontia.

Vastaajien kommentit:

Erään vastaajan mielestä, joka oli toiminnallisuutta käyttänyt, diagrammeille pitäisi pystyä asettamaan käytettävien värien oletusarvot, kuten esimerkiksi punainen väri kuvaisi hylättyjä testitapauksia ja vihreät hyväksyttyjä.

Omat kommentit:

Testaajana voin sanoa, että harvoin tulee tilanteita, joissa olisi ollut tarvetta luoda diagrammeja. Diagrammien luontia käyttävät hyväkseen enemmänkin leaderit sekä managerit kuten raporttien luomistakin. Ominaisuudesta on muutenkin ollut todella vähän mitään mainintaa, joten se saattaa olla monella aivan vieras ja täten käyttö on jäänyt vähemmälle.

9.6.3. Projektidokumentin luominen

TestDirectorin dokumenttigeneraattorilla voidaan luoda Microsoft Word -dokumentteja, joihin voidaan sisällyttää tietoa projektin vaatimuksista, suunnitelmista, testilistasta, testien suorittamisesta sekä virheiden seurannasta.

Kyseistä toiminnallisuutta testaajat olivat käyttäneet huomattavasti enemmän kuin raporttien ja diagrammien luontia. Tätä toiminnallisuutta tarvitaan, kun halutaan saada suuri määrä testitapauksia Word-dokumenttiin esimerkiksi testitapausten tarkastelua varten.

Vastaajien kommentit:

Vastauksista sain lukea, kuinka moni piti dokumenttigeneraattorin käytettävyyttä huonona ja toiminnallisuutta liian sekavana. Vastaajat olivat generoineet enimmäkseen Test Plan –moduulidataa. Eri kohtien ja kansioiden valinta toimii vastaajien mielestä oudosti. Valmiissa dokumentissa näkyy kaikki mahdolliset yläkansiot, joiden alla testitapaukset sijaitsevat. Syntyy pitkä lista otsikoita ja numerointeja, jos testitapaukset sijaitsevat useamman kansion alla.

Halutun tiedon määritteleminen on hiukan monimutkaista eri moduuleissa. Sen huomaa siitä, että usealle on syntynyt dokumentti, joka sisältää tietoa, mitä dokumenttiin ei olisi halunnut tulevan.

Dokumenttigeneraattorin käyttö on hidas prosessi. Kun dokumentista tulee vääränlainen ja on määriteltävä ja luotava dokumentin uudelleen, saa tähän kulutettua rutkasti aikaan. Moduulien määrittelyyn kaivattiinkin käytettävyysparannuksia, jotta dokumentin luominen olisi helpompaa ja yksinkertaisempaa.

Yksi ihmetyksen aihe oli dokumentin luomisen yhteydessä se, miksi aina aukeaa kaksi Word-dokumenttia, varsinaisen sekä tyhjän dokumentin.

Omat kommentit:

Olen itsekin kummastellut tyhjää dokumenttia. Aina kun yrittää generoida uutta dokumenttia, tulee ilmoitus, että kaikki Word-dokumentit tulisi sulkea ja tietysti aina löytyy se tyhjä dokumentti taustalta.

Valmiissa dokumentissa on pitkä lista otsikoita ja numerointeja, jotka häiritsevät. Joudun aina muokkaamaan dokumenttia niin, että siitä saa luettavamman ja selkeämmän sekä turhat otsikoinnit ja numeroinnit pois.

9.7. Muut kommentit

Kyselylomakkeessa oli myös kenttä, johon vastaajat saivat kirjoittaa vapaasti kommenttejaan aiheesta. Kommenteissa tuli parannusehdotuksia sekä ongelmia, mitä kysymyslomakkeessa ei muualla tullut esille.

Vastaajien kommentit:

Suurin osa kommenteista liittyi TestDirectorin suoriutumiskykyyn sekä vakauteen. Usea kommentti kuuluikin: ”TestDirector kaatuu erittäin usein” tai” on todella hidas”. Käyttäjälle tulee hitaudesta vaikutelma, että hän tuhlaa hyödyllistä työaikaansa odotteluun.

Ehdotuksia tuli myös siitä, mikä voisi lieventää TestDirectorin palvelimen kaatuilua. Olisi erittäin käytännöllistä toteuttaa sisäänkirjaamisen ja uloskirjaamisen toiminnallisuus TestDirectoriin. Testaaja voisi kirjata testejä ulos testejä suorittaakseen. Testit voitaisiin lukita uloskirjaamisen ajaksi. Kun testi kirjataan takaisin sisään, ajon yksityiskohdat voitaisiin päivittää. Tämä voisi ehkäistä jatkuvaa yhteyden tarvetta TestDirectorin palvelimeen ja täten lieventää palvelimen kuormitusta.

Palvelimen epävakauden lisäksi yhdeksi suureksi ongelmaksi nousi raportoinnin alkeellisuus. Kuten useissa ohjelmistopaketeissa, myös TestDirectorissa, raportointi tuntuu olevan ainoastaan myyntikikka Mercurylle, eikä yhtään sen enempää. Perusraportointi TestDirectorilla on mahdollista, mutta edistyksellisempi raportointi jää TestDirectorilla tekemättä.

Raportin luontimahdollisuuksia pidetään liian rajallisina eikä tarpeeksi joustavina. Suurena ongelmana eräs vastaaja koki sen, että TestDirector ei tue testitulosten raportointia siten, että käyttäjällä olisi mahdollisuus muokata raporttia mielensä mukaan. Järjestelmä ei mahdollista testaajien oikeudet omaavan käyttäjän muokata haluamiaan raportteja vapaasti. Moni tarvittava raportti ei ole käytettävissä ja raportit, jotka kehitetään raportointia varten, voi luoda ainoastaan TestDirectorin tukitiimi.

Hämäännystä herätti myös toisen testaajan tekemien testitapausten kopioiminen. Testien kopioiminen on hyvä ominaisuus, mutta siinä on myös ongelmansa. Kopioidussa testissä muutosoikeudet pysyvät alkuperäisen testin omistajalla. Kopioituun testiin ei siis saa tehtyä omia muutoksia eikä myöskään testiä pysty poistamaan. Olisi hyvä, jos kopioituun testiin tulisi omistajaksi testin kopioija. Näin toisen tekemistä testistä saisi kaikkein parhaiten hyödyn irti ja säästyisi ylimääräiseltä työltä.

Muista ohjelmista tuttuja pikanäppäimiä kaivatiin testitapausten luonnissa ja muokkauksessa. Tällaisia pikanäppäimiä ovat muun muassa Ctrl + B eli tekstin lihavointi ja Ctrl + I eli tekstin kursivointi. Tämän toteuttaminen voi olla hankalaa, sillä TestDirector aukeaa internet-selaimeen ja selaimella on omat pikanäppäimensä.

Erääksi, muista poikkeavaksi ongelmaksi yksi vastaajista mainitsi etätyömahdollisuuden puuttumisen. Moni saattaisi kaivata kyseistä mahdollisuutta, jotta töitä pystyisi tekemään muuallakin kuin työpaikalla. Esimerkiksi jos eteen tulisi tilanne, jossa töitä olisi tehtävä vaikkapa kotona.

Omat kommentit:

Tällä hetkellä kopioimalla toisen testejä saa vain haittaa aikaiseksi yllä mainitusta syystä. TestDirectoriin muodostuu ylimääräisiä ja käyttökelvottomia testitapauksia, jotka ajelehtivat paikasta toiseen ellei käyttäjä pyydä paremmat oikeudet omaavaa käyttäjää poistamaan kopioidut testaustapaukset.

Minua harmittaa TestDirectorin puutteellinen monivalintamahdollisuus. Test Plan –moduulin puolella testisuunnitelmapuusta ei voi valita kuin yhden testitapauksen kerrallaan. Minun on monta kertaa pitänyt siirtää useampaa testitapausta toiseen paikkaan. Olen joutunut siirtämään jokaisen testin yksitellen, koska useamman testitapauksen monivalinta ei ole mahdollinen. Yritän aina uudelleen ja uudelleen valita useampaa testitapausta, kun ei mieleni sopukoihin mene tieto siitä, että monivalinta ei ole mahdollista Test Plan -moduulissa.

TestDirectorin toimintaan kuuluu se, että muutokset tallentuvat heti palvelimelle muutoksen teon jälkeen. Erillistä tallennusta ei tarvitse suorittaa. Joskus kuitenkin toivoisin, että testitapauksia suunniteltaessa ja muokatessa tallentamista ei tapahtuisi. Nyt jos tekee virheen, ei ole paluuta aikaisempaan testitapauksen versioon. Jos tallentaminen tapahtuisi silloin, kun itse haluaa, olisi mahdollisuus saada käsiinsä viimeksi tallennettu versio testitapauksesta.

10. Yhteenveto

Koska Series 60 –ohjelmistoalusta on maailman johtava älypuhelinalusta, odotetaan alustan ohjelmistoilta laadukkuutta ja virheettömyyttä. Mahdollisimman suuri määrä virheitä pyritään löytämään ja korjaamaan ennen alustan lisensioimista asiakkaille. Suurin osa virheistä havaitaan testauksen aikana.

Ohjelmiston sovellusten testausta tekee ensin sovelluksen koodaaja ja tämän jälkeen henkilö, joka ei ole tunne sovelluksen koodia. Koodaaja ei millään kykene testaamaan omia moduuleita niin hyvin, että virheiden määrä saataisiin minimaaliseksi. Series 60 –ohjelmistoalustan testaajat suorittavat ohjelmistolle, koodaajan tekemän moduulitestauksen lisäksi, toiminnallisuustestausta, regressiotestausta, suorituskykytestausta, kielivarianttitestausta sekä käyttötestausta.

Testausprosessissa testit suunnitellaan ja määritellään annettujen vaatimusten mukaisesti. Testien ajon aikana löytyy suurin osa virheistä. Testausprosessi on yleensä niin laaja, että sen hallinta vaatii erillisen järjestelmän testien luotia, ajoa ja tulosten seurantaa varten. Series 60 –ohjelmistoalustan testauksessa on käytössä Mercury Interactiven TestDirector-testauksenhallintajärjestelmä.

Yleensä mitkään ohjelmistot eivät toimi aivan täydellisesti eivätkä tyydytä käyttäjän tarpeita. Näin on käynyt myös TestDirectorille. Työni toimeksiantona minun tuli selvittää TestDirectorin ongelmia testaajan näkökulmasta sekä selvittää mahdollisia parannusehdotuksia.

Lähetin kyselyn sähköpostitse noin 60 henkilölle. Sain vastauksen 22 henkilöltä. Olisin toivonut useampia lähetettyjä vastauksia kyselyyni. TestDirectorista löydettiin useita puutteita ja ongelmia, mihin itsekin olen testaajan työssäni törmännyt TestDirectorin kanssa.

Opinnäytetyöhöni olen koonnut yhteenvedon saamistani vastauksista, jonka perusteella Series 60:n sisällä voidaan pohtia, mitä tehdä käyttäjien kertomien on-

gelmien kanssa. TestDirectoriin on mahdollista lisätä uusia toiminnallisuuksia sen API rajapinnan kautta. Voi kuitenkin olla, että pahimmat ongelmat eivät korjaannu tätä kautta vaan järjestelmän valmistajan, Mercury Interactiven tulisi korjata TestDirectorin rakennetta.

Vastausten perusteella suurimmaksi TestDirectorin ongelmaksi osoittautui järjestelmän epävakaus. Epävakauden ja järjestelmän kaatumisen vuoksi testaajien työnteko keskeytyy ja täten hyödyllistä työaikaa menee hukkaan.

Toinen, ei niinkään ongelma, mutta kommentteja herättänyt asia oli oikeuksien vähäisyys. Joidenkin toimintojen suorittamiseen ei testaajan käyttöoikeuksilla ole mahdollisuutta. Tällaisia toimintoja ovat muun muassa kansion, virheraportin ja testisetin poistaminen. Tästä syystä TestDirectoriin kertyy turhia testisettejä sekä kansioita oikeuksien puuttumisen myötä. Ymmärrän myös sen, miksi oikeuksia on rajoitettu. Jossain tapauksissa testaaja voi ns. väärinkäyttää oikeuksiaan ja poistaa tapauksia, joita ei saisi poistaa.

Kummastusta ja ärsytystä herätti myös virheraportoinnin teko kahteen eri paikkaan: erilliseen virhetietokantaan sekä TestDirectoriin sekä niiden erillinen päivittäminen. Jokaisella uudella ajokerralla on TestDirectoriin lisättävä uusi virheraportti sekä suljettava edelliskierroksen virheet. Näin saattaa samaa testitapausta kohden olla useita virheraportointeja, jotka viittaavat yhteen ja samaan virheeseen. Paras vaihtoehto olisi, että TestDirectorin virheen tietoja saisi päivitettyä ja että yhteen TestDirectorin virhetallenteeseen voisi liittää useamman testitapauksen.

Neljänneksi esille nousi erilaisten raporttien ja dokumenttien laatiminen. Kaavioiden ja raporttien laatiminen on liian rajallista eikä riittävän joustavaa. TestDirectorilla on mahdollista suorittaa ainoastaan perusraportointia. Käyttäjä ei pysty vapaasti muokkaamaan raportista haluamakseen vaan hänen on tyydyttävä TestDirectorin raportti- sekä kaaviopohjiin. Dokumenttigeneraattorin avulla luotavien Word-dokumenttien laatimista pidettiin myös liian sekavana ja hitaana.

Käytön sekavuudesta kertoo sekin, että käyttäjät harvoin saavat laadittua ensimmäisellä yrityskerralla dokumentin, joka sisältö on sitä mitä he todellisuudessa haluavat.

Luovutan kyselyni vastaukset sekä saadut kommentit Series 60 – ohjelmistoalustan TestDirectorista huolehtivalle tiimillä. Kommenttien pohjalta tiimi voi alkaa pohtimaan, mitä TestDirectorin häiritseville tekijöille voitaisiin tehdä.

Jos raportissa mainitut toiminnat saataisiin korjattua sekä parannuksia lisättyä TestDirectoriin, varmasti käyttäjät olisivat järjestelmään huomattavasti tyytyväisempiä. Kuitenkin aina jokaisesta järjestelmästä löytyy kommentoitavaa ja parannettavaa. Korjaukset ja parannukset varmasti lisäävät mielipiteitä ja siirtävät käyttäjien huomion muihin toiminnallisuuksiin.

Itse olisin ajan salliessa laatinut kyselyni pohjalta jatkokyselyn, jossa olisin voinut tiedustella tarkemmin kyselyn kommenteista ja mahdollisista syistä esimerkiksi jonkin tietyn ominaisuuden käyttämättömyyteen. Olen kuitenkin tyytyväinen ensimmäisen ja ainoan kyselylomakkeeni antiin.

Lähteet:

Dutch encouragement 2003. [online] [viittaus 06.03.2005]. http://www.professionaltester.com/archive/october2003/ProTesterOct2003-VanVeenendaal.pdf

Haikala, Ilkka. Märijärvi, Jukka 2004. Ohjelmistotuotanto. Hämeenlinna: Karisto Oy.

Mercury TestDirector 2005. [online] [viittaus 14.09.2005]. http://www.mercury.com/us/products/quality-center/testdirector/integrations.html

Mercury Interactive 2003. Implementing an effective test-management process. Kurssimateriaali 06.04.2004. Tampere.

Mercury Interactive 2003. TestDirector, User's Guide, Version 8.0.

Nguyen, Hung Q. 2003. Testing Applications on the Web, test planning for Mobile and Internet-based Systems. Indianapolis: Wiley Publishing, Inc.

Ohjelmistotestauksen kehittäminen 2002. Kurssimateriaali. Tieturi.

Ohjelmistotestaus 2004. [online] [viittaus 23.11.2004]. http://gallia.kajak.fi/opmateriaalit/yleinen/ohjelmistotuotanto/ohjetuot/Testaus/materiaalit.html. Kurssimateriaali. VirtuaaliAMK.

Series 60 Platform 2005. [online] [viittaus 23.02.2005]. http://www.series60.com

Series 60 Platform Product Overview 2004. [online] [viittaus 10.01.2005]. http://www.series60.com/pics/pdf/Series_60_Platform_Product_Overview_June04.pdf

Series 60 Testing Process 2005. Nokia Corporation.

Software Test Management – Mercury TestDirector 2005. [online] [viittaus 01.03.2005]. http://www.mercury.com/us/products/quality-center/testdirector/

Software Testing ISEB Foundation Certificate Course 2003. Kurssimateriaali.

Tamres, Louise 2002. Introducing Software Testing. Lontoo: Addison-Wesley.

Tevanlinna, Antti 2004. Integrointitestaus. [online] [viittaus 08.03.2005]. http://www.cs.helsinki.fi/u/taina/ohte/s-2004/luennot/integrointitestaus.pdf

Liite 1.

Problems with TestDirector?

Problem grades:

0 = Never used the feature

1 = Problems appear all the time when used the feature

2 = Problems appear sometimes when used the feature

3 = There is any problems with the feature

4 = Feature can be used almost without problems

5 = Feature can be used completely without problems

Action	Problems? (Using grades)	Comments (What kind of problems and what kind of correction should be made)
TestDirector Basics		
Getting Started		
Login (selecting Domain, project)		
Working with TD data		
Arranging columns		
Defining a filter		
Sorting records		
Refreshing and clearing settings		
Saving data to Text file		
Saving data to Excel sheet		
Saving data to Word document		
Adding Attachments		
Attaching a File		
Attaching a URL		
Attaching a Snapshot		
Attaching an image from the clipboard		
Viewing attachments		
Modifying an attachment		
Saving an attachment		
Deleting an attachment		
Requirements Specification		
Finding requirements		

Viewing the requirements tree		
Refreshing the tree		
Viewing associated defects		
Generating a test from requirements		
Test Planning		
Developing the Test Plan Tree		
Creating a test plan tree		
Adding a new test to a test plan tree		
Viewing tests in the test plan tree		
Associating defects with a test		
Finding a folder or test in the tree		
Sorting a test plan tree		
Renaming a test or a folder		
Deleting a folder		
Deleting a test		
Linking Tests To Requirements		
Adding requirements coverage to a test		
Deleting requirements from a test's requirements coverage		
Adding tests coverage to a requirement		
Deleting tests from a requirement's tests coverage		
Building Tests		
Creating a test step		
Editing test steps		
Reordering test steps		
Renumbering test steps		
Resizing test steps		
Arranging columns		
Deleting a test step		
Copying test steps		
Finding text in the test steps		
Finding and replacing text in the test steps		
Finding and replacing step text		
Test Execution		
Creating Test Sets		
Adding a new folder to test sets tree		
Adding a test set		
Adding tests to a test set using Test Plan Tree view		
Adding tests to a test set using Req Test Coverage view		

Removing tests from a test set		
Copying a test set		
Renaming a test set		
Deleting a folder from test sets tree		
Deleting a test set		
Resetting a test set		
Running Tests Manually		
Running a single manual test		
Running two or more manual tests		
Editing Run Details		
Cancelling manual test run		
Starting the test run (Exec Steps –button)		
Adding new steps while running the test		
Editing test steps while running the test		
Deleting test steps while running the test		
Adding defects while running the test		
Changing test status		
Ending the test run		
Resuming a manual test run		
Viewing Test Results		
Viewing test run properties		
Viewing details of a test run		
Viewing results of all runs for a test		
Viewing the history of changes to a test run		
Defect Tracking		
Adding a new defect		
Entering the relevant defect details		
Clearing the data in the Add Defect dialog box		
Finding similar defects		
Finding similar text		
Updating defects		
Finding and replacing values		
Viewing the history of changed to a defect		
Viewing an associated test		
Deleting a defect		
TestDirector Analysis		
Generating Reports		
Creating a report		

Customizing a report		
Using a report template		
Adding a sub-report		
Deleting a sub-report		
Generating Graphs		
Creating a graph		
Customizing the graph content		
Customizing the graph appearance		
Customizing the graph description		
Generating Project Documents		
Launching the Document Generator		
Setting formatting instructions		
Specifying Requirements module data		
Specifying Test Plan module data (Subject Tree)		
Specifying Test Grid data (Tests List)		
Specifying Test Lab module data (Execution)		
Generating a document		

If you have other problems with TD in your mind, tell those problems to me.