TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tutkintotyö

Timo Sacklén

OHJELMISTOTESTAUKSEN AUTOMATISOINTI

Työn ohjaaja Lehtori Erkki Hietalahti
Työn teettäjä Cybersoft Oy
Tampere 2009

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Timo Sacklén

Opinnäytetyö

Tekijä	Timo Sacklén
Työn nimi	Ohjelmistotestauksen automatisointi
Sivumäärä	30
Valmistumisaika	2009

Työn ohjaaja	Lehtori Erkki Hietalahti, Tampereen ammattikorkeakoulu
Työn tilaaja	Cybersoft Oy

TIIVISTELMÄ

Tehtävänäni oli perehtyä Cybersoftin ylläpitämän Elnet-sovelluksen ohjelmistotestauksen automatisointimahdollisuuksiin. Elnet on sähköverkon hallinta- ja ylläpitotyökalu, jota Fingrid käyttää. Työssä perehdytään vain Elnetin toiminnalliseen puoleen eikä käsitellä tietokanta-puolta. Elnetin työkalu on useita vuosia vanha Oracle Forms 6i, joka asettaa haasteita sopivien työkalujen kartoittamiseen. Työn alkaessa ei ollut tiedossa ainuttakaan työkalua tätä varten.

Työssäni käsitellään IBM:n Rational Robot-sovellusta, jolla testausta voidaan automatisoida Oracle Forms 6i:llä luoduissa sovelluksissa. Selvitykseni avulla voi saada hyvän kuvan siitä, mitä IBM Rational Robotilla voidaan tehdä ja miten sekä mitä rajoituksia tai ongelmia sovelluksen käytössä on.

Työn lopussa kuvaillaan IBM Rational Robotin käyttömahdollisuuksia ja järkevyyttä Elnetin testauksen automatisoinnissa. Sovellusta ei havaittu tarpeeksi hyödylliseksi Cybersoftin käyttöön.

Avainsanat Ohjelmistotekniikka, automatisointi, testaus

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Timo Sacklén

Opinnäytetyö

Writer	Timo Sacklén
Thesis	Automatisation of software testing
Number of pages	30
Graduation time	2009

Thesis Supervisor	Lecturer Erkki Hietalahti, Tampere Polytechnic
Co-operating Instructor	Master of Science Jani Malinen, Cybersoft Oy

ABSTRACT

My assignment was to find out different ways for automatising some parts of El-net's software testing. Elnet is a tool for maintaining and controlling the electric network and it is used by Fingrid. This work concentrates on the functional side of Elnet and not on the database side. The tool for creating and maintaining Elnet is a very old program called Oracle Forms 6i. The age of this software development tool will cause challenges in finding useful software automatisation tools. When I started this work, there was no known tools that could be helpful.

This work concentrates on IBM's Rational Robot software. Software created on Oracle Forms 6i can be tested with this tool. With the help of my research a reader can get a good image of the advantages and disadvantages that this tool has.

The possibilities of using IBM Rational Robot with Elnet are discussed in the end of this work. Robot was not seen useful enough for Cybersoft.

Keyword Software engineering, automatisation, testing

SISÄLLYSLUETTELO

JOHDANTO

Testaus on yksi ohjelmistoprojektin tärkeimmistä vaiheista. Kasvavien laatuvaatimusten vuoksi sovelluksen laatuun ja virheettömyyteen tulee erityisesti kiinnittää huomiota. Testaus suoritetaan Cybersoft Oy:ssä tätä kirjoittaessa täysin manuaalisesti. Työ on aikaa vievää, mutta laadun takaamiseksi testauksesta ei voida tinkiä. Asiakkaiden laatuvaatimukset kiristyvät koko ajan, ja virheet voivat muodostua kalliiksi.

Työn tavoitteena on selvittää mahdollisia työkaluja Cybersoft Oy:n ohjelmistotestauksen automatisoimiseksi, sekä mitä hyötyjä näillä työkaluilla voidaan saavuttaa. Työssä keskitytään Orarcle Forms 6i:llä kehitetyn Elnet-sovelluksen testaamistarpeisiin. Työtä aloitettaessa ei ole tietoa onko Oracle Forms 6i:lle työkaluja testaamisen automatisoimiseksi millään tasolla. Työkaluilla haluttaisiin testata Elnetin toiminnallisuutta automatisoidusti.

Fingrid käyttää Elnetiä Suomen sähköverkon hallintaan ja ylläpitoon. Elnet toimii Windows-käyttöjärjestelmissä. Kehitystyökaluna on Oraclen Forms 6i, jolla voidaan luoda graafisia käyttöliittymiä. Sovellus on julkaistu vuoden 2000 alussa ja se toimii Windows-käyttöjärjestelmissä. Formsin uusin versio on jo 11g, mutta Elnetiä kehitetään asiakkaan toiveesta 6i-versiolla. Formsin 11g-versiolla ei voida ylläpitää Forms 6i:llä luotuja ohjelmistoja.

1 CYBERSOFT OY

Cybersoft Oy on tamperelainen ohjelmisto- ja konsultointiyritys. Yritys on perustettu vuonna 1989 ja on toiminut yli 15 vuoden ajan sähkönsiirto- ja jakeluyritysten, kuten kantaverkkoyhtiö Fingridin kanssa. Muita merkittäviä asiakkaita ovat Tekla, Empower, Kone ja Kymppivoima.

Cybersoftin liikevaihto vuonna 2007 oli noin miljoona euroa, ja yritys tuotti voittoa 30 000 euroa. Cybersoft työllistää noin 15 henkilöä. Organisaatio koostuu kolmesta ryhmästä, joista kaksi ryhmää keskittyy nykyisten tuotteiden kehitykseen ja asiakkaisiin ja kolmas uusiin tuotteisiin. Yritystä johtaa toimitusjohtaja Jussi Rikala.

Cybersoftin tuotteisiin kuuluu RelayManager, jolla hallinnoidaan releitä, WindForecaster, jolla voidaan ennustaa tuulivoiman tuottoa, sekä C-CIM, jolla voidaan mallintaa sähköverkkoja. Näiden sovellusten lisäksi Cybersoft ylläpitää ja kehittää sähköverkon ylläpito- ja hallintasovellus Elnetiä Fingridille. Sovellusten kehittämiseen käytetään Oraclea ja Microsoftin .NETiä.

2 OHJELMISTOTESTAUKSEN TERMEJÄ

Seuraavassa käsitellään opinnäytetyössä käytettyjä keskeisiä termejä. Termit on selitetty siinä laajuudessa, että lukija ymmärtää tekstissä käytettyä sanastoa.

2.1 Skripti

IBM Rational Robotilla käyttäjä luo testiskriptin. Rational Robot luo SQA Basic-koodilla tiedoston, joka on kirjannut käyttäjän tekemät hiiren liikutukset ja näppäinten painallukset. Skripti on toistettavissa Rational Robotilla, jolloin Robot suorittaa skriptiin kirjatut hiiren- ja näppäimistön toiminnot automaattisesti. Skriptillä voidaan siis simuloida luotua testitapausta.

2.2 Lokitiedosto

IBM Rational Robot luo ajetuista skripteistä lokitiedoston. Lokista voidaan nähdä, tapahtuiko skriptiä toistettaessa virheitä, toistuivatko kaikki varmistuspisteet halutusti ja kauanko skriptin ajoon käytettiin aikaa. Lokeja voidaan hallita Test-Manager-sovelluksella.

2.3 LOV – Valintalista

Oraclessa käytetty ’List Of Values’ on omaan ikkunaansa aukeava vieritettävä lista, johon tiedot haetaan yleensä tietokannasta. LOV-ikkunassa on myös hakumahdollisuus, koska tietokannasta voi LOViin tulla satoja tietoja. LOVissa valittu rivi viedään (OK painikkeella) tietokantariville/riveille näyttöön, josta LOV avattiin. LOVin yleisin käyttötarkoitus onkin viedä tietokantakenttään käyttäjän valintalistasta valitsema arvo.

3 OHJELMISTOTESTAUKSEN TEORIAA LYHYESTI

Ohjelmistotestaus voidaan jakaa karkeasti kolmeen vaiheeseen: moduuli-, integraatio- ja järjestelmätestaukseen.

3.1 Moduulitestaus

Moduulitestauksessa testataan yhtä yksittäistä moduulia. Cybersoftin Elnet-kehityksessä tämä tarkoittaa usein yhden näytön Oracle Forms 6i- kooditiedostoa tai tietokantapakettia. Testauksen voi suorittaa testaaja itse tai toteutuksen auditoija, joka viime kädessä hyväksyy toteutuksen. Kun moduuli on läpäissyt moduulitestauksen, se voidaan liittää integraatiotestaukseen.

3.2 Integraatiotestaus

Integraatiotestauksessa on yhdistettynä moduulit yhdeksi kokonaisuudeksi tai pieniksi ryhmiksi. Näin voidaan testata eri osien toiminnallisuutta yhdessä. Esimerkiksi sovelluksen päivityksien yhteydessä suurin osa koodista on vanhaa ja vain osa muutettua. Integraatiotestauksella halutaan varmistua siitä, että uudet toiminnot toimivat vanhojen kanssa ongelmitta sekä toisin päin.

3.3 Järjestelmätestaus

Järjestelmätestauksessa testataan koko järjestelmän toimintaa. Järjestelmätestaus voidaan jakaa toiminnalliseen ja ei-toiminnalliseen testaukseen. Ensin mainitulla varmistetaan, että sovellus toimii määrittelyjen mukaisesti. Toiseksi mainitulla voidaan tarkastella esimerkiksi suorituskykyä. Pienikin muutos esimerkiksi tietokantakyselyyn voi aiheuttaa todella dramaattisia muutoksia kyselyn kestoon.

4 OHJELMISTOTESTAUS CYBERSOFTISSA

Tässä luvussa keskitytään Cybersoftin ylläpitämän Elnet-sovelluksen ohjelmistotestaukseen.

Ohjelmistotestaus Cybersoftissa toteutetaan toimintamallin mukaisesti, joka sisältää luvussa 4 mainitut testausvaiheet. Elnet-sovellus toimitetaan asiakkaalle pääsääntöisesti kaksi kertaa vuodessa kahtena päätoimituksena. Päätoimitukset sisältävät esimerkiksi bugikorjauksia, uusia näyttöjä, uusia raportteja tai uusia toiminnallisuuksia vanhoissa näytöissä tai raporteissa. Päätoimitusten lisäksi erittäin kiireellisiä korjauksia tai muutoksia voidaan toteuttaa muuttamalla asiakkaan tuotannossa olevan sovelluksen tiedostoja ilman, että asiakkaan järjestelmää tarvitsisi ajaa alas.

Asiakkaalle toimitettava ohjelmisto muodostuu yksittäisistä hankkeista ja kehitystöistä. Hanke on suurempi kokonaisuus, joka sisältää esimerkiksi täysin uuden näytön, raportin tai toiminnallisuuden rakentamisen. Kehitystyö on pienimuotoisempi toiminnallinen muutos jo olemassa olevaan koodiin tai virhekorjaus. Toimitukseen kerätään siis koko sovelluksen materiaali, joka sisältää sekä uudet muutokset ja lisäykset että vanhan muuttumattoman materiaalin.

4.1 Moduulitestaus

Hankkeiden ja kehitystöiden moduulitestausprosessi on samanlainen. Molemmissa tapauksissa toteuttajaksi nimetään ohjelmoija(t), moduulitestaaja ja auditoija. Usein moduulitestaajana toimii auditoija. Ohjelmoijan saatua työnsä valmiiksi moduulitestaaja aloittaa testauksen.

Testauksessa varmennetaan, että toiminnallisuus vastaa määrittelyä sekä yritetään selvittää, löytyykö toteutuksesta muita ongelmakohtia, kuten hitautta tai sovelluksen odottamatonta käyttäytymistä. Mikäli moduulitestaaja löytää virheitä, ohjelmoija korjaa virheet ja moduulitestaaja suorittaa testauksen uudelleen. Sovelluksen moduulitestaus on läpäisty, kun mainittavia virheitä ei enää löydy. Auditoinnissa,

eli katselmuksessa, tutkitaan vielä toiminnallisuus ohjelmoijien, moduulitestaajan ja auditoijan kesken.

4.2 Integraatiotestaus

Integraatiotestaus toteutetaan, kun hankkeet ja kehitystyöt ovat valmistuneet. Elnet-sovellus on jaettu osiin toiminnallisuuden mukaan, esimerkiksi vikapaikkalaskentaan, kunnossapitoon jne. Testaajille jaetaan sovelluksen osia testattavaksi ja testaus suoritetaan käyttötapauslistan mukaisesti. Elnetissä on noin 350 näyttöä ja 140 rapottia. Näille näytöille ja raporteille on muodostettu käyttötapausluettelo. Toimituksen yhteydessä osa käyttötapauksista voidaan poistaa käytöstä tai lisätä uusia käyttötapauksia. Tätä kirjoitettaessa käytössä olevia käyttötapauksia on noin 550.

Integraatiotestauksessa muodostetaan uusi tietokanta uusilla binääreillä. Testaus suoritetaan yleensä 1-2 työpäivän aikana. Koodeihin, joihin on tehty muutoksia kiinnitetään erityisesti huomiota. Näissä testaajana on myös itse hankkeen tai kehitystyön toteuttanut henkilö integraatiotestaajan lisäksi. Löydetyt virheet kirjataan ja niille nimetään korjaajat. Korjauksien jälkeen voidaan tarvittaessa suorittaa toinen integraatiotestauskierros.

4.3 Asiakastestaus

Koodien läpäistyä integraatiotestauksen koodit välitetään asiakkaalle, ja sovellusvastaavat suorittavat asiakastestauksen. Sovellusvastaavat raportoivat mahdollisista virheistä, jolloin niille nimitetään korjaaja ja normaali testausrutiini suoritetaan. Asiakastestauksessa löytyneiden bugien korjaamisen jälkeen sovellus toimitetaan asiakkaalle tuotantoon.

5 IBM RATIONAL ROBOT

Rational Robot on IBM:n kehittämä sovellusten testausohjelma, jolla voidaan automatisoida sovellusten *funktionaalista* testausta. Funktionaalisen testauksen tarkoituksena on todeta, että ohjelmiston toiminnallisuudet toimivat, kuten niiden tuleekin toimia. Funktionaalista testausta käytetään esimerkiksi valmiiseen ohjelmistoon, johon luodaan uusia ominaisuuksia. Uuden version myötä tulee varmistua, että uusien toiminnallisuuksien jälkeen vanhatkin toiminnallisuudet toimivat oikein. Robot ei tutki testien aikana ohjelmakoodia.

Robotilla voidaan testata sovelluksia, joiden ohjelmointikieli on html, DHTML, Java, VS.NET, Microsoft Visual Basic ja Visual C++, Oracle Developer/2000, PeopleSoft, Sybase PowerBuilder tai Borland Delphi. Rational Robot generoi testiskriptinsä SQABasic-kielellä. Sovellusta voidaan käyttää seuraavissa käyttöjärjestelmissä: Windows XP, ME, 2000, 2003, 98 ja NT. /2/

5.1 Asennus

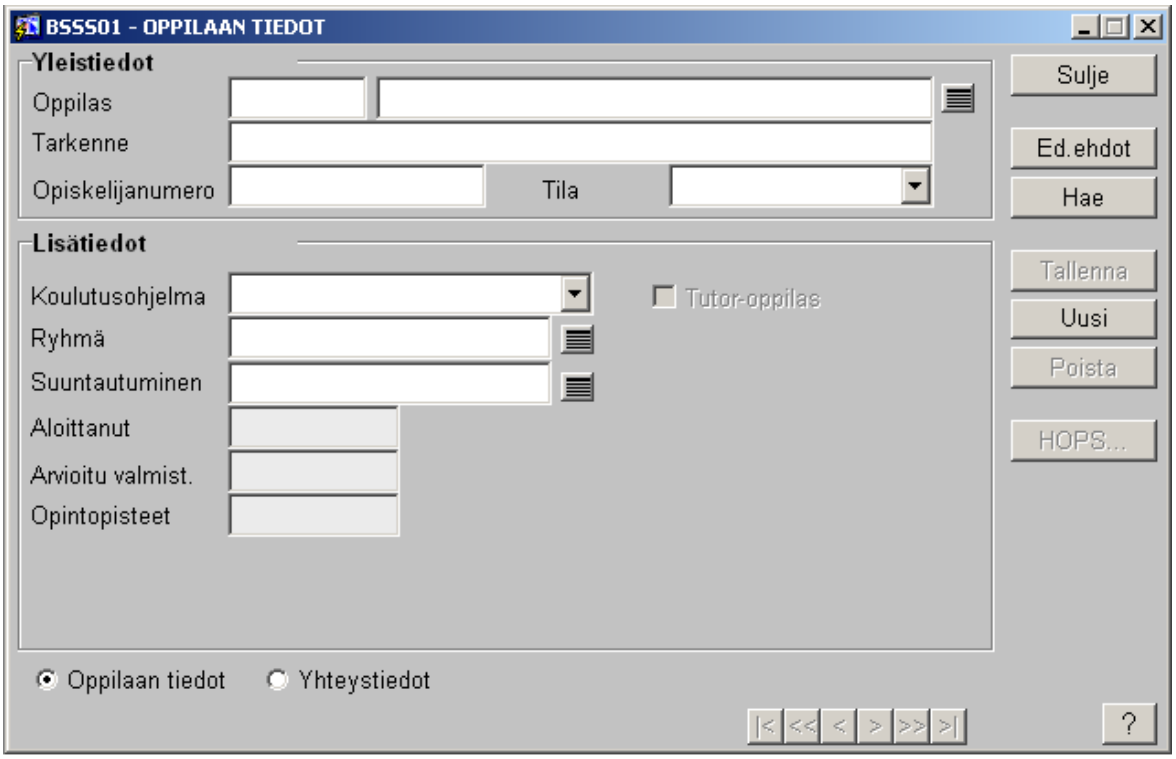
Rational Robotin ilmaisversion hankkimiseksi on tehtävä tunnukset IBM:n sivuilla (http://www.ibm.com/us/en/). Syöttämällä hakuun'rational robot' voidaan siirtyä halutulle sivulle. Robotin lataamiseksi koneelle tulee ladata Robotin 15 päivän avain ja itse Rational Robot V7.0.1. Asentaminen on samanlainen kuin missä tahansa normaalissa Windows-sovelluksessa. Kun Rational Robot on asennettu, aktivoidaan ladattu 15 päivän avain IBM Rational License Key Administratorilla. Mikäli sovellus on ostettu yritykseen, ladataan vain asennuspaketti tai asennetaan cd:ltä sekä asetetaan License Key Administratoriin ostettu avain.

5.2 Testiskriptin suunnittelu

Testitapausten suunnittelu suurelle ohjelmistolle sisältää useita haasteita. Testattavissa ohjelmissa on useita näyttöjä, jotka sisältävät erilaisia toimintoja. Testaussuunnitelmaa tehdessä tulee miettiä, miten eri näyttöjen testaus suoritetaan mahdollisimman tarkasti, mutta samalla luomatta turhan paljon testiskriptejä. Yhden näytön toiminnallisuuksia ei siis kannata jakaa useaan eri skriptiin, koska tästä seuraa lisää työtä henkilölle, joka skriptejä ajaa. Esimerkiksi näyttö, joka sisältää toiminnot luo uusi, hae ja poista, voidaan testata kattavasti yhdellä skriptillä, jolla voidaan

todeta ohjelman toiminta. Skriptien määrä pysyy alhaisempana, kun jokaiselle yllä mainituista toiminnoista ei luoda omaa skriptiä varmennuspisteineen.

Elnet-sovelluksessa on useita käyttäjätunnuksia, joilla on eriasteiset käyttöoikeudet. Tästä johtuen esimerkiksi uusien tietojen luonti- ja poisto-oikeus on rajattu valituille käyttäjätunnuksille (kuva 1). Luodut skriptit, joilla tehdään käyttöoikeuksilla rajattuja toimintoja, voidaan siis ajaa onnistuneesti vain tietyillä tunnuksilla. Mikäli skripti ajetaan läpi ilman vaadittuja käyttöoikeuksia, käyttäjä saa ajon jälkeen lokin, jossa on useita virheitä. Virheet syntyvät esimerkiksi tiettyjen painikkeiden puuttumisesta Elnetin näytöltä. Painikkeet kuten ”Luo uusi” näkyy vain käyttäjätunnuksella, jolla on oikeudet luoda uusia tietoja. Skriptien ajo tunnuksilla, joilla ei ole näytön kaikkiin ominaisuuksiin oikeuksia, on kumminkin tarpeetonta. Oikeudet toimintojen käyttöön pitää antaa erikseen, eivätkä oikeudet muutu tai mene sekaisin esimerkiksi uuden sovelluksen käännön yhteydessä. Kaikilla käyttäjätunnuksilla toimivat ominaisuudet, kuten tietojen haku tulisi siis testata samalla tunnuksella, jolla on oikeudet tietojen muuttoon.



Kuva 1: Esimerkkiskriptissä käytetty Elnet-näyttö, jossa on kaikki oikeudet

Näyttöjen keskeisimpien toimintojen toiminnallisuuden lisäksi tulee skriptiin sisällyttää myös kaikkien perustoimintojen testaus, kuten oikean LOVin ja helpin au-

keaminen. Esimerkiksi selauspainikkeiden oikeaa toiminnallisuutta ei voida varmentaa, koska selattavat tiedot haetaan tietokannasta. Tiedot voivat muuttua, jolloin esimerkiksi kuvan 1 Asema-kentän tietoa tarkastettaessa varmistuspisteellä ei voida pitää järkevänä.

Testitapausten suunnittelusta on suotavaa luoda ohje. Ohje sisältäisi kaikki perustestaukset, jotka skripteihin tulee sisällyttää. Ohjeen avulla kuka tahansa voi äkkiä perehtyä, mitä tulee ottaa huomioon skriptiä luodessa. Ohjeen tulisi olla samankaltainen kuin esimerkiksi Cybersoftin moduulitestausohje.

Funktionaalisia testejä luodessa tulee keskittyä toteamaan keskeisimpien ominaisuuksien toiminta. Esimerkiksi tietokantakenttään ”Etunimi” syötettäessä vaikkapa ”Testi” tavoitteena on varmistua siitä, että syötetty etunimi viedään onnistuneesti tietokantaan. Syötettyä tietoa haettaessa halutaan varmistuspisteellä varmistua, että syötetty kirjainjono haetaan onnistuneesti tietokannasta. Kyseisen esimerkin tarkoituksena ei siis ole havaita mahdollisia virheitä. Esimerkiksi voidaanko kenttään syöttää nimi, joka sisältää muita merkkejä kuin aakkosia (esimerkiksi ”T3st!”) ja seuraako tästä virheilmoituksia. Nämä virheet tulee löytää jo sovelluksen luontivaiheessa manuaalisesti testaten eivätkä siis kuulu funktionaalisen testauksen piiriin.

5.3 Testiskriptin luonti

Uuden testiskriptin luonti voidaan suorittaa kahdella tapaa. Ensimmäinen tapa on painaa Robotin työkalupalkista (Kuva 9) punaista palloa, jonka alla lukee GUI. Tämän jälkeen Robot pyytää skriptille nimen. Kun nimi on annettu, tulee esille skriptin nauhoituspalkki (Kuva 2) ja käyttäjän näppäinpainallukset ja klikkaukset talletetaan skriptiin. Nauhoitus voidaan keskeyttää millä hetkellä tahansa painamalla pause-painiketta nauhoituspalkista. Toinen tapa luoda uusi testiskripti on valita File – New script. Ohjelma pyytää käyttäjältä nimen skriptille. Tämän jälkeen eteen avautuu testikoodin runko Robotissa. Nauhoittaminen voidaan aloittaa painamalla punaista palloa, jonka alla lukee GUI. Listasta valitaan äsken luotu skripti ja aloitetaan nauhoitus. Ensimmäistä tapaa tulee suosia, koska se on nopeampi eikä itse tyhjään skriptin runkoon tarvitse tehdä manuaalisesti muutoksia ennen nauhoituksen aloittamista. Esimerkiksi skriptiä kuvaavat kommentit kannattaa lisätä vasta

nauhoituksen jälkeen skriptin alkuun. Kommenteista tulisi käydä selvästi ilmi, mitä skripti tekee ja millä näytöllä.

Kuva 2: Robotin nauhoituspalkki

Testitapaus voidaan luoda myös TestManagerin puolelta painamalla samaa punaista palloa, jonka alla lukee GUI. Kun painiketta on painettu, Rational Robot käynnistyy. Tämän jälkeen voidaan edetä, kuten yllä on selostettu.

Robotin aloitettua nauhoituksen tulee käyttäjän olla huolellinen. Kaikki tarpeettomat tai väärätkin klikkaukset rekisteröidään skriptiin, kuten myös jokainen näppäimistön painalluskin. Mikäli näitä virheitä syntyy skriptin nauhoituksen aikana, voidaan ne karsia nauhoituksen päätyttyä manuaalisesti. Poistettaessa skriptistä ylimääräisiä painalluksia tulee olla tarkka, että ei poista väärää klikkausta tai painallusta. Hyvä tapa todeta tämä on kommentoida ylimääräisiksi oletetut rivit skriptistä ja kokeilla, voidaanko skripti ajaa oikein. Mikäli skripti pystytään ajamaan läpi ongelmitta, voidaan kommentoidut rivit siivota pois testikoodista. Testikoodin luettavuuden ja siistinä pysymisen kannalta on tärkeää, että virhepainalluksia ei jäisi koodiin.

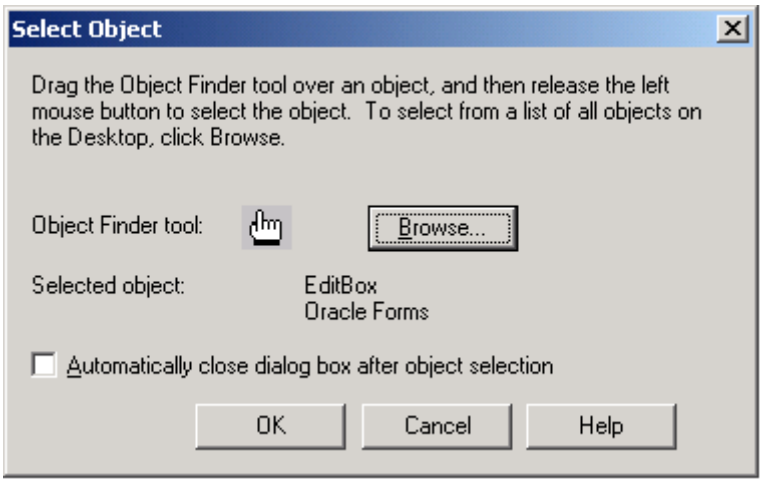
5.4 Varmistuspisteet

Varmistuspisteillä (Verification point) voidaan varmistua halutusta toiminnosta skriptin ajon aikana. Toiminnallisuuden testaamisessa varmistuspisteitä voidaan hyödyntää esimerkiksi LOVien, uusien syötettyjen testitietojen varmentamisessa ja ohjetiedostojen kanssa. Oracle Forms 6i voi uuden sovelluksen käännön yhteydessä aiheuttaa LOVien kanssa kummallisuuksia. Esimerkiksi LOV-painiketta painettaessa avautuu väärä LOV tai koko sovellus voi kaatua. Varmistuspistettä voidaan käyttää hyväksi, kun skriptissä klikataan auki haluttu LOV ja varmistuspisteellä tarkistetaan LOV-ikkunan otsake. Tällä tavalla voidaan automaattisesti varmistua oikean LOVin avautumisesta.

Myös itse LOVien sisältö voidaan tarkistaa, mutta koska kyseessä on tietokannasta haettavia arvoja, niin tämä ei välttämättä ole viisasta. Mikäli tietokannasta haettu lista on muuttunut skriptin tekohetkestä, ei skripti enää mene läpi onnistuneesti. Esimerkiksi kun tietokantaan on lisätty uusi asematyyppi, ei skriptin asematyyppi-listauksen varmistus enää läpäise testiä. Formsin ohjetiedostot ovat tyypillisiä Windowsin .hlp-muotoisia ohjetiedostoja, kuten useissa muissakin ohjelmistoissa. Näistä Rational Robotilla voidaan tarkistaa vain ikkunan otsake, joka on Elnetissä aina näyttökohtainen. Näin voidaan varmistua oikean helppitiedoston avautumises-ta oikealla näytöllä.

Kuva 3: Verification pointin lisääminen

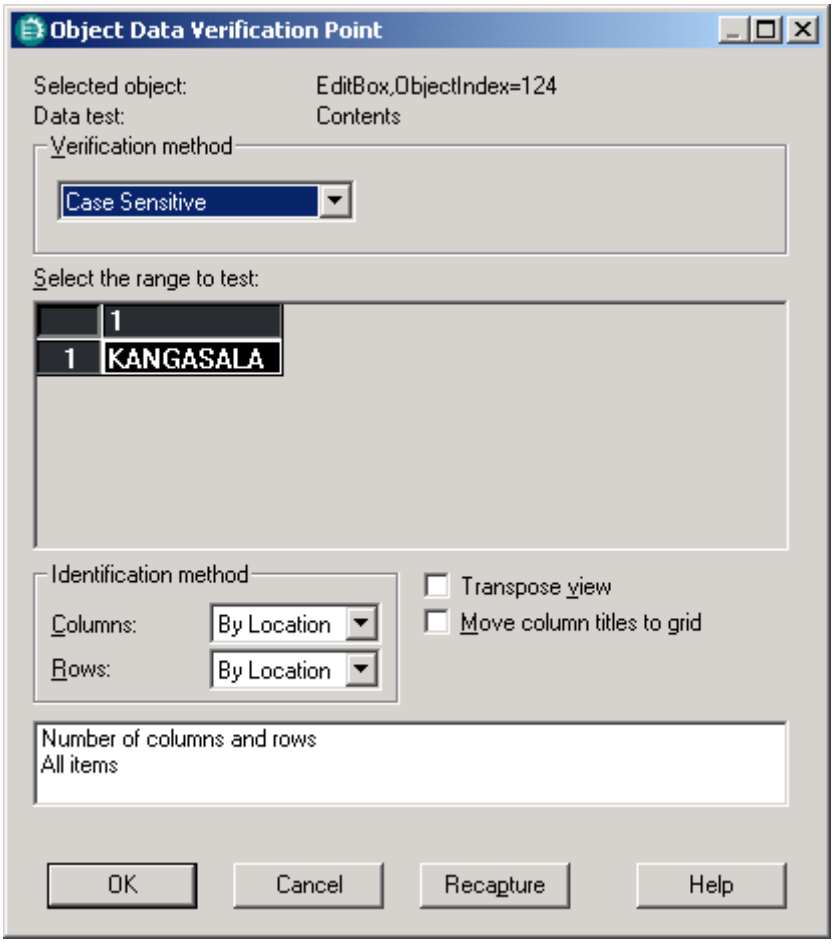
Varmistuspiste voidaan lisätä skriptiin sitä tehtäessä valitsemalla nauhoituspalkis-ta(Kuva 1) ”Display GUI Insert toolbar” -näyttö, jolloin avautuu kuvan 3 mukainen palkki. Oikean puolimmaista ikonia painettaessa voidaan asettaa ”Object Data” -tyyppinen varmistuspiste.



Kuva 4: Robot varmistuspisteen valitseminen

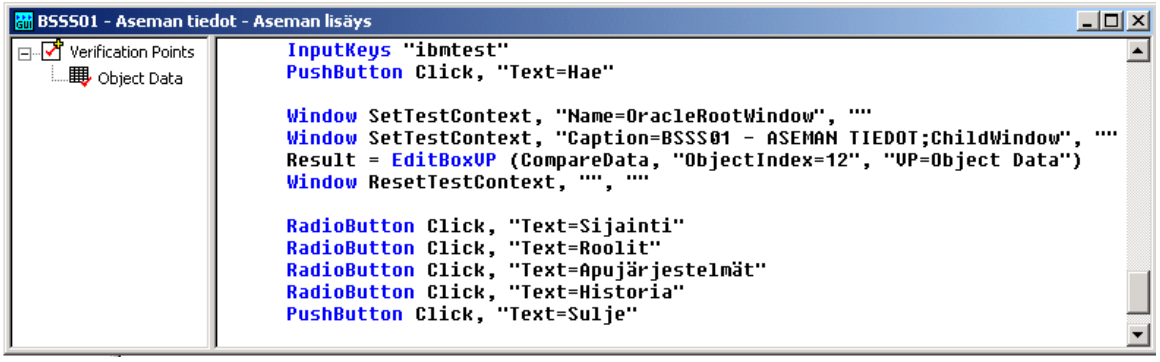
Käyttäjän valittua ”Object Data” -varmistuspisteen lisäämisen Robot avaa kuvassa 4 olevan ikkunan. Käyttäjä klikkaa kuvassa näkyvää sormea ja vetää sen haluaman-

sa objektin päälle. Kuvassa sormi on viety editboxiin, eli muokattavan tekstikentän (tietokantakentän) päälle. Kun haluttu objekti on valittu ja Robot on onnistuneesti tunnistanut objektintyypin kuten kuvassa, voidaan antaa ehdot varmistuspisteelle. Mikäli ”Selected object” -kenttä jää tyhjäksi, Forms ei ole tunnistanut käyttäjän tarkoittamaa Forms-objektia eikä varmistuspistettä voida lisätä.



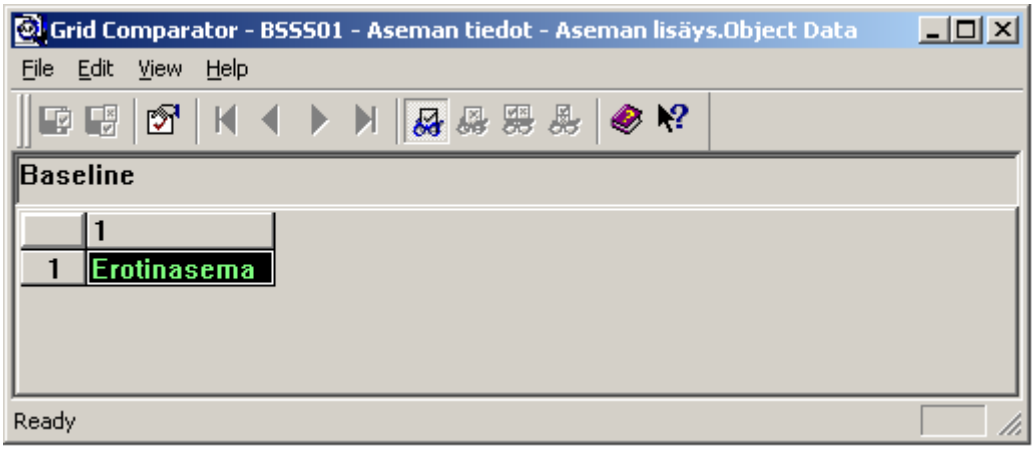
Kuva 5: Object Data tyyppisen varmistuspisteen asetukset

Kun haluttu objekti on valittu ja Robot tunnistanut objektin tyypin kuten yllä, voidaan antaa ehdot varmistuspisteelle. Kuvan esimerkissä tietokantakentässä pitäisi olla teksti ”KANGASALA” (Kuva 5). Olennaisin asetus tekstikenttää tarkistettaessa on, että pitääkö varmistettavan tekstin olla täsmälleen oikein (case sensitive), kuten kuvassa 5.



Kuva 6: Varmistuspiste skriptissä

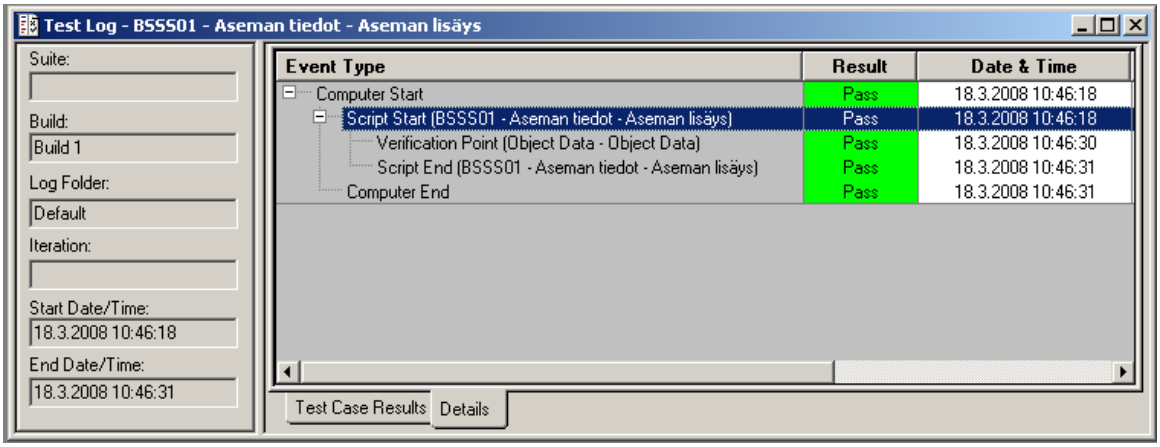
Skriptissä olevat varmistuspisteet voidaan katsoa Robotin skripti-ikkunan vasemman reunan puurakenteesta. Verification Points puurakenteen sisällä on nähtävillä skriptin varmistuspisteet kuvan 6 mukaisesti. Skriptissä voidaan kuvan keskiosassa nähdä ”VP=Object Data”, joka tarkoittaa, että VP (verification point) on nimeltään Object data. Kuvan vasemman reunan Object data -varmistuspistettä klikkaamalla avautuu kuvan 7 mukainen kuva. Kuvan skriptissä on vain yksi varmistuspiste. Mikäli varmistuspisteitä on useampia, ne ovat puurakenteessa omilla riveillään, ja varmistuspistettä klikkaamalla käyttäjä voi nähdä, mitä varmistuspiste tekee.



Kuva 7: Object Data varmistuspiste(et) skriptissä

Kuvassa 7 on kuvan ikkunan otsakkeen nimisen skriptin varmistuspiste. Varmistuspisteen tyyppi on Object Data. Varmistuspisteellä tarkistetaan Elnetin tietokantakentässä olevaa informaatiota. Mikäli skriptissä varmistettaisiin tietokantataulukon kenttien sisältöä, näkyisivät ne taulukkomaisesti kuvassa.

Kuvassa 8 (loki) on kuvattu erään skriptin loki, joka avautuu kun skripti on ajettu läpi. Skriptissä on yksi varmistuspiste, ja varmistuspisteen tulos on lokissa omalla rivillään. Varmistuspiste on onnistunut, joten tuloksena on ’Pass’ vihreällä tekstillä. Mikäli varmistuspisteen tieto ei vastaa haluttua, tekstinä on ’Fail’ punaisella. Kuvan tapauksessa on luotu skripti, jossa luodaan uusi tietue, tallennetaan, haetaan luotu tieto ja varmistetaan yhden tietokantakentän hakutulos. Varmistuspistettä luotaessa ehdoksi on asetettu, että tuloksen pitää olla tosi. Haun jälkeen kyseisessä kentässä on oltava teksti ”EROTINASEMA”. Mikäli kentässä on jokin muu teksti (tai ei lainkaan tekstiä) tai teksti ei ole merkkikokoriippuvainen, nähdään varmistuksen epäonnistuminen lokista skriptin ajon jälkeen. Mikäli skriptissä olisi useampia varmistuspisteitä, ne näkyisivät lokissa omilla riveillään ja jokainen saa itsenäisesti muista varmistuspisteistä riippumatta arvon pass/fail. Esimerkiksi alla olevan kuvan lokissa klikattaessa Verification Point riviä, aukeaa kuvan 7 mukainen näkymä. Näkymä on hyödyllinen tarkastettaessa, mikä teksti tai arvo ei täsmännyt haluttua.

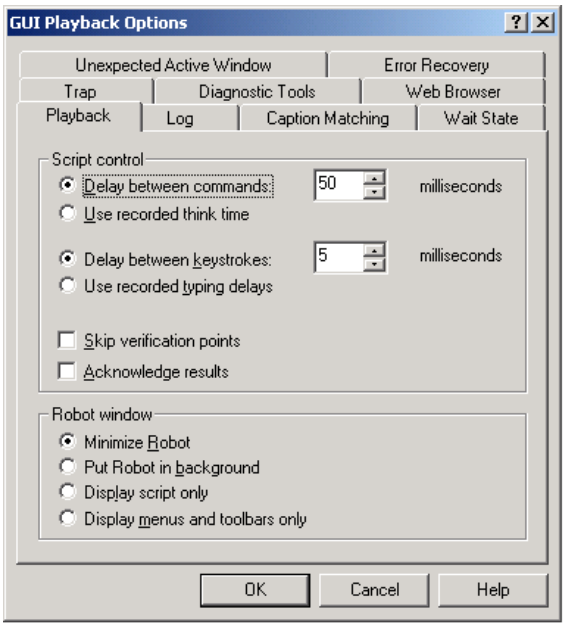


Kuva 8: Skriptin loki – varmistuspiste onnistunut

5.5 Testitapauksen ajaminen

Testitapauksen ajamista varten tulee ensin käynnistää Rational Robot. Käyttäjä voi halutessaan ensin tarkastella skriptiä ja sen kuvausta avaamalla skriptin javalitsemalla File valikosta Open -> Script. Listasta valitaan testiskripti, jota halutaan tarkastella ja painetaan OK. Haluttu testitapaus avautuu käyttäjälle, ja käyttäjä avaa Elnetin ja kirjautuu sisään tunnuksilla, joilla testitapaus ajetaan. Testitapauksen ajoon tarvittava käyttäjätunnus voidaan kommentoida vaikkapa skriptin alkuun, jotta käyttäjä tietää, millä tunnuksilla on kirjauduttava Elnetiin.

Ennen testitapauksen ajoa käyttäjä voi muokata ajoasetukset haluamikseen Tools-valikon ”GUI Playback Options” -ruudulta (Kuva 9).



Kuva 9: GUI Playback Options Rational Robotissa.

Olennaisimmat asetukset ovat Playback -välilehden ”Script control” aika-asetukset. Näillä asetuksilla voidaan asettaa, mikä on Robotin viive ennen skriptin seuraavan rivin suorittamista ja mikä on näppäinsyötteissä Robotin kirjoitusnopeus merkkien välillä. Ensin mainittua viivettä kasvattamalla voidaan skriptistä tehdä hyvin seurattavissa oleva. Käyttäjä voi itse nähdä selvästi skriptin eri suoritusvaiheet halutessaan. Käytännössä viiveitä kannattaa muuttaa suuremmiksi vain, kun skripti luo-

daan. Tällöin voidaan varmistua silmämääräisesti, että kaikki olennainen on varmasti sisällytetty skriptiin.

Uuden buildin jälkeen testaaja voi ajaa skriptit läpi esimerkiksi kuvan 9 aikaarvoilla. Robot voi vaikuttaa hyvinkin hitaalta jos skriptin rivillä suoritetaan tietokantahaku, joka kestää esimerkiksi kymmeniä sekunteja. Tällöin seuraavalle riville siirtyminen kestää hakuajan sekä annetun viiveen verran. Asetuksista voidaan myös halutessa ohittaa varmistuspisteet.

Kun käyttäjä haluaa avata skriptin tarkasteltavaksi ja kun ajoasetukset ovat asetettu halutuiksi, skripti voidaan käynnistää. Ajaminen suoritetaan Robotin työkalurivin (Kuva 10) Play painikkeesta. Kun Play-painiketta on painettu, Robot kysyy mikä skripti ajetaan, eikä siis vakiona suorita esimerkiksi Robotissa auki olevaa skriptiä. Skriptejä ei myöskään voida ajaa ajoitetusti. Käyttäjä valitsee listasta haluamansa skriptin ja mikäli skripti on jo ennen ajettu, Robot kysyy kirjoitetaanko lokitiedosto ajon jälkeen yli, sillä lokitiedostot ovat vakiona samassa kansiossa. Robotissa voidaan määritellä lokikansio, kun on valittu ajettava skripti. Lokit olisi kätevä luoda Elnetin version (build) mukaisesti omiin kansioihin. Tällöin ei tarvitse kirjoittaa yli vanhoja lokeja eikä ohjelma luonnollisesti kysy, kirjoitetaanko vanhan lokin päälle.

Kuva 10: Robotin työkalurivi – Skriptin käynnistys

Skriptin ajon aikana käyttäjä ei voi käyttää konetta, joten kädet on pidettävä irti näppäimistöltä ja hiireltä. Robot liikuttaa skriptin mukaisesti hiirtä ja antaa näppäinpainalluksia.

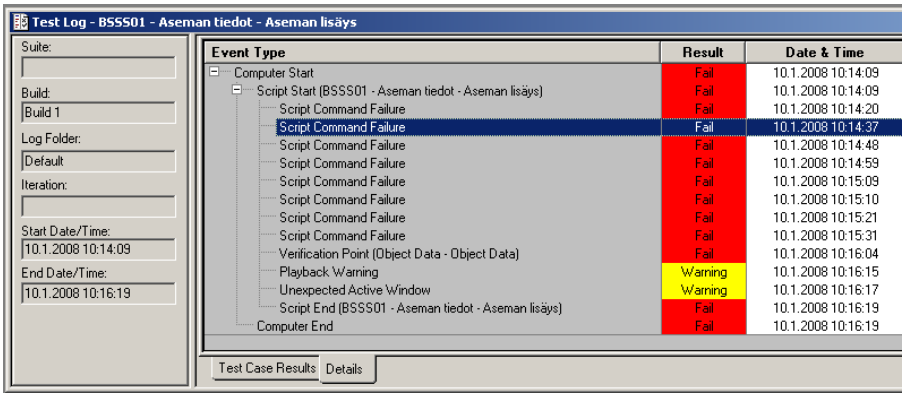
Mikäli ajovaiheessa tulee odottamattomia virheitä, skriptin ajo on hitaampi. Robot osaa itse ohittaa ongelmatilanteet, mutta ajoaika pitenee. Mitä enemmän virheitä

esiintyy, sitä pitempi ajoaika onnistuneeseen ajoon verrattuna. Ajon aikaisiin virheisiin voivat helposti johtaa esimerkiksi testaajan huolimattomuusvirheet.

Mikäli testaaja kirjautuu Elnetiin tunnuksilla, joilla ei ole oikeuksia skriptin suorituksessa vaadittaviin toimintoihin, seuraa virheitä ja pitempi ajoaika Robotin selviytyessä odottamattomien virheiden ohitse. Ongelman ohittamiseen käytettävä maksimiaika on asetettavissa Robotin aika-asetuksista.

5.6 Testitapauksen ajotulosten tarkastelu

Robot avaa käyttäjälle skriptin lokin Rational TestManageriin skriptin ajon tultua loppuunsa. Kuvassa 8 on esitettynä loki skriptin ajosta, jossa ei ollut virheitä. Virheettömyys voidaan helpoiten havaita, kun kaikki testin kohdat ovat lokissa vihreällä. Kuvassa 11 on yritetty suorittaa samaa skriptiä käyttäjätunnuksella, jolla ei ole oikeuksia luoda uutta tietuetta. Käyttäjälle ei siis myöskään ole näkyvissä näytöllä painikkeita kuten ’Uusi’, ’Poista’ tai ’Talleta’. Skriptin yrittäessä painaa painiketta, jota ei ole, syntyy virherivi ”Script command failure”, ja Result -sarakkeessa on virheilmoitus ”Fail”. Varmistuspiste epäonnistui, koska tietuetta ei luotu ja sitä hakemalla ei löytynyt tuloksia, joista voisi varmistusta suorittaa. Mikäli skriptissä on kohtia, joita ei onnistuttu suorittamaan, tekstin väri on punainen. Keltaisella oleva ”Warning” ei tarkoita, että skriptin ajossa tapahtui virheitä. Tällä ilmoitetaan käyttäjälle, että ajon aikana tapahtui jotain odottamatonta, joka ei kuitenkaan vaikuttanut skriptin suoritukseen. Kuvan 11 lokissa käyttäjä on kesken skriptin aktivoinut hiirellä jonkin toisen sovelluksen ja välittömästi aktivoinut Elnetin takaisin.



Kuva 11: Loki skriptin ajosta

Lokin vasemmasta reunasta käyttäjä voi hahmottaa skriptin ajoajan katsomalla aloitus- ja lopetusaikaa. Onnistuneessa tapauksessa kuvassa 8 skriptin ajoon kului 13 sekuntia, mutta yllä olevassa epäonnistuneessa ajossa lähes 2 minuuttia enemmän. Tämä johtuu siitä, että Rational Robot jää hetkeksi ihmettelemään, miksi esimerkiksi painiketta ei löydy. Ohjelma jatkaa pienen tauon jälkeen automaattisesti skriptin suorittamista.

Skriptin suorituskykyä voidaan valvoa esimerkiksi kommentoimalla luodun skriptin alkuun nimen, kuvauksen ja käytettävän käyttäjätunnuksen lisäksi sen ajoaika skriptin luontihetkellä. Ajettaessa skriptejä jatkossa testaaja voi skriptin avatessaan nähdä, mitä skripti tekee ja kauanko ajon pitäisi kestää.

Toinen tapa on pitää lokit tallessa. Vanhoja lokeja voidaan tarkastella TestManager-sovelluksessa ja lokista nähdään suoritusaika. Lokien tarkasteluun kuluu kumminkin aikaa, joten ajoajan kommentointi skriptin alkuun on tehokkaampaa. Vaikka skripti menisi läpi virheittä, ei ajoajan pitäisi olla selvästi edellistä ajokertaa pitempi. Suurin ajoaikaan vaikuttava tekijä on tietokannasta tehtävät kyselyt. Tietokannan suorituskykyyn voivat vaikuttaa useat tekijät, mutta suorituskyky ei saisi selvästi laskea.

5.7 Rational TestManager

Rational TestManager on IBM:n testaussovellusten hallintaohjelma. TestManager tulee mukana ilmaiseksi, kun ostaa jonkin IBM:n ohjelmistotestaussovelluksista, kuten Rational Robotin. TestManager luo lokit Rational Robotin ajamista skripteistä. Lokit tallennetaan kiintolevylle ja niitä voidaan avata TestManagerilla tarkastettavaksi. Sovelluksella voidaan myös luoda raportteja tai graafeja esimerkiksi suorituskyvystä, mutta Rational Robotin kanssa näistä ominaisuuksista ei ole juuri hyötyä. Lokien katselun ja hallinnoinnin lisäksi Robotin kannalta olennaisimmat TestManagerin toiminnot ovat menun pikalinkit Robotin skriptien nauhoitukseen.

5.8 Rational Robot -lisenssi

Mikäli yritys on kiinnostunut investoimaan Rational Robotiin, tulee valita kahden lisenssin välillä: yhden koneen tai useiden koneiden (floating license) lisenssin välillä. Ensin mainitulla voidaan Rational Robot asentaa vain yhdelle koneelle, ja sitä saa käyttöehtojen mukaan käyttää vain yksi nimetty henkilö. Henkilöä ei voida vaihtaa väliaikaisesti, vain pysyvästi. Lisenssin mukana tulee tuotetuki 12 kuukauden ajalle maksutta. Hinta tätä kirjoitettaessa on veroineen 6731,96 euroa.

Jälkimmäisessä vaihtoehdossa Robot voidaan asentaa monelle koneelle. Mikäli lisenssejä on ostettu vain yksi kappale, voi ohjelmaa yhtäaikaisesti käyttää vain yhdeltä koneelta. Toinen käyttäjä voi käyttää sovellusta omalta tietokoneeltaan vasta ensimmäisen lopetettua työskentelynsä Robotilla. Lisenssin hinta on kaksinkertainen yllä mainittuun verrattuna ja se oikeuttaa myös ilmaiseen tuotetukeen 12 kuukauden ajan. Lisälisenssien avulla voidaan yhtäaikaista käyttäjämäärää nostaa. Lisälisenssejä voi ostaa haluamansa määrän, jolloin samaan aikaan sovellusta voi käyttää lisenssin määrän verran työntekijöitä. Siispä mikäli Robotia halutaan käyttää useammalta kuin yhdeltä koneelta kerrallaan, voidaan ostaa lisälisenssejä hieman halvempaan hintaan. Lisenssi (floating license) voidaan ostaa myös vain 12 kuukaudeksi, jolloin hinta on puolet mainitusta ja sisältää myös ilmaisen tuotetuen. Tämä lisenssi voidaan uusia käyttöajan tullessa umpeen hintaan joka on 80 % alkuperäisestä ostohinnasta. Hinta on siis edullisempi kuin täyden lisenssin ostaminen, mikäli tuotetta käytetään 3 vuotta tai vähemmän. Joka tapauksessa sovelluksen lisenssejä voidaan pitää erittäin hintavina. Kyseenalaista on myös riittääkö, mikäli on vain yksi Robotilla testaava henkilö koko yrityksessä.

6 YHTEENVETO

Rational Robotilla voidaan automatisoida osa Cybersoftin ohjelmistotestaamisesta. Ohjelma ei sovellu uusien ominaisuuksien tai näyttöjen testaamiseen, koska tehdyt skriptit matkivat ihmisen tekemää testaamista. Uusille näytöille ja muuttuneille toiminnallisuuksille jouduttaisiin rakentamaan aina uusi testiskripti. Ihminen havaitsee uusien näyttöjen ja ominaisuuksien kanssa virheet paremmin ja osaa myös testata ominaisuuksia käyttämällä niitä toisin kuin on tarkoitettu. Robotia voidaan siis käyttää pääasiassa funktionaaliseen testaamiseen.

Robotin hyödyt tulevat esiin automatisoidessa uuden sovellusversion vanhojen ominaisuuksien testaamista. Kaikki vanhat ja muuttumattomat ominaisuudet on hyvä testata uuden version myötä ja jo luoduilla skripteillä voidaan varmistua siitä, että perusominaisuudet toimivat oikein. Oracle Forms 6i saattaa tehdä virheitä käännettäessä, joka voi aiheuttaa sovelluksen kaatuilua tietyillä toiminnoilla. Tärkeimpien toimintojen, eli kriittisten käyttötapausten, testaamisessa voidaan käyttää myös Robotia. Kriittisten käyttötapausten pitää ehdottomasti toimia, joten ei ole suotavaa laskea vain automatisoidun testaamisen varaan.

Uusien Elnet-versioiden testaaminen vie vuosittain arviolta noin xx työpäivän verran aikaa. Tätä työmäärää voitaisiin alentaa käyttämällä Robotia vanhojen ominaisuuksien testaamiseen. Uusissa Elnet-versioissa muuttuu (tai tulee lisää) vain pieni osa Elnetiä. Arviolta voidaan sanoa, että uuden version testaamiseen käytettyä työmäärää voitaisiin vähentää melko vähän. Työmäärä koostuisi tämän jälkeen kriittisten sekä uusien käyttötapausten manuaalisesta testaamisesta sekä Robotin käyttöön kuluvasta ajasta. Skriptien suunnittelu, luonti, testaaminen ja mahdolliset muutokset ovat myös aikaa vievää työtä, mutta voi maksaa itsensä takaisin ohjelmakehityksen jatkuessa pidempään nykyisellä käyttöalustalla.

Rational Robot on sovelluksena erittäin hintava. Elnetin käyttöalusta ei loputtomiin tule pysymään samana. Nykyinen Oracle Forms 6i on toiminut Elnetin käyttöalustana jo vuosikausia. Investointi Rational Robotiin saattaa siis olla kyseenalaista

kustannustehokkuuden ja saatavan hyödyn suhteen. Robot ei ole yhteensopiva esimerkiksi uudempien Oracle Forms-versioiden tai Microsoftin .NET:n kanssa, jota saatetaan tulevaisuudessa käyttää Elnet-kehityksessä. Elnetin korvaavan järjestelmän Elviksen kehitystyökaluja ei ole vielä valittu. Tulevan sovelluksen määrittely alkanee lähivuosina.

Rational Robotista ei siis ole tarvittavaa hyötyä Cybersoftin Elnet-testaamisessa.

LÄHTEET

1 Ohjelmistotestaus ja siinä käytettävät työkalut [www-sivu] [viitattu 17.4.2008] Saatavissa: http://www.mit.jyu.fi/opiskelu/seminaarit/ohjelmistotekniikka/testaus/

2 Rational Robot tuotesivu [www-sivu] [viitattu 17.4.2008] Saatavissa: http://www-128.ibm.com/developerworks/rational/products/robot/

LIITE 1

Testiskripti jossa luodaan uusi tietue, haetaan luotua tietuetta ja käytetään varmistuspistettä, jolla varmistetaan, että luotu tietue onnistuneesti löytyy, sekä lopuksi poistetaan luotu tietue. Koordinaatit viittaavat avatun Elnet-näytön kenttien sijanteihin, eikä tiettyä koordinaattia ruudulla. Näytön sijainti ruudulla ei siis muodostu ongelmaksi. Skriptiä ei kannata myöskään muuttaa käsin, vaan on suositeltavampaa suorittaa nauhoitus uudestaan.

Testiskripti – Uuden oppilaan luonti, haku ja varmistuspisteen käyttö, jotta haulla voidaan varmistua tiedon lisääntyneen oikein.

```
Sub Main
    Dim Result As Integer

    'Initially Recorded: 30.1.2008  14:35:03
    'Script Name: Oppilastiedon luonti
    'Kuvaus: Skriptissä luodaan kuvitteellinen uusi oppilas joka saa
    'Oppilaan tiedot lehdellä seuraavat tiedot: oppilas, opiskelijanumero,
    'tila, koulutusohjelma, ryhmä ja aloittanut (aloitus pvm) ja
    'Yhteystiedot välilehdellä:

    Window SetContext, "Name=OracleRootWindow", ""
    MenuSelect "Sovellukset->Asemien perustiedot->Aseman tiedot"

    Window SetContext, "Caption=BSSS01 - OPPILAAN TIEDOT;ChildWindow", ""
    PushButton Click, "Text=Uusi"
    InputKeys "m2meika" 'oppilastunnus
    EditBox Click, "ObjectIndex=124", "Coords=103,11"
    InputKeys "Matti Meikäläinen" 'nimi
    EditBox Click, "ObjectIndex=126", "Coords=106,11"
    InputKeys "01234567" 'opiskelijanumero
    ComboBox Click, "ObjectIndex=7", "Coords=133,12"
    ComboListBox Left_Drag, "ObjectIndex=7", "Coords=92,13,88,23" 'tila
    ComboBox Click, "ObjectIndex=1", "Coords=189,8"
    ComboListBox Left_Drag, "ObjectIndex=1", "Coords=137,43,133,52" 'koulutusohjelma

    GenericObject Click, "Class=uiViewcore;ClassIndex=2", "Coords=9,5"
    Window SetContext, "Caption=Ryhmä", "" 'Ryhmä, esim I224-4, listassa ylin
    PushButton Click, "Text=OK"
    Window SetContext, "Name=OracleRootWindow", "" 'paluu lovista

    Window SetContext, "Caption=BSSS01 - OPPILAAN TIEDOT;ChildWindow", ""
    EditBox Click, "ObjectIndex=11", "Coords=48,15"
    InputKeys "t" ' t asettaa kuluvan päivämäärän Aloittanut päiväksi

    RadioButton Click, "Text=Yhteystiedot"
    EditBox Click, "ObjectIndex=127", "Coords=105,11"
    'asetetaan katuosoite ja postiosoite, jolloin postinumeron mukaan määräytyy
    'paikkakunta, voidaan valita myös lovista, esimerkissä syötetty manuaalisesti
    InputKeys "Katuosoite 1 A 3{TAB}33100{TAB}"
```

```
    EditBox Click, "ObjectIndex=18", "Coords=40,12"
    InputKeys "matti.meikalainen@ee.tpu.fi" 'annetaan e-mail osoite

    PushButton Click, "Text=Tallenna"
    RadioButton Click, "Text=Oppilaan tiedot"

    PushButton Click, "Text=Hakuehdot"      'tyhjennetään kentät
    EditBox Click, "ObjectIndex=125", "Coords=43,10"
    InputKeys "m2meika"
    PushButton Click, "Text=Hae"           'Haetaan juuri syötettyä tietoa

    'tarkistetaan vastaako m2meika haulla löydetty oppilaan nimi Matti Meikäläista
    'verification point
    Window SetTestContext, "Name=OracleRootWindow", ""
    Window SetTestContext, "Caption=BSSS01 - OPPILAAN TIEDOT;ChildWindow", ""
    Result = EditBoxVP (CompareData, "ObjectIndex=12", "VP=Object Data")
    Window ResetTestContext, "", ""

    PushButton Click, "Text=Sulje"

End Sub
```