LAUREA
AMMATTIKORKEAKOULU
Uuden edellä

Web-sovelluksen manuaalinen testaaminen Selenium IDE-työkalulla.

Najjar, Rian

2014 Leppävaara

Laurea-ammattikorkeakoulu
Leppävaara

Web-sovelluksen manuaalinen testaaminen Selenium IDE-työkalulla

Najjar Rian
Tietojenkäsittely
Opinnäytetyö
Huhtikuu, 2014

Laurea-ammattikorkeakoulu **Tiivistelmä**
Leppävaara
Tietojenkäsittely

Najjar, Rian

Web-sovelluksen manuaalinen testaaminen Selenium IDE-työkalulla.

Vuosi 2014 Sivumäärä 42

Opinnäytetyöni käsittelee Selenium IDE-työkalua osana manuaalista testausta web-sovelluksessa. Testaus on tärkeä osa-alue toimivassa web-sovelluksessa. Web-sovellusta ei voi päästää markkinoille ilman, että sitä testataan. Testauksen aikana yleensä löytyy virheitä, jotka joudutaan korjaamaan, jotta web-sovellus toimisi kokonaisuudessaan hyvin.

Opinnäytetyön teoriaosuudessa kartoitetaan toimintatapoja, käytäntöjä ja käsitteitä. Tarkoituksena saada näkemys mitä testaus on, miten se toimii käytännössä ja mikä työkalu helpottaa testausta. Tässä tapauksessa tarkastelen Selenium IDE-työkalua.

Opinnäytetyön toisessa teoriaosuudessa on tarkoitus kartoittaa Selenium IDE-työkalun käyttötarkoitusta, käytettävyyttä ja tarkoitusta. Tarkoituksena on kertoa mitä sillä voi tehdä, mihin se pystyy ja minkä takia sitä käytetään. Tuodaan esiin Selenium IDE-työkalun työkuvan.

Opinnäytetyössä tuloksena selkiytyi Selenium IDE-työkalun kokonaiskuva testauksessa. Selenium IDE -työkalu helpottaa testiajoa manuaalisessa testauksessa. Se säästää aikaa, rahaa ja resursseja. Selenium IDE-työkalua käytetään eräänlaisena nauhurina, jolla voi nauhoittaa toimintoja. Se on automaatio työkalu, joka toistaa esimerkiksi tekstinsyötön, niin ettei tekstiä tarvitse joka kerta manuaalisesti kirjoittaa uusiksi testin aikana.. Tämä helpottaa testiajoa huomattavasti. Selenium IDE –työkalu on käyttäjäystävällinen testiajoa tehtäessä. Lisäksi Seleniumin kotisivuilta ja keskustelupalstoilta löytyy työkalusta runsaasti tietoa ja avustavia ohjeita.

Avainsanat Selenium IDE, automaatiotyökalu, testiautomaatio

Laurea University of Applied Sciences **Abstract**
Laurea Leppävaara
Bachelor's Degree Programme in Information Technology

Najjar, Rian

Manual testing for web software using the Selenium IDE automatic tool

Year 2014 Pages 42

This thesis focuses on Selenium IDE, an automatic internet tool in manual testing. Testing is an important field in web software and it is impossible to launch web software on the market without testing the system.

The theory section of the thesis report surveys procedures, practices and context. The purpose of the thesis is to give an overview of what testing is, how it works in practice and which tool is proper and helpful to use in the testing area. In this case the Selenium IDE automatic tool is examined.

In the second section of the theory Selenium IDE is surveyed from the perspective of use, usability and purpose. The objective is to tell for what, how and why it can be used. Selenium IDE's function description is also presented.

As the final result of the thesis project, Selenium IDE's whole working area became clear. The Selenium IDE automatic tool facilitates the process when carrying out manual testing. It saves time, money and resources. Selenium IDE is used as a kind of recorder which can record functions. It is an automatic tool which can repeat text without typing the text all time again and again. It helps the writing process considerably. Selenium IDE is a user friendly automatic tool when operating the test case. Further information and helpful advice for the automatic tool can be found in the Selenium homepages and forum sites.

Keywords Selenium IDE, automatic tool, automatic testing

ALKUSANAT

Haluaisin kiittää, Teemu Vesalaa (laatukonsultti), asiantuntija-avusta tätä opinnäytetyötä tehdessä. Olen todella kiitollinen avustasi, ajastasi, neuvoistasi ja etenkin hermoistasi.

Olen kiitollinen ystävilleni ja perheelleni, jotka jaksoivat vuodesta toiseen kannustaa, avustaa ja uskoa valmistumiseeni.

Koululle kiitokset joustavuudesta.

Äidilleni, tädilleni, vaarilleni, iso-äidilleni ja niille muille, jotka eivät ole enää näkemässä valmistumistani, olen kiitollinen avustanne, ymmärryksestänne ja tukemisestanne vaikeina aikoina. Olisin halunnut valmistua vielä, kun olitte paikalla. Valitettavasti en ehtinyt.
Nyt kuitenkin sen tein.

Kaikille teille iso kiitos.

Nöyrästi Kiittäen
Rian Najjar

Espoossa 30.04.2014

Sisällys

1 Johdanto

Web-sovellus on valtavan laaja kokonaisuus, johon sisältyy hyvin paljon mm. tietoa, tekniikkaa ja tietoliikennettä. Jotta web-sovellus saadaan toimimaan kokonaisuudessaan, se vaatii mm. eri kokonaisuuksien toimimista keskenään, eri osa-alueiden järjestelmällistä hallintaa, paljon tekniikkaa ja rahoitusta. Monien eri asioiden ja osa-alueiden on toimittava keskenään. Jotta voitaisiin varmistua, että tuote, tässä tapauksessa, web-sovellus, olisi laadultaan ja kokonaisuudeltaan käytettävissä oleva tuote, se täytyy testata ennen laskemista markkinoille ja käyttöön.

Tähän on olemassa oma ammattikuntansa: testaajat. Testaajien tehtävänä on testata ohjelmisto ennen kuin se menee laajempaan levitykseen tai saavuttaa asiakkaat. Testaajat etsivät mm. bugeja eli virheitä, raportoivat niistä ja ilmoittavat eteenpäin seuraavalle taholle mahdollisista ongelmista. Kaikkia vikoja on mahdoton havaita, mutta suurimmat viat on mahdollista korjata. Vikojen korjaaminen kannattaa, koska ohjelmistosta saattaa löytyä virhe, joka estää ohjelmiston käytön. Virheestä on haittaa kaikille osapuolille ja tahoille. Testaajan kokonaistyökuva on hyvin laaja-alainen.

Kiinnostus tehdä opinnäytetyö johonkin testaukseen liittyvästä aihealueesta sain, kun vierailin vuosittain järjestettävillä Testaus – päivillä. Testauspäivät järjestettiin 04.06.2013 Aalto-yliopiston tiloissa Otaniemessä, Espoossa. Testauspäivät sisälsivät useita luentoja, testilabran ja testaus ympäristön. Osallistuin useille luennoille, testilabran tarjoamille harjoitus ja testaus ympäristöön sijoitettujen tehtävien tekoon. Testauspäivät oli todella onnistunut kokonaisuus ja sitä voi suositella kaikille alasta kiinnostuneille. Testipäivillä sai paljon uusia ideoita, syventää vanhaa tietämystä ja tutustua käytännössä testaustyöhön. Halusin hyödyntää oppimaani ja kysyin asiantuntijoilta ideoita opinnäytetyöhöni. Testausalan laatukonsultti suositteli minulle syventymistä alaan ja yhdessä tutkimme sopivaa aihetta. Päädyimme Selenium IDE-työkaluun, koska se on tärkeänä osana manuaalista testiajoa.

Selenium IDE työkalu on automaatiotyökalu, joka toistaa tehdyn toiminnon. Se helpottaa manuaalista testaamista käytännössä. Selenium IDE-työkalu on eräänlainen nauhuri, joka mm. nauhoittaa toimintoja. Selenium IDE on Firefox-selaimen lisäosa. Tarkastelen Selenium IDE-työkalun käyttötarkoitusta, toimintatapaa ja hyötyä. Selenium HQ tuoteperheeseen kuuluu erilaisia tuotteita. Tarkastelen opinnäytetyössä nimenomaan Selenium IDE automaatiotyökalua.

Tiedonhankintana hyödynnän testausalan ammattikirjallisuutta, web-sivustoja, sosiaalista mediaa ja alaan liittyvää kirjallisuutta. Opinnäytetyöni koostuu kirjallisesta materiaalista, web-sivustojen testaus alaan liittyvistä ohjelmistoista ja internetistä haetuista, luotettavasta tiedosta.

1.1 Työn tehtävä ja tavoite

Opinnäytetyössä pyritään hahmottamaan testauksen kokonaisuus yleisellä tasolla. Opinnäytetyö rajataan yhteen työkaluun. Tutustutaan Selenium IDE-työkaluun. Tarkastellaan työkalun käyttötarkoitusta, käytettävyyttä, hyötyä, ja tehokkuutta osana manuaalista testiajoa.

Tavoitteena on ymmärtää ja muodostaa kokonaiskäsitys Selenium IDE työkalusta osana testiautomaatiota, joka on osana manuaalista testausta. Tarkastelussa on Selenium IDE 2.4.0-versio. Selenium IDE:stä on olemassa vanhempia ja uudempia versioita. Selenium IDE:n ulkoasu muuttuu hieman, riippuen mikä versio on käytössä. Perustoiminnot ovat melko lailla sama ja suorituskyky testiajossa ei juuri muutu, oli versio mikä tahansa.

Minulla ei ole toimeksiantoa millekään yritykselle, joten olen tehnyt omia testitapauksia ja rakentanut testiautomaation niiden ympärille. Esimerkit olen ottanut tehdyistä ja suoritetuista omista testitapauksista. Yritän tehdä ne mahdollisimman yksinkertaisessa muodossa. Standardisoidut ja sertifioidut mallit ovat erikseen ilmoitettu tekstissä.

Käsitteitä avataan, koska käsitteet ja sanasto ovat vaikeasti ymmärrettävää jos ei ole itse tekemisissä alan kanssa. Tekstissä käytetään englanninkielisiä termejä, koska ne ovat yleisesti tunnettuja ja standardisoituja. Suomennettuna ovat ne osat tekstiin, joille on suomenkielinen vastine. Osaa käsitteistä ei voi oikein suomentaa, koska niille ei ole vastineita. Tällöin käytetään englannin kielisiä nimiä. Osassa tekstiä on sekä suomen, että englanninkielinen versio, selkeyden vuoksi.

1.2 Selenium IDE

Selenium IDE (engl. Integrated Development Environment) on integroitu kehitysympäristö Selenium skripiteille. Selenium:lla on mahdollisuus tehdä monia toimintoja. Näitä ovat mm. skriptien nauhoitus, editointi, ja debuggaus. Selenium IDE:llä on mahdollisuus myös käsin korjata ja muotoilla skriptiä tai muuttaa nauhoitettavan skripitin kapasiteettia.

Selenium IDE integroituna kehitysympäristönä antavat laajat mahdollisuudet soveltaa skriptiä automaatio testaukseen. Automatisoitu testiajo yhdistettynä nopeasti käsin korjattavaan

skripitiin, nopeuttaa testiajon tekoa. Päivitettyjä versioita tulee säännöllisin väliajoin Selenium:in kotisivuille.

Opinnäytetyössä tutustutaan Selenium IDE automaatio työkaluun. Selenium tuoteperhe on laaja. Siihen kuuluu paljon eri tuotteita ja toimintoja. (Selenium IDE 2013)

1.3 Sanasto ja käsitteitä

Nämä sanastot ja käsitteet ovat testiajoa varten koottu tähän listaan. Ne auttavat ymmärtämään eri tilanteissa esiintyviä komentoja, toimintoja tai käskyjä. Sanastoa ja käsitteitä käytetään It-alalla yleisesti. Ne saattavat hieman tarkoittaa eri asioita, riippuen siitä mihin niitä käytetään.

automaatio	automaattinen toisto, tarkoittaa tässä tapauksessa nauhoitettua toimintoa, joka toistaa toiminnot kerta toisensa jälkeen jatkuvasti.
bugi	virhe järjestelmässä (koodivirhe)
check list	tarkistuslista. Listataan merkitään näkyviin toiminnot ym. tärkeät asiat, jota tulee suorittaa vaihe vaiheelta.
debuggaus	vikojen jäljitys ja korjaaminen ohjelmistokoodista
editointi	korjaustoimenpide. Korjaustoimenpide voi olla mikä vaan toimenpide mikä pitää korjata. Se voi olla skriptin korjaamista tai ohjelmiston korjaamista.
error	virhe. Tarkoittaa virhettä koodissa, minkä voi mahdollisesti korjata.
fault	virheellinen ohjelmakoodi.
failure	vika joka aiheuttaa häiriön ohjelmistossa
informaatio	tieto, data
input	syöte

invalid	epäkelpo
kieli	tarkoitetaan ohjelmisto kieltä. Niitä on erilaisia ja ne toimivat eri tavalla, eri ohjelmistoissa Eri ohjelmointikielet eivät tunnista toisiaan ilman kääntäjää.
kääntäjä	kääntää koodikielen toiselle kiellelle.
manuaalinen	ei-automaattinen toisto, käsin kirjoitettu tai tehty toiminto tai asia
maksimi	isoin (arvo)
minimi	pienin (arvo)
normi	sovittu määritelmä, miten asian tehtävän tulisi olla
output	tulos
skripti	komentosarja
standardi	normi, määritelmä minkä mukaan asia tulisi tehdä
ISO-standardi	kansainvälisen standardointi järjestön normi.
testaus	ohjelmiston testausta vallitsevilla menetelmillä
validi	kelpaava/voimassa oleva

It-alalla koodi-ja käyttökieli on yleensä englanti. Suomenkielisiä vastineita on olemassa ja niitä saatetaan käyttää jossain kohtaa testiajossa. Jos esimerkiksi testiautomaatio työkalun asetukset ovat englanninkieliset, niin automaatio ei ymmärrä suomenkielisiä vastineita.

2 Laadun määritelmä

Laatua voidaan määritellä monella eri tavalla. Laadun määritelmällä ei ole yhtä ja oikeaa näkemystä. Esittelen kaksi eri näkemystä asiasta, jossa määritellään laatua eri näkökulmista. Näkemykset eroavat toisistaan.

”Pyritään tekemään sitä, mitä asiakas haluaa ja tarvitsee. Laadukas ohjelmisto tuottaa rahaa tai säästöjä liiketoimintamallin odottamalla tavalla. Ei tehdä mitään turhaa tai mitään sellaista, mistä ei ole hyötyä projektin kannalta. Ensisijaisesti pyritään aina tekemään voittoa.” (Laatukonsultti. 2014)

”Laatu on todettua yhdenmukaisuutta vaatimusten kanssa. ” (Ruuska 2014, 234.)

Laatukonsultin tekemä määritelmä perustuu käytännön kokemukseen It-alalta. Kai Ruuskan näkemys perustuu teoriaan. Teoria ja käytäntö eroavat toisistaan käytännön työssä.

3 Mitä testaus on?

Testaus on ohjelmiston testausta. Testaus on eri osa-alueiden testausta, kokeilua ja yhteensovittamista. Testitapoja on erilaisia, joita sovelletaan kunkin projektin kohdalla. Testaus vaatii hyvää kokonaisnäkemystä asioista, paineensietokykyä ja järjestelmällisyyttä. Testausta tehdään, jotta kokonaisuus toimisi.

Testausta tehdään, jotta voitaisiin varmistua ohjelmiston laadusta. Toimiva ohjelmisto säästää aikaa, rahaa ja resursseja. Liiketoimintamallin mukainen, laadukas ohjelmisto tuottaa liikevoittoa. Pyritään tekemään se mitä asiakas haluaa.

Ohjelmistosta saattaa myös löytyä virheitä testiajon aikana. Osalla virheistä ei ole merkitystä, mutta virheet voivat olla kalliita, hankaloittaa toimintaa, ovat aikaa vieviä ja osa jopa vaarallisia.

Ihmiset tekevät inhimillisiä virheitä. He eivät välttämättä huomaa omia virheitään. Sen takia olisi hyvä testata tai antaa jonkun muun testata ja tutkia oma työ, jotta vältyttäisiin virheiltä. Ohjelmiston toimivuus ja tuottavuus ovat päämääriä.

Therac-25 on surullisen kuuluisa esimerkki epäonnistuneesta ohjelmistosta, joka tuhosi muutamien ihmisten elämän vuosina 1985–1987. Kyseessä oli säteilykone Therac-25 johon oli asennettu virheellinen ohjelmisto. Kone ei tunnistanut säteilyn määrää vaan säteilytti pienen linssin läpi liian suuren määrän säteilyä. Muutamia ihmisiä kuoli liian suureen säteilymäärään.

Testaus säästää aikaa, rahaa ja lisää käytettävyyttä. Toimiva sovellus tuottaa enemmän hyötyä, toimivuutta, käytettävyyttä, liikevoittoa, laatua ja näkyvyyttä.
Testaamatonta sovellusta ei pitäisi koskaan päästää tuotantoon, koska se saattaa sisältää virheitä tai ei toimi halutulla tavalla. Testaamaton sovellus tuottaa tappiota, on hyödytön ja voi olla vaarallinen.

4 Testauksen eri osa-alueita

Testiautomaation voi tehdä monella eri tavalla. Testaus sisältää eri osa-alueita ja erilaisia tapoja testata sovellus, joita voidaan räätälöidä projektin mukaan. Taitava testaaja osaa valita projektin kannalta oleellisimmat apuvälineet ja toimintatavat. Testaaminen voi olla työlästä ja aikaa vievää jos testiajo suoritetaan vain manuaalisesti. Projektin ja prosessin kannalta on järkevää automatisoida osa toiminnoista. Muutamia erilaisia menetelmiä on esitelty seuraavissa alaotsikoissa.

Onnistunut testiajo tarvitsee myös muita osa-alueita, toimiakseen. Opinnäytetyössä on esitelty testaukseen liittyviä teoriaosia, joita tarvitaan, kun suoritetaan testiajoa. Testiajot ovat erilaisia ja ne rakennetaan tarpeen mukaan.

Eri tekniikkoihin kannattaa tutustua huolella. Ne eroavat toisistaan. Testiajo voi epäonnistua, jos valitaan väärä tekniikka. Projektin kannalta vain oleellisimmat asiat kannattaa testata.

4.1.1 Black box testing

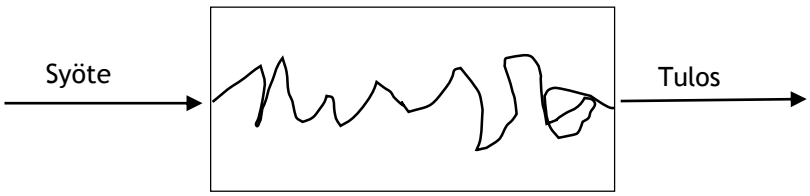
Black Box testing (engl.) eli mustalaatikkotestauksella tarkoitetaan sitä, että koodisto ei ole näkyvissä ja laatikko prosessoi ”itsekseen” toiminnon. Testaaja ei näe koodia, eikä pysty vaikuttamaan siihen. Testaaja ei tiedä miten systeemi tai komponentti on rakennettu. Mustalaatikkotestauksessa testaaja keskittyy ohjelmiston toimimiseen, ei siihen miten syöte käsitellään koodin sisällä.

Kuva 1: Mustalaatikko (engl. black box)

Input:lla tarkoitetaan sisääntulo syötteitä. Input on syöte, joka menee laatikkoon. Output on tulos, joka tulee ulos mustasta laatikosta. Testaaja valitsee ja tutkii odotetut arvot.(Tutorialspoint 2014.)

4.1.2 White box testing

White box testing (engl.) eli valkolaatikkotestauksella tarkoitetaan sitä, että koodisto on näkyvissä. Testaaja pystyy seuraamaan koodia ja saamaan tietoa koodiston rakenteesta. Valkolaatikkotestauksessa testaajan tarvitsee tutkia lähdekoodia ja rakennetta, jotta testaaja saa selville, mitä lähdekoodi sisältää. Lähdekoodista näkyy mahdolliset virheet tai miksi lähdekoodi käyttäytyy ei-toivotulla tavalla.
White Box- testausta kutsutaan myös valkolaatikko tai lasilaatikkotestaukseksi.(Tutorialspoint 2014.)



Kuva 2: Valkolaatikkotestaus (engl. white box testing)

4.1.3 Grey box testing

Grey box testing (engl.) eli harmaalaatikkotestauksella tarkoitetaan mustalaatikko- ja valkolaatikkotestauksen välimuotoa. Harmaalaatikkotestauksessa pyritään hyödyntämään musta-ja valkolaatikkotestauksen parhaat puolet. Testaajalla on käytössä paljon tietoa ohjelmistosta ja koodistosta, joita hän voi käyttää saadakseen projektin kannalta parhaan lopputuloksen. (Tutorialspoint 2014)

4.1.4 Equivalence partitioning

Ekvivalenssiluokat (engl. eguivalence partitioning) tarkoittavat syötearvojen jakoluokkia. Syötteiden määrä on valtava. Samankaltaiset syötteet jaetaan luokkiin, jotta niitä olisi selkeämpi käsitellä. Testaaja testaa testitapauksia arvojen perusteella. Testaaja tutkii taulukoista arvoja ja rajaa testitapausten määrän minimiin, koska muuten testattavia testitapauksia tulee liikaa. Testaajalla ei ole aikaa, eikä resursseja tehdä ylimääräisiä testitapauksia. (Eguivalence partitioning, 2014)

Testaaja ottaa jokaisesta luokasta muutaman arvon ja testaa arvojen perusteella testitapauksen. Testitapausten perusteella voidaan olettaa tulos testattavassa järjestelmässä. Jos esimerkiksi otettu on arvo on negatiivinen, niin se antaa testitapausten yhteydessä

negatiivisen tuloksen. Tulos on silloin invalidi. Jos taas otetaan positiivinen arvo testattavaksi, niin se antaa positiivisen tuloksen. Tulos on silloin validi. Testaaja ei voi kuitenkaan olettaa mitään arvoja, vaan testaajan on testattava valitut arvot, jotta tulos on varmasti oikea ja tarkoituksenmukainen.

Esimerkiksi jos tarkastelemme taulukkomuodossa Suomen alkoholilainsäädännön määräämiä ikärajoja, niin taulukossa on kolme saraketta. Alkoholilainsäädännössä on määrätty ikärajat alkoholituotteiden tilavuuden mukaan. Ensimmäinen luokka on alle 18 vuotiaat. Siihen kuuluu kaikki alle 18-vuotiaat. Toinen luokka on 18-20 ikävuotta täyttäneet. Kolmas luokka on 20 ikävuotta tai yli täyttäneet. Alle 18 vuotiaat eivät saa lainsäädännön mukaan ostaa alkoholia. 18-20 vuotiaat saavat ostaa mietoja alkoholeja. Yli 20-vuotiaat saavat ostaa mitä tahansa alkoholijuomia.. Laki velvoittaa tarkastamaan alkoholin ostotilanteessa henkilön virallisen henkilötodistuksen. Ala sarakkeessa, toisella alarivillä on ikiä, mitä voidaan merkitä, esimerkkinä ikäraja tarkistuksista.

Alle 18 vuotta	18 tai yli vuotta alle 20	20 tai yli vuotta
10, 15,17	18, 19	22, 25, 35

Kuva 3: Ekvivalenssi-luokat

Lainsäädännön mukaan henkilötodistus on näytettävä. Se kirjataan kahteen eri järjestelmään, sähköiseen järjestelmään ja manuaaliseen järjestelmään. Näin tehdään, jotta järjestelmiä pystytään verrata keskenään. Ekvivalenssiluokat helpottavat järjestelmän seurantaan ja arvojen kirjaamista.

4.1.5 Boundary value analysis

Boundary value analysis (engl.) tarkoittaa raja-arvoanalyysiä. Raja-arvoanalyysillä tarkoitetaan arvoja, jotka sisältyvät tiettyjen rajojen sisäpuolelle. Kaikkia arvoja ei voi analysoida. Sen takia on käytettävä raja-arvoja, jotta saataisiin rajattua lukuja tiettyjen raamien sisään. Raja-arvoista saatavia tuloksia verrataan haluttuihin arvoihin.

Minimi ja maksimi arvojen sisälle saadaan rajatut arvot, muut arvot jäävät ulkopuolelle. Jos esimerkiksi testataan järjestelmää, jolle halutaan määrittää arvoja missä rajoissa arvojen tulee olla, niin minimi arvona voidaan pitää 1 ja maksimi arvona 99. Kaikki tuloksista saadut arvot välillä 1-99 mahtuvat raja-arvoon. Jos tuloksesta tulee nolla (0) tai 101, niin ne eivät mahdu raja-arvoon ja ovat invalideja arvoja. Nolla (0) arvona, kerää virheitä.
Saadut arvot ovat joko valideja tai invalidia. Invalidista arvosta tulee virheilmoitus.

Testaaja tarvitsee raja-arvoja, jotta voi tulkita syötteistä saatuja arvoja. Testaaja näkee arvoista onko tulos haluttu validi arvo vai ei. Esim. white box testauksessa saatu output arvo, testaaja arvioi ja vertailee. (Boundary value analysis 2014)

Esimerkiksi jos tarkastelemme raja-arvoja Suomen alkoholilainsäädännön määräämiä ikärajoja, niin taulukossa on kolme eri saraketta. Ensimmäinen sarake on alle 18-vuotiaat. Tämä tarkoittaa kaikkia, jotka ovat 18 vuotta täyttäneet. Esimerkiksi 18 vuotta ja 2kk täyttänyt ihminen ei voi kuulua alle 18-vuotiaiden sarakkeeseen. Se ei sisälly raja-arvoon. Seuraava luokka on 18-20 vuotta. Tämä tarkoittaa, että päivän yli 18 vuotiaasta, rajataan päivää vaille 20 vuotiaaseen. Kolmas luokka on 20 vuotta eli päivää yli 20 vuotta ja siitä eteenpäin. Yläikärajaa ei ole.

Alle 18 vuotta	18 tai yli vuotta	alle 20 vuotta	20 tai yli vuotta
päivää vaille 18 vuotta	18, päivän yli 18 vuotta	päivää vaille 20 vuotta	20, päivää yli 20 vuotta

Kuva 4: Raja-arvo luokat

Esimerkiksi testaaja voi ottaa ekvivalenssiluokkien ja raja-arvotaulukoiden arvoja ja testata niiden perusteella ikäryhmä testitapauksen. Testaaja testaa kuinka monta alle 18 vuotta täyttänyttä alkoholin ostoyritys kertaa on tehty. Testaaja arvioi tulokset.

Ekvivalenssiluokat ja raja-arvoanalyysit ovat samankaltaisia testimenetelmiä. Olen käyttänyt samankaltaisia taulukoita, jotta niistä näkyy pieni ero. Niitä käytetään yhdessä, jotta saadaan mahdollisimman tarkka tulos. Molemmat käytännöt tukevat toisiaan.

4.1.6 Check list

Check list (engl.) eli tarkistuslistalla tarkoitetaan listaa, johon listataan esim. toiminnot ja niiden toimivuus, käytettävyys ja mahdolliset viat. Tarkistuslistoja on erilaisia ja eritasoisia. Tarkastuslistoihin voi kirjoittaa vaikka koko prosessin toimivuuden ja vikatilanteet, jos se on mahdollista. Tarkistuslistoja tehdään muistin tueksi ja helpottamaan suunnittelua. Tarkistuslista ei ole ohje vaan siihen on koottu joko laajasti tai suppeasti prosessin tapahtumia.

Tarkistuslistaa voidaan käyttää yleisesti tai se voi olla vain yhden henkilön tukena, muistuttamassa tehtävistä. Testaajilla on omat tavat työskennellä ja jotkut testaajat tarvitsevat vain osia tarkistuslistasta.

Tarkistuslista pitää säilyttää asiallisesti asiallisessa paikassa. Tietosuojan vuoksi, sitä ei saa julkisesti näyttää ulkopuolisille. Se on osa tietoturvallisuutta.

4.1.7 Dokumentointi ja raportointi

Testausta suoritettaessa dokumentointi ja raportointi ovat tärkeitä osia projektin kannalta. Dokumenteista ja raporteista näkee mitä on tehty, milloin ja missä vaiheessa. Dokumentit ja raportit kertovat tarvittavan informaation eri sidosryhmille.

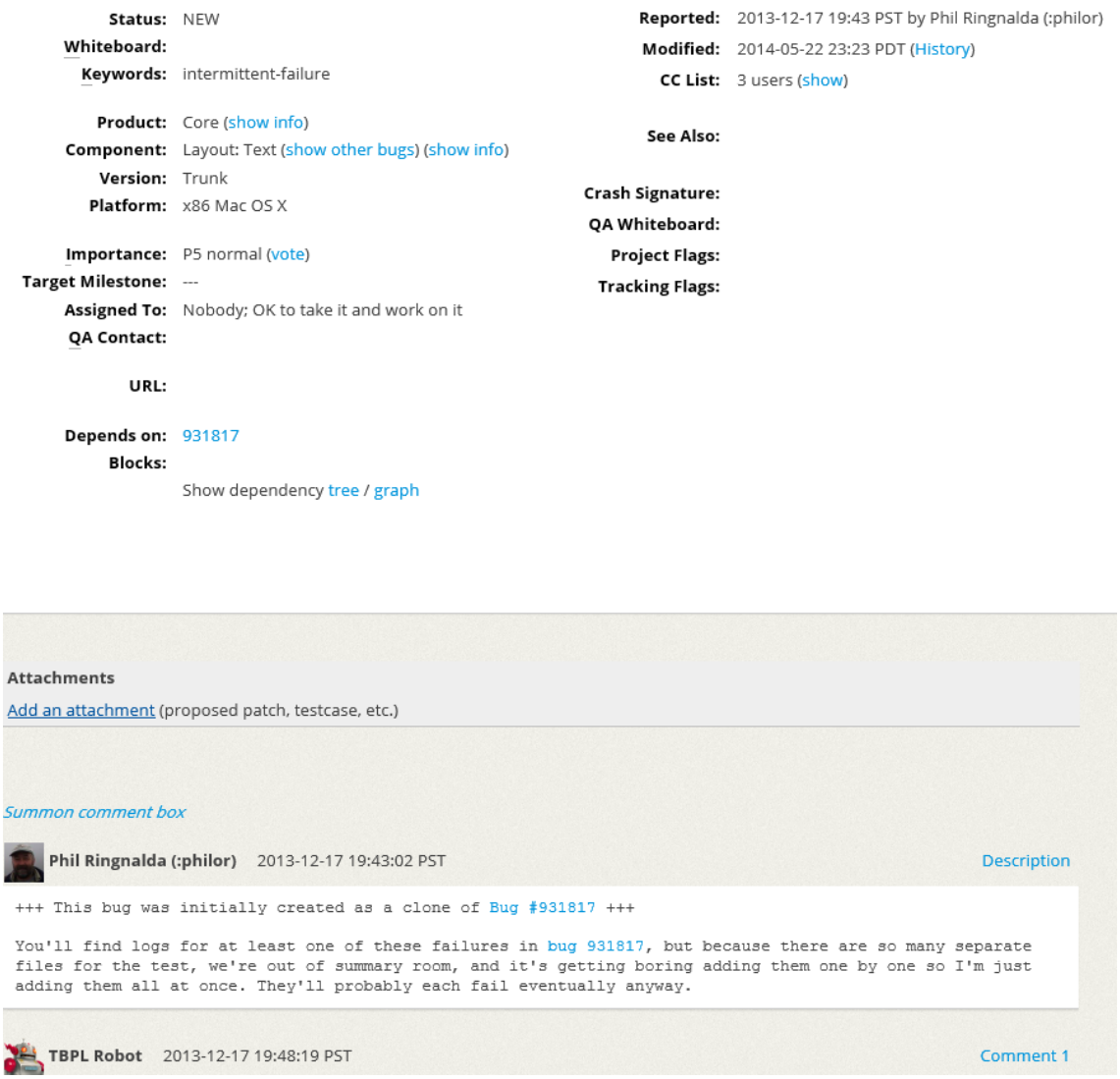
Dokumentoinnilla tarkoitetaan tallennettua informaatiota. Dokumentaatio perustuu asiakeskeiseen informaatioon. Projektin kannalta on tärkeätä, että informaatio on faktaa. Informaatiota käytetään ilmaisemaan todellinen fakta. Dokumentteja saatetaan käyttää projektin muissa yhteyksissä, joten on erittäin tärkeätä, että informaatio on faktaa, ajanmukaista ja oikeellista.

Raportoinnilla tarkoitetaan selontekoa asiasta, joka keskittyy olennaisten tietojen välitykseen. Raportit voivat olla hyvinkin laajoja, asiakeskeisiä selontekoja. Raportteihin voi sisältyä asiatekstin lisäksi esimerkiksi taulukoita, virheilmoituksia ja korjaustoimenpiteitä.

4.1.8 Virheraportti

Testiajoa tehdessä testaaja saattaa löytää virheitä. Osalla virheistä ei ole merkitystä kokonaisuuden kannalta, mutta haitallisemmat virheistä pitää korjata. Muuten ne voivat hankaloittaa ohjelmiston käyttöä, aiheuttaa ongelmia tai olla jopa hengenvaarallisia. Syitä virheiden ilmaantumiseen testiajossa voi olla useita. Usein virhe on lähdekoodissa.

Virheen löytyessä, testaajan täytyy analysoida virheen laatu ja tehdä virheraportti. Virheraportista tulee ilmetä mahdollisimman tarkasti kaikki virhettä koskeva tieto niin että virheen korjaaja saa mahdollisimman tarkat tiedot virheestä. Virheraportti tulee täyttää (katso kuva 5) asiallisesti, huolella ja ymmärrettävästi ja toimittaa nopeasti organisaatiossa eteenpäin.



Status: NEW
Whiteboard:
Keywords: intermittent-failure

Product: Core (show info)
Component: Layout: Text (show other bugs) (show info)
Version: Trunk
Platform: x86 Mac OS X

Importance: P5 normal (vote)
Target Milestone: ---
Assigned To: Nobody; OK to take it and work on it
QA Contact:

URL:

Depends on: 931817
Blocks:
Show dependency tree / graph

Reported: 2013-12-17 19:43 PST by Phil Ringnalda (:philor)
Modified: 2014-05-22 23:23 PDT (History)
CC List: 3 users (show)

See Also:

Crash Signature:
QA Whiteboard:
Project Flags:
Tracking Flags:

Attachments
Add an attachment (proposed patch, testcase, etc.)

Summon comment box

Phil Ringnalda (:philor) 2013-12-17 19:43:02 PST Description

+++ This bug was initially created as a clone of Bug #931817 +++

You'll find logs for at least one of these failures in bug 931817, but because there are so many separate files for the test, we're out of summary room, and it's getting boring adding them one by one so I'm just adding them all at once. They'll probably each fail eventually anyway.

TBPL Robot 2013-12-17 19:48:19 PST Comment 1

Kuva 5: Bugiraportti

Virheraportti toimii selontekona virheestä. Se liitetään muiden raporttien ja dokumenttien mukaan. Projektin johdosta riippuu, miten siihen suhtaudutaan.

4.1.9 Ongelmat

Testausta suorittaessa saattaa tulla odottamattomia ongelmia eteen. Ohjelmistossa voi olla vakavanlaatuisia virheitä, jotka tulee korjata. Korjaamattomina virheet saattavat aiheuttaa vakavaa haittaa. Testiautomaatio suorittaa halutun eli koodatun toiminnon. Testiautomaatio ei korjaa virheitä vaan ilmoittaa missä kohtaa ja minkälainen virhe on. Virheen vakavuudesta riippuu, miten projektissa edetään. Virheet analysoidaan ja tarvittaessa ryhdytään jatkotoimenpiteisiin.

Ongelmien ilmaantuessa, testaajan on ilmoitettava ongelmista seuraaville tahoille. Ongelma tilanteissa testaaja joutuu olemaan tekemissä eri sidosryhmien kanssa. Tämä saattaa tarkoittaa jopa konsultaatiota projektin johdon kanssa.

Projektin johto asettaa rajat, missä määrin ohjelmistoa voidaan korjata. Testaaja joutuu perustelemaan korjaamistarvetta eri sidosryhmille. Korjaustarpeesta päättää projektin johto.

5 Prosessimallit

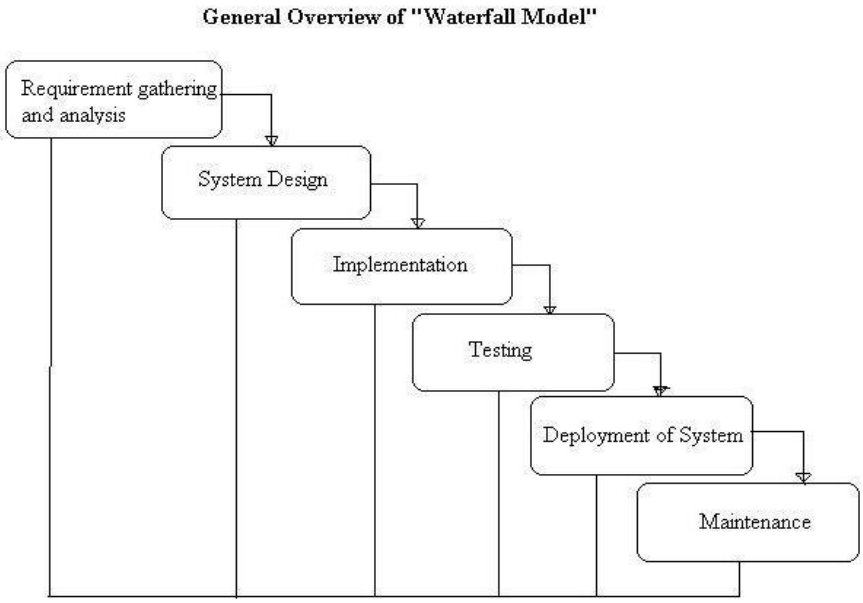
Projekteissa voidaan käyttää monenlaisia prosessimalleja. Yleisesti tunnettuja prosessimalleja käytetään maailmanlaajuisesti, koska prosessien standardisointi helpottaa työn tekemistä, hahmottamista ja (testi)tulosten tarkastelua. IT-alalla on hyvin yleistä, että osa prosesseista ulkoistetaan toisiin maihin. Esimerkiksi vaatimusmäärittelyt voidaan suorittaa Suomessa ja tekninen toteutus tehdä Intiassa. Tunnetut prosessimallit auttavat yhdentämään käytäntöjä. Esittelen tässä muutaman tunnetun prosessi mallin, jota käytetään, kun tehdään testiajoja.

Vesiputousmalli ja V-malli ovat yleisesti tunnettuja malleja maailmanlaajuisesti. Aasiassa mallit ovat tunnettuja ja niitä käytetään yleisesti. Länsimaissa ja Aasiassa ovat yhtenevät mallit käytössä. Se helpottaa työskentelyä.

Vesiputousmalli ja V-malli eroavat toisistaan. Ne ovat samankaltaisia malleja, mutta niissä on eroavaisuuksia. Projektin koosta riippuu, kumpaa mallia käytetään vai valitaanko joku muu prosessimalli.

5.1 Vesiputousmalli ja testaus

Vesiputousmalli (engl. waterfall model) on yleisesti tunnettu prosessimalli. Vesiputousmalli on ensimmäinen prosessimalli, joka on esitelty yleisesti (katso kuva 5). Se on vaiheellinen (lineaarinen) ohjelmistotuotannon malli. Vesiputousmallia käytetään suunnittelu-ja tuotantomallina projekteissa. Vesiputousmallia tulkitaan siten, että jokainen vaihe käydään läpi ja suoritetaan loppuun ennen kuin siirrytään seuraavaan vaiheeseen. Tässä mallissa ei palata takaisin edelliseen vaiheeseen vaan jokainen vaihe käydään läpi ja siirrytään seuraavaan. Jos osiossa on jotain ongelmaa, niin se korjataan, tehdään valmiiksi ennen kuin siirrytään seuraavaan. Tuloksia tarkastellaan ja vertaillaan ennen kuin siirrytään seuraavaan vaiheeseen.



Kuva 6: ISTBQ standardi vesiputousmalli (ISTBQ 2013)

Vasemmalta ylhäältä ensimmäinen vaihe on vaatimusten määrittely (engl. reguirement gathering and analysis). Tässä vaiheessa määritellään vaatimukset ohjelmistolle. Vaatimusmäärittelyn tekevät eri sidosryhmät. Ne voi olla mitä vaan esim. ohjelmiston laadusta laitteistoon. Tämä on tärkeä vaihe, koska ohjelmiston kokonaiskuva hahmottuu tässä kohtaan.

Seuraava vaihe on suunnittelu (engl. system desing). Tässä vaiheessa suunnitellaan mm. tekninen rakenne, komponentit, tietorakenteet yms. tietopuoliset kokonaisuudet. Suunnitteluvaihe on aikaa vievä ja tärkeä vaihe, koska se pitää sisällään informaation rakenteen josta kehitellään esim. toimivia komponentteja.

Seuraava vaihe on toteutus (engl. implementation). Tässä vaiheessa keskitytään saamaan ohjelmistosta toimiva kokonaisuus. Toteutuksessa keskitytään ohjelmiston toimivuuteen ja käytettävyyteen.

Seuraava vaihe on testaus (engl. testing). Testataan ohjelmistoa, komponentteja yms. Jos löytyy virheitä, niin ne korjataan ja testataan uudelleen.

Seuraava vaihe on käyttöönotto (engl. deployment of system). Ohjelmisto otetaan käyttöön. Käyttäjiä koulutetaan käyttämään ohjelmistoa. Käyttäjät antavat palautetta ohjelmiston toimivuudesta ja rakenteesta.

Seuraava ja viimeisin vaihe on ylläpito (engl. maintenance). Reaaliaikaiset päivitykset ja tarvittavat korjaukset.

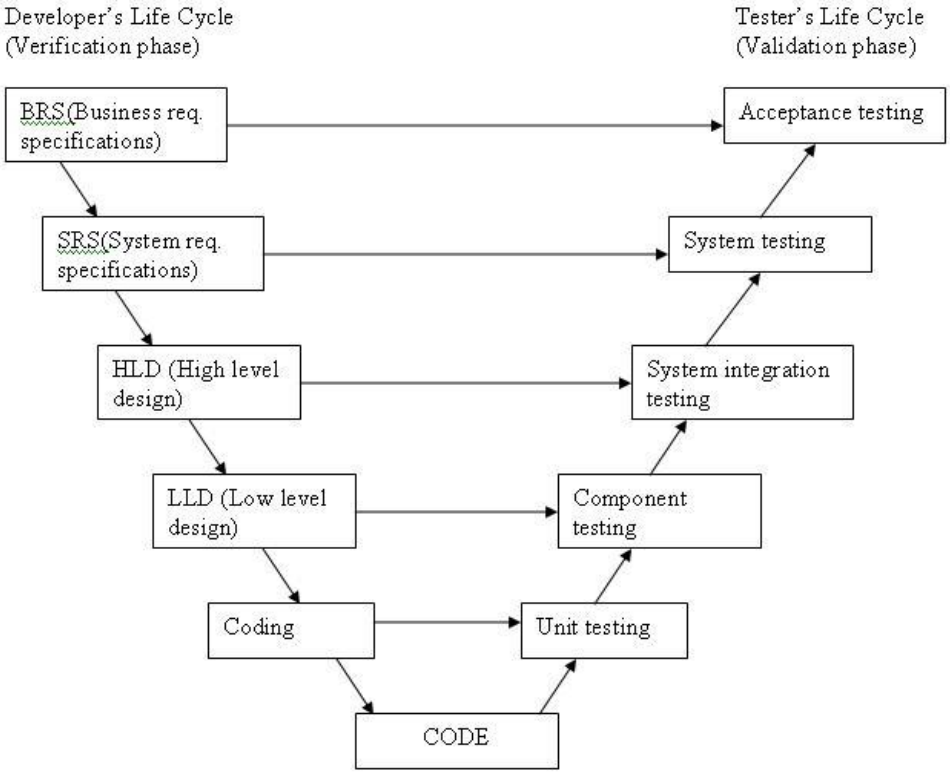
Vesiputousmallin eduiksi voidaan lukea sen helppokäyttöisyys ja selkeys. Vesiputousmallissa näkyy selkeästi, että missä vaiheessa prosessia ollaan menossa ja mikä seuraava polku on. Vesiputousmalli sopii pieniin projekteihin.

Vesiputousmallin hankaluuksiin voidaan lukea muutosten korjaaminen. Jos esim. testauksessa ilmenee virheitä, niin niitä on vaikea korjata, koska vaihetta ei voi mennä taaksepäin. Jos virhe löytyy jostain keskiväliltä prosessia, niin sen korjaaminen voi tulla todella kalliiksi. Vesiputousmalli on luonnehdittu hankalaksi ja kankeaksi malliksi. Vesiputousmalli ei toimi käytännöntyössä hyvin. (ISTQB, Vesiputousmalli 2014)

5.2 V-Malli

V-malli on prosessimalli. V-malli on johdettu versio vesiputousmallista. Johdetulla mallilla tarkoitetaan sitä, että jokaiseen vaiheeseen lisätään aihetta vastaava testaus. V-malli on myös vaatimusmäärittelydokumentti. V-malli helpottaa projektin hallintaa. Kuviota tulkitaan niin, että esim. ohjelmointi vaatimusmäärittelyt testataan yksikkötestauksena, tarkastellaan tuloksia ja jos ne on hyväksyttäviä, niin siirrytään seuraavan vaiheeseen etc.

V-mallin ideana on, että jokainen prosessi (laatikko) käydään läpi ennekuin siirrytään seuraavan vaiheeseen. V-mallilla tarkoitetaan veritification ja validation prosessimalleja. V-mallia käytetään suunnittelussa ja käytännöntyössä. V-mallia toteuttaessa tiedetään kuka tekee mitäkin ja missä vaiheessa ollaan menossa. V-mallia lähdetään toteuttamaan vasemmasta ylärivistä ja edetään alaspäin. Sitten edetään taas ylöspäin. Toiminnot toistetaan niin kauan, kun on saavutettu haluttu tulos. V-mallia käytetään pienissä ja keskisuurissa projekteissa.



Kuva 7: ISTBQ standardi V-malli ohjelmistomalli (ISTBQ 2013)

Vaatimusmäärittelyt on tässä dokumentissa aloitettu V-mallin vasemmalta puolelta (katso kuva 6). Testauksen suunnittelu ja tulosten vertailua tapahtuu joka tasolla. V-mallia kuviota tutkiessa, ensimmäinen laatikko vasemmalta on BRS (engl. Business Requirement Specification) eli vaatimusmäärittely, jossa on mukana liiketoiminta.

Seuraava vaihe on SRS (engl. System Requirement Specification) eli järjestelmä vaatimus määrittely. Tässä mallissa tehdään liiketoimintamalli asiakkaan ja eri sidosryhmien kanssa. Tässä kohtaa suunnitellaan tulevaa järjestelmää mitä sisältöä sinne tulee yms. Järjestelmälle asetetaan toiminnallisia ja ei-toiminnallisia vaatimuksia.

Seuraava vaihe on HLD (engl. High level desing) joka tarkoittaa arkkitehtuurin suunnittelumallia. Tässä vaiheessa yleiskatsaus ratkaisuihin, sovellusalustoihin, tuotteeseen ja palveluun/sovellukseen.

Seuraava vaihe on LLD (engl. Low level desing) eli suunnitteluvaihe. Tällä tasolla suunnitellaan logiikka eli käyttöjärjestelmä komponentteihin. Komponenttien testaus suoritetaan tällä tasolla myös.

Seuraava vaihe on koodaus (engl. Coding). Tällä tasolla sovellusta laitetaan toimeenpanoon. Toteutus (engl. implementation) suoritetaan tällä tasolla.

Seuraava vaihe on ohjelmointia (engl. Code). Tämä on alin vaihe, jossa tuloksena on koodi. Tästä siirrytään V-mallin oikealle puolelle toisenlaisiin toteutuksiin eli testaamiseen eri tavalla.(ISTQB V-malli 2013)

Testaussuunnitelmat on sijoitettu V-mallin oikealle puolelle. V-mallin oikeinpuoleisen alimmainen vaihe on yksikkötestaus (engl. Unit testing). Yksikkötestaus tarkoittaa yksittäisten ohjelmistokomponenttien testausta. Yksikkötestaus tunnetaan myös moduuli-ja komponenttitestauksena. Ne tarkoittavat samankaltaista testausta.

Seuraava vaihe on komponentti testaus. (engl. Component testing) Tämä on samankaltaista testausta kuin yksikkötestaus. Yksikkötestausta ja komponenttitestausta voidaan kutsua moduulitestaukseksi. Moduulitestauksella tarkoitetaan, että testataan yksittäinen moduuli. Moduuli on yksittäinen komponentti eli yksikkö. Moduulin sisältö on tietomäärittelyä ja erilaisia funktioita. Funktiot sisältävät käsiteltävää tietoa. Yleisesti testaaja käyttää mustalaatikkotestaus tekniikkaa. Jotkut prosessit vaativat lasilaatikkotestauksen, koska ovat vaativia. Testaajalla on käytössä tiedot kyseisestä komponentista mm. ohjelman lähdekoodi ja komponentin määrittelyt. Moduulin toimintaa verrataan arkkitehtuurinsuunnittelun tuloksiin ja moduulisuunnitteluun. Moduuleita verrataan tekniseen vaatimusmäärittelydokumenttiin.

Moduulitestauksen tarkoituksena on keskittyä sisäiseen tietorakenteeseen ja logiikkaan. Moduuli suorittaa vain yhden funktion kerrallaan. Tässä kohtaa testitapausten määrä vähenee ja virheet on helpompi havaita. Moduulitestauksessa yritetään löytää ristiriitoja määrittelyjen ja toiminnan välillä. Moduulitestaus ei käy ainoaksi testimenetelmäksi, koska moduulitestauksessa testataan vain yhtä moduulia, eikä laajaa kokonaisuutta.
(ISTQB V-malli 2013, V-malli, Software business competence 2014)

Seuraava vaihe on integraatiotestaus (engl. System integration testing). Integraatio testauksessa yhdistellään moduuleita ja moduuliryhmiä yhteen. Testaus aloitetaan pienemmistä komponenteista. Ne kerätään yhteen ja niitä aletaan testaamaan niin kauan että koko systeemi on saatu kasaan. Tässä keskitytään moduulien välisten rajapintojen toimivuuden tutkimiseen. Moduulitestaus ja integraatiotestaus etenevät yleensä samanaikaisesti. Ne tukevat toinen toisiaan.

Integraatiotestaus voidaan suorittaa kahdella tavalla. Yksi tapa on koota alhaalta ylöspäin alimman tason moduuleita. Toinen tapa on koota jäsentävällä eli osittavalla tavalla eli integroida päinvastoin.

Integraatiotestauksen tarkoituksena on, että kasatut osat toimivat keskenään määritellysti. Tarkoituksena on myös korjata virheitä ja yhtyeensopimattomia moduuleita. Virheitä saattaa löytyä, vaikka yksittäiset moduulit olisivatkin näyttäneet toimivilta. Testitapaukset muodostetaan siten, että voidaan havaita virheitä moduuleiden yhteistoiminnassa.

Seuraava polku on järjestelmätestaus (engl. System testing). Järjestelmätestauksessa on tarkoitus tarkastella koko järjestelmää. Tuloksia on tarkoituksena verrata määrittely vaiheessa olevaan dokumentaatioon. Järjestelmätestauksen suorittaa sellainen testaaja, joka on mahdollisimman riippumaton kehitystyöprosessista. Järjestelmätestaus sisältää myös ei-toiminnolliset ominaisuudet. Näitä ovat mm. kuormitustestit, luotettavuustestit, asennustestit, käytettävyystestit jne.

Jos järjestelmätestauksessa havaitaan virheitä, niin ne tulee korjata. Virheiden korjauksessa voi ilmaantua uusia virheitä. Virhettä korjatessa, useisiin moduuleihin voidaan joutua tekemään muutoksia. Kaikki virheitä ei välttämättä huomata. Muut moduulit tulisi testata ja järjestelmä testaus suorittaa uudestaan korjatuilla tiedoilla. Tälläista uudelleen testausta kutsutaan regressiotestaukseksi. Regressiotestaus voi tulla erittäin kalliiksi jos testausta ei saada automatisoitua.

Regressiotestausta tarkoittaa aikaisempien testien ajamista uudelleen. Kaikkia testejä ei ole mahdollista ajaa läpi, koska se vaatii paljon aikaa ja resursseja. Tarkoituksena on testata muutetun ohjelmakoodin vaikutusalue. Muut testit, jotka ovat esim. ajaneet onnistuneesti itsensä läpi, eivät vaikuta korjaustoimenpiteisiin. Regressiotestit voidaan kohdistaa ohjelmiston tiettyihin komponentteihin, jolloin suoritettavien testien määrän voi rajata. Automatisointi helpottaa regressiotestauksen suorittamista. Se säästää paljon aikaa ja resursseja.

Seuraava polku on hyväksymistestaus (engl. Acceptance testing). Hyväksymistestauksella tarkoitetaan, että järjestelmä pystyy suoriutumaan sille asetetuista vaatimuksista. Asiakas on määritellyt vaatimukset. Hyväksymistestaus suoritetaan ajallisesti viimeisenä vaiheena, osana järjestelmätestausta. Hyväksymistestaus voidaan suorittaa myös yksittäisenä vaiheena. (Software business competence 2014)

Verification & Validation tarkoittavat molemmat vahvistamista, mutta eri tavalla. Ne sijoittuvat hieman eri kohtiin V-mallissa ja vahvistavat hieman eri asioita, mutta ovat aika samankaltaisia vahvistuksia. Kun prosessi on vahvistettu, voidaan siirtyä seuraavaan kohtaan.

Validation (engl.) eli validointi eli vahvistaminen tarkoittaa, että tuote tai palvelu on sillä edellytyksellä toteuttanut/suorittanut tarkoituksen mukaisen käytön mitä asiakas haluaa. Validation kuvastaa sitä mitä vaatimuksia on määritelty tuotteelle tai palvelulle asiakkaan tarpeiden mukaan Yleisesti voidaan kysyä: rakennammeko tuotteen oikein? (Honolulu 2013.)

Verification (engl.) verifiointi eli vahvistaminen tarkoittaa, että tuote tai palvelu on asianmukaisesti rakennettu asiakkaalle. Verifiointi kuvastaa sitä mitä on vaatimuksia on määritelty tuotteelle. Yleisesti voidaan kysyä: rakennammeko tuotetta oikein? (Honolulu 2013.)

5.3 Agile-methods ja testaus

Agile:a (engl. Agile testing) kutsutaan ketteräksi menetelmäksi. Ketterä menetelmä on ohjelmistomenetelmä. Agile-testaus ei ole vain yksi menetelmä, vaan sisältää useita eri menetelmiä ja tapoja testata. Näitä erilaisia ketteriä menetelmiä on mm. ASD, XP, Scrum, Crystal, FDD yms. Koska samankaltaisia menetelmiä, pienillä eroilla on paljon, on päätetty käyttää yhteisnimitystä eli Agile-testausta. Agile-testaus on kehitelty, toistuva metodi, jossa vaatimukset kehittyvät yhteistyössä asiakkaan, itseohjautuvien tiimityöskentelyn kehityksessä, asiakkaiden tarpeiden mukaan ottaen.

Agile-testaus on lähtenyt kehittymään omaksi menetelmäksi V-mallin ja vesiputousmallin pohjalta. Loppukäyttäjät eivät olleet tyytyväisiä jäykähköihin prosessimalleihin, vaan halusivat vaikuttaa prosessin kulkuun ja palautteeseen jo aiemmissa vaiheissa. Tästä lähti eritasoiset Agile-menetelmät kehittymään.

Vuonna 2001 perustettiin ns. Agile Manifesto. Se tarkoittaa 12 kohdan-menetelmää, jossa on lueteltu Agile:n perustuvia periaatteita, jotka tulisi toteutua ketterässä menetelmässä. Agile manifeston ovat kehittäneet samat kehittäjät, jotka ovat luoneet aiemmat, aluksi pienemmät ohjelmistot mm ASD, XP, Scrum, Crystal, FDD yms.

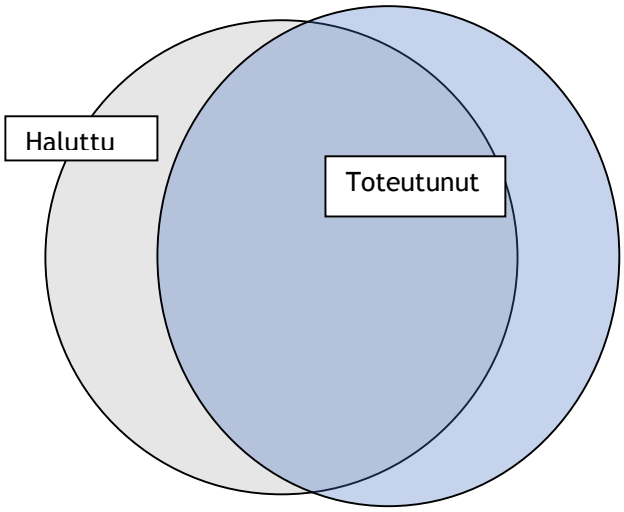
Ketterässä testauksessa pyritään ottamaan asiakkaan tarpeet huomioon mahdollisimman aikaisessa vaiheessa. Keskustelu projektin etenemisestä tiimin kanssa ja eri osa-alueiden kehittämistä eteenpäin tehdään joka päivä. Testaaminen ja palaute pyritään suorittamaan myös mahdollisimman aikaisessa vaiheessa, jotta vältyttäisiin huonolta tuotteelta.
Ketterässä menetelmässä ei ole perinteistä vaatimusmäärittely dokumentteja. Suppeammat dokumentit, josta löytyvät tarvittavat tiedot. ovat käytössä.
Ketterässä menetelmässä perinteiset roolijaot eivät päde. Tiimin jäsenillä on laajemmat mahdollisuudet tehdä projektissa erilaisia osa-alueita.

Erilaisia Agile menetelmiä on alettu ottamaan käytäntöön ympäri maailman. Se on nopeasti kasvava menetelmä, jonka monimuotoisuutta voidaan soveltaa eritasoisissa projekteissa. (Agile 2014.)

5.4 Tutkiva testaus

Exploratory testing (engl.) eli tutkivalla testauksella tarkoitetaan epämuodollista testaussuunnittelutekniikkaa. Testaaja aktiivisesti valvoo testejä samalla, kun testejä tehdään. Testaaja käyttää tietoa hyväkseen ja suunnittelee parempia testejä. (ISTQB:n sanasto 2014.)

Vaatimusmäärittelyt ohjaavat testaajaa tekemään ja saavuttamaan sisällön. Se miten testaaja testaa testitapauksia, selviää kokeilemalla erilaisia tekniikoita ja keskustelemalla muiden tiimin jäsenten kanssa. Tutkivassa testauksessa testaajan rooli korostuu, koska siihen tarvitaan monenlaista erikoisosaamista, eri aloilta. Tutkivaa testausta käytetään myös Agile-testauksen yhteydessä. Ne eroavat toisistaan hieman.



Kuva 8: Exploratory testing

Esimerkiksi testitapausten pohjalta yleensä testataan ”haluttu” (kuviossa vaalean liila), kun taas tutkiva testaus testaa ”toteutuneen” (kuviossa vaalean sininen). Kuviot eivät ole yhteneviä, koska käytännön ”toteutunut” ja teorian ”haluttu” osuus poikkeavat toisistaan. Yhteneviä kuvioita ei saavuteta, koska projekteissa on monta muuttujaa, jotka saattavat vaikuttaa lopputulokseen.

6 Testiautomaation periaatteet

Testiautomaation voi suorittaa monella eri tavalla. Se sisältää monta eri vaihetta ja eri toiminta tapaa. Tässä osiossa on esitelty osia, jotka ovat vaikuttaneet onnistuneen testiautomaation syntyyn. Osiot koostuvat teoria ja käytännön osuuksista.

Tässä osiossa on eroteltu prosesseja, koska ne vaikuttavat eri tavalla lopputulokseen. On tärkeää huomioida eri vaiheet, koska ne vaikuttavat kokonaisuudessa projektiin. Testiajosta muodostuu erilainen, jos prosessia käytetään esim. pelkkää manuaalista prosessia tai automaatiota.

Osiossa esitellään myös työvaiheet. Työvaiheet vaihtelevat eri projektien mukaan. Esitellyt työvaiheet tarvitaan, jotta voidaan tehdä testiajo.

6.1 Manuaalisen ja automaattisen testauksen ero

Manuaalisessa testauksessa toimintoja tehdään käsin eli manuaalisesti. Manuaalisessa testauksessa tarvitaan testaajaa suorittamaan testaus. Manuaalisessa testauksessa voidaan suorittaa testauksen osa-alueita automaatiolla. Kaikkia toimintoja ei yleensä kannata tehdä automaatiolla, vaan testaaja tarvitaan testaamaan tuote. Testaajalla on tärkeä rooli testiajoa suorittaessa. Testaaja havaitsee mahdolliset virheet ja osaa valvoa testiprosessia.

Testiautomaatiossa ei tarvita testaajaa suorittamaan manuaalisesti toimintoja. Automaatio hoitaa prosessin. Automaatio testauksesta tiettyjä testauksen osa-alueita automatisoidaan, joka nopeuttaa tiettyjen osa-alueiden läpivientiä. Automaatio testaukseen tarvitaan ohjelmiston kanssa yhteensopivia internet työkaluja, jotka sopivat käyttötarkoitukseen. Testityökaluja on paljon erilaisia, joista tarvitsee valita käyttötarkoitukseen sopiva työkalu.

6.2 Testiautomaation suunnittelu

Testiajoa rakennettaessa olisi hyvä konkreettisesti miettiä, mitä sisältöä halutaan testiautomaation suoritettavaksi. Ohjelma suorittaa annetut toiminnot, jotka testaaja syöttää manuaalisesti ohjelmaan (Selenium IDE). Ohjelma huomaa testattavassa sovelluksessa olevat virheet, mutta ei ymmärrä mitä kokonaisuudessaan halutaan. Testaajan tehtävänä on syöttää haluttu tieto automaatioon. Sisällön suunnitteluun vaikuttavat mm. projektin luonne, eri sidosryhmät, tavoitteet, aikataulut, kapasiteetti ja rahoitus. Mahdolliset ongelmat tulee karsia pois jo suunnitteluvaiheessa, jotta säästytään virheiltä, viiveeltä ja lisäkustannuksilta.

Suunnitteluvaihe sisältää monen eri asian huomioimista, yhteensovittamista ja aikataulutusta. Tässä vaiheessa rakennetaan testiautomaation ”runko”. Suunnitteluvaiheessa otetaan huomioon kaikki tarpeellinen tieto ja hyöty, joka hyödyttää projektia. Suunnitteluvaiheessa karsitaan kaikki ylimääräinen, mistä ei ole hyötyä projektissa, pois.

Testiautomaation suunnittelu vaihe on tärkeä vaihe. Testiautomaatio voi epäonnistua, jos siinä vaiheessa ei oteta kaikkia vaikuttavia tekijöitä huomioon. Suunnitteluvaiheessa tarvitaan paljon tietoa, kokonaisnäkemyksen hallintaa ja hyvää kommunikointi kykyä muiden eri sidosryhmien kanssa.

6.3 Testaajan rooli testiautomaatiossa ja testiajon työvaiheet

Testaajan tehtävänä on testata ohjelmisto ja varmistua ohjelmiston laadusta. Testaajat joutuvat olemaan tekemissä monien eri sidosryhmien kanssa projektissa. Projektin luonteen kannalta on hyvä hahmottaa kokonaisuus ja halutut tarpeet. Testaaja varmistaa, että syötetyt tiedot ovat oikeita ja johtavat haluttuun lopputulokseen. Testaaja tekee testiajon, havaitsevat virhekohdat, konsultoi muita sidosryhmiä, katsoo, että ohjelma suorittaa ja saavuttaa halutun toiston. Testaaja suorittaa myös raportoinnin.

Jokainen testiajo vaatii työvaiheet, jotta testiajo onnistuu. Työvaiheita voi olla erilaisia ja ne saattaa hieman erota toisistaan, projektin luonteesta riippuen. Projektin kannalta on tärkeää huomioida työvaiheiden tärkeysjärjestys. Automaatiotestiajo etenee suunnitellusti, koska muuten se ei voi suorittaa haluttuja toimintoja. Onnistuneen testiajon työvaiheita ovat mm. suunnittelu, konsultointi eri sidosryhmien kanssa, tietojen syöttäminen, seuranta, virheiden havaitseminen ja korjaus, tavoitteiden saavutus liiketoiminta mallin mukaisesti.

Testiajon työvaiheet määräytyy projektin luonteen mukaan. Suunnitteluvaiheessa mukaan määrätyt prosessit vaikuttavat työvaiheisiin ja testiajon pituuteen. Testaaja määrittelee ja suunnittelee työvaiheiden sisällön.

6.3.1 Testiajon aloitus ja kulku

Testiajo aloitetaan, kun on varmistettu siitä, mitä toimintoja halutaan suoritettavaksi automaatiolla. Tiedot syötetään kenttiin, lisätään halutut toiminnot ja tehdään koeversio automaatiosta. Jos tarvittavat tiedot ja toiminnot vastaavat haluttuja toimintoja, testaaja hyväksyy testiajon käytettäväksi tuotannossa.

Testiajo alkaa (tässä testiajossa käytetään Selenium IDE-työkalua) vasemmalta yläriviltä, editor ikkunassa. Command kohdassa näkyy ensimmäinen suoritettava toiminto. Jos ei tule

virheilmoitusta, niin automaatio siirtyy seuraavaan kohtaan, joka on manuaalisesti kirjoitettu tai nauhoitettu. Automaatio etenee näin niin kauan, kunnes nauhoitetut toiminnot loppuvat tai tulee error eli virhetoiminto, jolloin automaatio pysähtyy.

Testiajo alkaa, kun painetaan punaista Record-nappulaa. Sen jälkeen aletaan manuaalisesti syöttämään tietoja. Tietoja syötetään niin paljon, kun on tarvetta. Selenium IDE nauhoittaa kokoajan kaikki toiminnot. Kun kaikki halutut tiedot on syötetty, painetaan Record-toimintoa, jotta nauhoitus loppuu. Sen jälkeen aloitetaan automaattinen toisto. Automaattista toistoa tehdään niin paljon, kuin on tarvetta.

6.3.2 Koe testiautomaatio

Testiautomaatiota tehdessä, tehdään ensin ns. koeversio, johon syötetään tarvittavat, halutut tiedot. Testaaja testaa koeversiolla automaation ja löytää mahdolliset ongelma- ja virhekohdat. Tässä vaiheessa prosessia on korjattavissa mahdolliset ongelmat. Mahdolliset lisäykset voidaan suorittaa tässä kohdassa. Mahdollisten korjausten jälkeen koeversio automaatiosta täytyy uudestaan koeajaa läpi. Jos huomautettavaa vielä löytyy ne täytyy korjata ja tehdä uusi koeversio. Koeversioita automaatiosta tehdään niin kauan, kuin haluttu automaatio on toimiva ja odotusten mukainen.

Koeversiota ei koskaan saa käyttää valmiina viimeisenä versiona. Koeversiossa olevat tiedot saattavat olla virheellisiä tai sieltä saattaa puuttua tietoa, joka vaikuttaa koeajoon ja lopputulokseen.

Koeversiot kannattaa numeroida. Koeversioita saatetaan joutua tekemään useita kappaleita. Vertailun helpottamiseksi, numerointi kannattaa. Koeversioita saatetaan verrata keskenään tai muuten etsiä tietoa tai virheitä eri koeversioista. Numeroinnin avulla haluttu versio löytyy nopeammin.

Testaaja valmistelee koeversiosta tehtävän lopullisen testiautomaatioversion, jolla aletaan tekemään testiautomaatiota. Testaaja on tarkistanut sisällön ja karsinut pois mahdolliset virheet. Testiajo voidaan aloittaa tarkistetulla versiolla.

6.3.3 Testiajon kesto ja pituus

Testiajon kestoon ja pituuteen vaikuttavat syötetyt tiedot. Testaaja määrittelee tarvittavan sisällön. Se määrää testiajon keston ja pituuden. Testiajoja voi olla erilaisia ja eripituisia riippuen halutusta tarpeesta. Minimitestiajo voi olla yksi rivi. Maksimi testiajo voi olla useita riviä. Virheet hidastuttavat testiajoa.

Projektin kannalta on järkevää suhteuttaa testiautomaation kesto. Testiajon kesto vaikuttaa prosessin etenemiseen. Se ei saa kestää liian kauan vaan sen on suoriuduttava määräajassa valmiiksi. Määräajat määrittelee projekti.

Jos testiajo kestää kauan, se täytyy tarkastaa miksi se kestää. Hidastuksen syitä voivat olla mm. virheet järjestelmässä, oikeinkirjoitusvirheet ja ennalta odottamattomat asiat. Testiajon pituus otetaan huomioon projektissa. Se ei saa hidastuttaa projektia.

6.3.4 Testiautomaation lopputulos

Testiautomaatiolla pyritään projektin kannalta haluttuun lopputulokseen. Testiautomaation lopputulos riippuu automaation syötettyjen tietojen oikeellisuudesta ja toimivuudesta. Oikein syötetyt tiedot, testiajon virheetön kulku, tuottavat halutun testiautomaatio prosessin. Prosessi on silloin onnistunut.

Jos testiautomaatiossa esiintyy virheitä, ne täytyy korjata. Testaaja analysoi virheen laadun, määrän, vaikuttavuuden ja vahingollisuuden. Testaaja raportoi virheet bugiraporttiin. Virheen laadusta riippuen on tehtävä uusi testiajo, muutetuilla tiedoilla. Testiajo alkaa alusta, kun virheet on saatu korjattua tai minimoitua.

Joskus testiajossa saattaa ilmetä vakavanlaatuisia virheitä. Tällöin testiajo täytyy keskeyttää. Testiajo täytyy uusia uudistetuilla tiedoilla.

6.3.5 Loppuraportointi

Testaaja tekee loppuraportoinnin testiajosta ja mahdollisista virheistä. Testaaja kirjaa prosessin eri vaiheita raporttiin. Raportin huolellinen täyttäminen on tärkeä osa testiautomaatio prosessia. Raportointi on tärkeä osa prosessia, koska raporteista näkee prosessin kulun, mahdolliset virhekohdat ja palautteen.

On tärkeää täyttää raportti huolellisesti. Raportista ilmenee tarvittava tieto. Jos raporttia ei täytä huolellisesti, niin tietoa jää saavuttamatta. Tämä voi aiheuttaa ongelmia projektissa.

Loppuraportointi tallennetaan järjestelmään. Loppuraportoinnin voidaan tarvittaessa rajata käyttäjäryhmä, jotka näkevät raportin. Riippuu projektin luonteesta ja siitä, että onko tiedoissa mitään salassa pidettävää vai ei. Tarvittaessa loppuraportointi lähetetään organisaatiossa eteenpäin.

6.3.6 Hyöty ja käyttötarkoitus

Selenium IDE on yksinkertainen ja helposti käyttöönotettava työkalu. Sen käyttöönotto ei vaadi erikoisosaamista. Selenium IDE nopeuttaa testiajossa prosessissa. Selenium IDE:n hyöty on, että sillä voidaan nauhoittaa ja kääntää skriptiä useille eri kielille. Automaatiota käytettäessä virheiden määrä minimoidaan, koska toiminto on automaattinen. Automaattinen tarkoittaa valmiiksi ohjelmoitua toimintoa, joka toistaa toiminnot samalla kaavalla joka kerta kun testiajo suoritetaan. Testaajan tehtävänä on tarkastaa ohjelmoidut toiminnot, niin että ne vastaavat tarkoitusperää.

Selenium IDE:n käyttötarkoitus on helpottaa ja nopeuttaa prosessia, kun tehdään manuaalista testiajoa. Selenium IDE on automaatiotyökalu, joka tekee tehdyn toiminnon. Ohjeita on erilaisia ja ne löytyvät esimerkiksi Selenium HQ kotisivuilta. Käytännön tukea saa Seleniumin kotisivustolta, keskustelupalstoilta ja sosiaalisesta mediasta. Selenium:n kotisivuilta löytyy useita eri sivustoja, mistä voi hakea apua virtuaalisesti. Käyttäjien keskuudessa keskustelu foorumit ovat nopea tapa esittää kysymyksiä ja saada vastauksia.

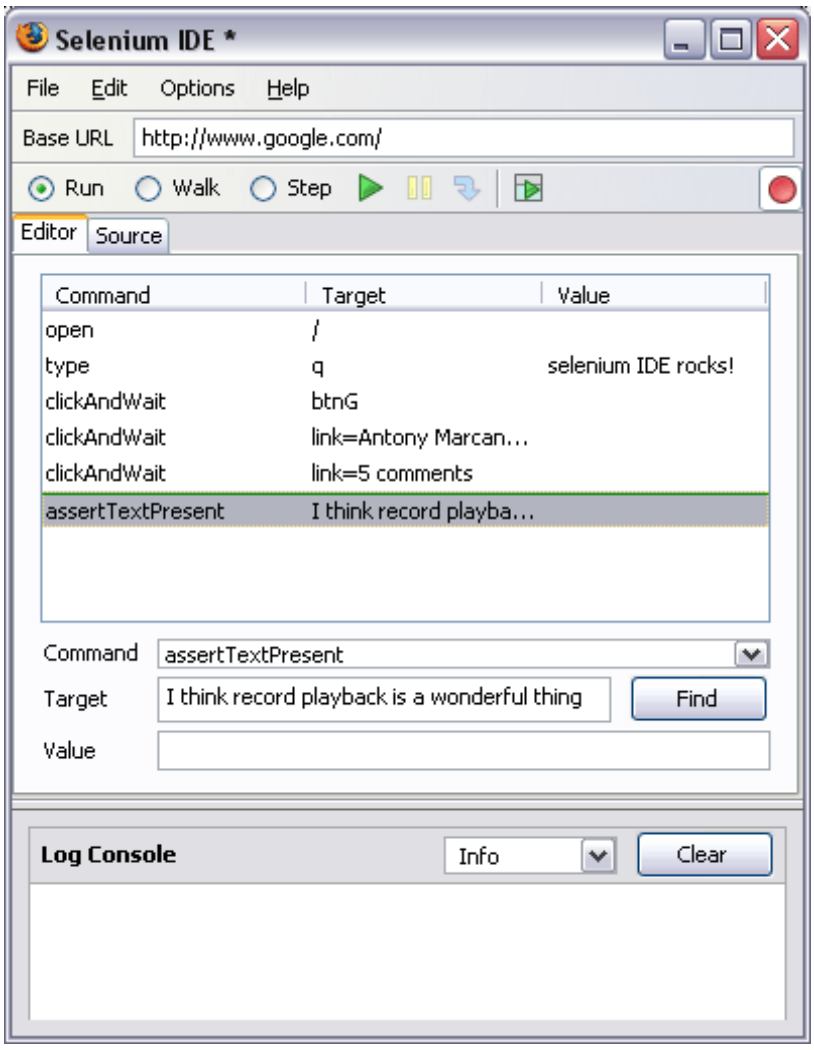
Selenium:in hyöty on myös nopeasti päivittyvät uudet versiot tuotteesta. Ne on toteutettu järkevästi, niin ettei koko tuotetta uudisteta kokonaan kerralla vaan osia uudistetaan tarpeen mukaan. Ulkoasua ja käytettävyyttä uudistetaan samannäköiseksi, mitä vanha versio on ollut. Niihin osiin, jotka ei tarvitse päivitystä, ei uusita. Ne uusitaan myöhemmin, jos tarvetta on. Hyötynä on, että käyttäjän ei tarvitse joka kerta opetella uusia komentoja tai toimintoja.

7 Selenium IDE-työkalu

Selenium IDE työkalu on eräänlainen nauhoitin, joka nauhoittaa skriptiä. Selenium IDE on automaatio, joka toistaa tehdyn toiminnon. Selenium IDE:llä voi nauhoittaa ja editoida skrptiä. Sillä voi myös tehdä debug-testejä Selenium IDE on Firefox-selaimen lisäosa. Selenium IDE helpottaa manuaalista testaamista, koska jokaista toimintoa ei tarvitse tehdä alusta loppuun asti manuaalisesti vaan automaatio tekee prosessin. Automaatiotyökaluja käytetään manuaalisessa testaamisessa apuvälineinä, koska se helpottaa ja nopeuttaa osaa toiminnoista. Automaatiot sijoitetaan testiajossa omiin kohtiinsa, mikä tukee ja nopeuttaa testiajon tekemistä.

Selenium IDE:ä voi käyttää kaikki, ketkä ovat kiinnostuneita testiautomaation tekemisestä tällä työkalulla. Selenium HQ tuoteperheeseen kuuluu myös muita työkaluja ja sovelluksia. Ne tukevat toinen toisiaan ja ovat yhteenkuuluvia. Selenium:a ja sen eri osia voi myös käyttää erikseen. Selenium HQ on rekisteröinyt kaikki sovelluksensa ja työkalunsa.

Selenium IDE täytyy ladata koneelle ennen käyttöä. Sen voi ladata (SELENIUM HQ) sivulta. Selenium IDE sijaitsee Firefox- selaimen komento (työkalu) riviltä. Selenium IDE on työkalu, joka perustuu avoimeen lähdekoodiin. Selenium IDE on ilmainen työkalu Selenium IDE:ä ei voi käyttää muun kuin Firefox-selaimen kanssa.



Kuva 9: Selenium IDE työkalu (Selenium IDE)

Tietokoneella täytyy olla selain, joka on Mozilla Firefox. Ensimmäiseksi ladataan Selenium kotisivu osoitteesta http://www.seleniumhq.org/, Sen jälkeen oikealla, keskellä on download-toiminto. Sitä klikataan. Selenium IDE otsikon alta valitaan ladattavaksi uusin versio. Sitä klikataan ja Selenium IDE latautuu koneelle. Selenium IDE haetaan työkalu riviltä, tools (engl.) kohdasta. Sieltä valitaan Selenium IDE. Näytölle ilmestyy automaatio työkalu. Selenium IDE on valmis käytettäväksi. (Polkuna: http://www.seleniumhq.org/ -> download -> latest version x.x .x -> set up. Oikopolkuna asennuksen jälkeen Ctrl + alt + s.)

7.1 Toiminnot

Selenium IDE:llä voi mm. nauhoittaa, toistaa ja editoida skriptejä. Sillä voi myös etsiä järjestelmästä virheitä. Selenium IDE:een voi kerätä erilaisia nauhoituksia arkistoon ja tallentaa ne. Selenium IDE:llä voi tehdä myös muita toimintoja. Selenium IDE tukee monenlaisia toimintoja eri koodikielillä.

Selenium Idellä voi myös nauhoittaa ja kääntää toiselle kielelle skrptiä.
Tunnetuimmat kielet, mitä Selenium IDE kääntää suoraan ovat: Python, Java, Ruby ja, C#.
Muillakin kielillä kääntäminen onnistuu, mutta vaatii lisää toimintaa.

Ennen kun aletaan nauhoittamaan testitapauksia, on hyvä tutustua ohjelmiston, tehtäväpalkkiin ja komentoriviin. Tehtäväpalkista riviltä löytyy toiminnot, joita tarvitaan, kun suoritetaan prosessia. Komentoriville on koottu tärkeimmät toiminnot, mitä tarvitaan, kun suoritetaan prosessia. Ne on laitettu eri väreillä ja kuvioilla, jota ne erottuisivat ja olisivat helpompi erottaa muista toiminnoista. Muut toiminnot löytyvät tehtäväpalkista ja alasvetovalikoista. Poikkeuksen tekee Record-toiminto, joka on punaisen pallon muotoinen symboli, joka sijaitsee oikealla. Se tarkoittaa nauhoitusta.

Testiautomaatio näkyy Editor ikkunassa suoritettuna toimintona. Automaation toimintaa voi seurata koko testiajon Editor-ikkunasta. Testiautomaatio etenee vasemmalta alaspäin. Testiajossa näkyy skripti ja skriptin päällä joko vihreä tai punainen palkki. Testiajon alkaessa, vihreällä kuvatut toiminnot Editor-ikkunassa, tarkoittavat hyväksyttyä toimintoa. Punaisella kuvatut toiminnot Editor-ikkunassa tarkoittavat virhettä, joka ei mene läpi testiajossa. Testiajon aikana tulee siis joko vihreä tai punainen palkki, skriptin päälle. Eriväriset palkin ovat laitettu helpottamaan automaation kulkua ja havaitsemaan paremmin virheet. Toiminnot täytyy aluksi syöttää käsin, jotta automaatio toimisi halutulla tavalla ja tuotaisi haluttua hyötyä. Sen jälkeen voidaan automatisoida toiminnot ja laittaa testiajo toimimaan automaattisesti.

7.2 Yleisimmät toiminnot

Työkaluriviltä, joka sijaitsee ylimmällä rivillä (katso kuva 8), heti Selenium IDE – tunnuksen alla, löytyy erilaisia toimintoja, jolla voi muokata testiajoa. Niihin kannattaa tutustua, ennen kuin alkaa suorittamaan testiajoa. Työkaluriviltä löytyy tarvittavat tiedot ja toiminnot.

Table/Editor-ikkuna näyttää kaikki toiminnot. Se täytyy klikata päälle, jotta voi alkaa tekemään testiajoja. Table/Editor-ikkuna tulee perusasetuksena näkyviin ensimmäisenä.

Source- ikkunasta näkyy skriptit ja koodiosoitteet. Scrollbar voi seurata skriptiä ja tutkia toimintoa. Niitä voi myös kopioida

Record toiminnon tarkoituksena on suorittaa nauhoitus. Record toiminto nauhoittaa skriptiä. Record-toiminto nauhoittaa kaiken toiminnon, kun se on päällä, Myös välilyönnit ja klikkaukset.
Record-toiminto löytyy oikeasta yläkulmasta ja on punaisen pallon muotoinen. Sitä klikatessa, alkaa nauhoitus. Record-toiminto nauhoittaa niin kauan, kun toiminto pidetään aktiivisena. Nauhoitus pitää sulkea samasta napista. Nauhoitettu skripti näkyy toimintona Test Cases-ikkunassa vasemmalla. Nauhoituksia voi suorittaa useita ja niitä voi kerätä, uudelleen nimetä ja arkistoida.

Test case-ikkunaan tulee suoritettu testiajo. Sinne kerääntyy automaattisesti kaikki suoritetut testiajot. Ne nimetään erikseen ja toistetaan tarvittaessa. File- toiminnosta, joka sijaitsee työkalurivillä, Test Case- ikkunan yläpuolella, voi tallentaa ja tehdä muita asetuksia testiajolle. Test case-ikkunasta voi etsiä tarvittavan testiajon ja toistaa sen tai tarvittaessa toistaa kaikki.

Play resent test case- toiminto suorittaa nauhoitetun toiminnon. Editor-ikkunassa näkyy nauhoitettu toiminto, kun painetaan Play resent test case-toimintoa. Editor-ikkuna näyttää testiajoa. Play resent test case-toiminto täytyy pysäyttää Pause – toiminnosta. Muuten se pyörittää automaatiota koko ajan. Editor-ikkuna näyttää kaikki suoritetut toiminnot.

Play entire test suite- toiminto suorittaa kaikki nauhoitetut testiajot, jotka ovat tallennettu Test Case-toiminnon alle. Ne toistetaan automaattisesti järjestyksestä. Punainen palkki näyttää missä kohtaan ollaan menossa.

Pause/Resume toiminnosta toiminto pysähtyy. Pause/Resume toiminnosta toiminto myös jatkuu, kun kyseessä on sama testitapaus. Toiminto jatkuu samasta kohdasta, mihin se jäi. Testi automaatio lähtee jatkumaan eteenpäin kohti seuraavaa toimintoa.

Editor-ikkunaan tulee käynnissä oleva testiajo. Siinä näkyy testiajon nauhoitettu kulku ja komennot. Editor-ikkunasta voi seurata mitä testiajossa tapahtuu, missä kohtaa ollaan menossa ja missä kohtaan tulee mahdolliset virheet. Virhe ilmoitus näkyy editor ikkunassa eri värillä virheenä Virheilmoitus tulee tarkemmin näkyviin Log Console – ikkunaan.

Base Url-toiminto kenttään laitetaan haluttu toiminto, mikä halutaan suoritettavaksi. Tämä voi olla esimerkiksi www.google.com sivusto mitä nauhoitetaan tai joku muu skripti, mikä halutaan nauhoittaa. Toiminto näkyy koko automaation ajan kentässä.

Log Console–ikkunaan tulee koodirivistöt. Log Console ikkunassa näkyy suoritetut onnistuneet skriptit. Ne näkyvät mustalla värillä. Log Console – ikkunaan tulee ilmoitus error eli virhe koodista. Se näyttää myös virheellisen skriptin, joka ei mene testiautomaatiossa läpi. Error-koodi näkyy ikkunassa punaisella, mikä tarkoittaa virhettä.

Fast-Slow toiminnolla voidaan nopeuttaa tai hidastaa automaation suoritusta. Se täytyy asettaa manuaalisesti, haluttuun nopeuteen. Asetusta voidaan muuttaa manuaalisesti eri testitapausten takia.

Commond-toiminnossa näkyy valmiiksi automatisoidut toiminnot. Alasvetovalikosta tulee lisää toimintoja. Command toiminnolla voidaan muuttaa automatisoidun toiminnon kulkua, lisäämällä komentoja tai poistamalla niitä. Command toiminto näyttää mikä toiminto tulee tapahtumaan.

Target-toiminnosta seurataan koodikieltä. Target toimintoon tulee skripti, joka kertoo missä kohtaan ollaan menossa. Target toiminnosta näkyy keskeinen skripti. Target-toiminnosta voidaan mm. kääntää toiselle kielelle toimintoja, jos ne eivät jostain syystä toimi testiajoa suoritettaessa. Joskus testiajo ei toimi kyseisellä kielellä ja tarvitaan samankaltaista toimintoa suorittamaan testiajo. Target toiminnosta löytyy useita vaihtoehtoja. Target-toimintoon voidaan myös lisätä manuaalisesti komentoja, jotka testiajo suorittaa, kun painetaan Play resent test case-toimintoa. Kohdistin pitää siirtää oikeaan kohtaan ennen toiminnon suorittamista.

Value toiminnosta seurataan koodikieltä. Value toimintoon tulee arvoja ja erikoismerkkejä esimerkiksi € yms. Value toimintoon voidaan lisätä manuaalisesti komentoja.

Selenium IDE sisältää myös muita toimintoja, joilla voi hienosäätää skriptiä. Ne löytyvät komentoriviltä ja niitä voi lisätä tai poistaa testiajossa, käyttötarkoituksen mukaan. Jokainen testitapaus räätälöidään toimeksiantajan käyttötarkoitusta varten. Siihen sisältyvät toiminnot täytyy suhteuttaa ja räätälöidä tavoitteiden saavuttamiseksi.

Testiajon toimintojen tekeminen voidaan tehdä eri tavoin. Yleisin tapa on kirjoittaa Base URL kohtaan haluttu sivusto tai skripti. Sen jälkeen valitaan haluttu toisto ja suoritetaan haluttu toisto. Tämä voi olla lyhyt tai pitkä toiminto. Se voi sisältää useita eri toimintoja ja suorituksia. Table-ikkunasta näkyy kaikki halutut toiminnot. Test Case-ikkunassa näkyy tallennetut testitoiminnot.

Toimintoja voidaan syöttää manuaalisesti muutamiin ikkunoihin, mutta ei kaikkiin. Jossain kohdissa on manuaalisen syöttämisen sijaan laitettu Scroll bar vaihtoehto. Klikatessa Scroll bar vaihtoehtoa (nuoli alaspäin-symboli), avautuu valikko, josta voi valita eri vaihtoehtoja. Ne ovat jo valmiiksi automatisoituja toimintoja, jossa on toimiva komento. Ne suorittavat annetun komennon. Niitä ei voi muuttaa.

Jos halutaan vain nauhoittaa sivustoja, niin silloin laitetaan päälle Record toiminto, joka nauhoittaa kyseisen sivuston skriptin. Sama toistetaan, niin kauan kuin haluttuja sivustoja on tarvittava määrä. Nauhoittaminen lopetetaan painamalla Record-näppäintä.

7.3 Onnistunut ja virheellinen skripti

Testiajo joko onnistuu tai ei. Toimintojen syöttö on tärkeä osa onnistuneen testiautomaation tulosta. Testaaja valitsee toimivat toiminnot testiajoon. Testaaja syöttää toiminnot automaatiotyökaluun. Testaaja seuraa koe automaation kulkua ja tuloksia. Jos virheitä tai muuta korjattavaa löytyy, niin automaatio työkalu ilmoittaa mahdolliset virheet ja pysähtyy virhekohtaan. Testaaja huomio virheet ja mahdollisuuksien mukaan korjaa ne. Sen jälkeen suoritetaan uusi koe testiautomaatio.

Koe testiautomaatiota tehdään niin kauan, kunnes testiautomaatio menee läpi. Yleensä onnistuneisiin ajoihin ei kiinnitetä huomiota lainkaan, koska ne ovat vihreitä. Vihreä väri tarkoittaa onnistunutta tietäjiä. Testaaja hyväksyy ne.

Opinnäytetyössä on tehnyt muutama koetestiautomaatio ajo. Ne liittyvät testausalaan ja opiskeluun Selenium IDE:stä. Opinnäytetyössä on tutustuttu Selenium IDE:n nauhoitus ominaisuuksiin ja yleiseen käytettävyyteen. Opinnäytetyössä on haluttu tehdä onnistunut skripti ja ei-onnistunut skripti. Opinnäytetyössä on pyritty näyttämään niiden erot, jotta ne pystytään erottamaan toisistaan ja huomata mahdolliset virheet.

7.3.1 Skripti onnistunut

Onnistuneessa skriptistä ei tule virheilmoitusta. Se menee suoraan läpi testiajossa. Onnistunut skripti näkyy suoritettuna toimintona. Automaatio lähtee toimimaan ylhäältä alaspäin. Testiajo lähtee vasemmalta etenemään alaspäin. Automaatio suorittaa yhden skriptin kerrallaan. Sen jälkeen automaatio suorittaa seuraavan toiminnon. Automaatio etenee näin testiajon loppuun asti, jos ei tule virhettä.

testi_1		
open	/search?q=suomi-englanti-suomi+sanakirja&ie=utf-8&oe=utf-8&rls=org.mozilla:fi:official&client=firefox-a&gws_rd=cr&ei=NWvqUtewKKKv4QTv44DYCQ	
clickAndWait	//ol[@id='rso']/li/div/div/h3/a/em[4]	
click	link=en-fi	
type	name=q	resume
clickAndWait	css=input.button	
click	css=input.button	

Kuva 10: Skripti onnistunut

Skriptiä voi seurata koko testiajon. Se näyttää toiminnot reaaliaikaisena. Testiajo etenee järjestelmällisesti. Näytössä esiintyy palkki, missä kohtaa ollaan menossa. Palkki vierii eteenpäin, kun toiminto on suoritettu.

Kohta, jossa palkki vierii, on väriltään neutraali. Testiajossa se näkyy haalean vihreänä. Haalean vihreä tarkoittaa onnistunutta toimintoa. Punainen palkki tarkoittaa virheellistä informaatiota.

7.3.2 Skripti virheellinen

Virheellinen skripti eli ei onnistunut toiminto, ei toimi. Sitä ei voi hyödyntää mitenkään. Se täytyy korjata. Epäonnistuneesta skriptistä tulee virheilmoitus. Se tulee komentoriville ja on merkitty eri värillä, jotta se huomataan. Testiajo pysähtyy siihen kohtaan, josta virheilmoitus tulee. Error- ilmoitus näkyy punaisella värillä.

```
[info] Executing: |open | / | |
[info] Executing: |type | id=masthead-search-term | gnarls barkley crazy |
[info] Executing: |clickAndWait | //ol[@id='search-results']/li/div[2]/h3/a/span | |
[error] Element //ol[@id='search-results']/li/div[2]/h3/a/span not found
```

Kuva 11: Skripti virheellinen

Tässä malliesimerkissä testiajo on pysähtynyt viimeiselle riville. Neljännen rivin ensimmäiseen informaatio kenttään on tullut error-ilmoitus. Automaatio ei etene seuraavaan kohtaan, koska error ilmoitus on tullut. Tässä kohtaa testaajan tulee huomata virhe ja korjata se.

Palkki on kokonaan punaisella värillä, mikä tarkoittaa virhettä. Log-kentässä näkyy error-ilmoitus punaisella. Test case-ikkunaan tulee Failure-ilmoitus. Failure ilmoituksesta näkee virhe rivistöjen määrän. Virheellinen tieto pitää korjata. Korjaustapoja voi olla useita.

7.3.3 Testiajo Selenium IDE:llä

Jotta testiajosta pystytään rakentamaan toimiva kokonaisuus, on syytä tutustua testauksen teoriaan. Teoria tukee käytännön työtä. Opinnäytetyössä on käytetty eri teoria malleja eri testiajossa. Esimerkiksi (katso kuvio 9) on suoritettu, onnistuneella skriptillä, testiajo. Siihen on tarvittu mm. suunnittelua (mitä toimintoja halutaan nauhoittaa), kokonaisnäkemystä (mihin testiajolla pyritään) ja toimintojen syöttämistä työkaluun.

Suunnitteluvaiheessa käydään läpi, mitä toimintoja halutaan tai tarvitaan testiajon suorittamiseen. Suunnitteluvaiheessa tarvitaan tässä testiajossa mm. ekvivalenssiluokkia tai raja-arvo-taulukoita (katso kohdat 4.1.4. ja 4.1.5). Niitä käytetään, jotta pystytään rajaamaan hakukriteerejä ja saavuttamaan onnistunut testiajo tietyillä ehdoilla. Tässä testiajossa ehdot täyttyivät.

Kokonaisnäkemyksenä tässä testiajossa (katso kuva 9) on konkreettisesti haluttu mm. todentaa miltä näyttää onnistunut skripti. Tarkoituksena on myös näyttää mikä ero on onnistuneella (katso kuva9) ja ei onnistuneella skriptillä (katso kuva10). Kuvat näyttävät erilaisilta, koska tulokset eroavat toisistaan. Työkalun käyttöominaisuudet ja visuaalinen näkemys hahmottuvat käyttäjälle konkreettisemmin tehtyjen toimintojen kautta. Tekemällä erilaisia testiajoja käyttäjä oppii työkalun käytettävyyden .

Oikein syötetyt toiminnot tuottavat onnistuneen testiajon. Sitä ei tarvitse korjata, koska se on suunniteltu oikein, toteutettu oikein ja toiminnot on syötetty oikein, niin ettei niissä ole kirjoitusvirheitä yms. virheitä, jotka vaikeuttavat skriptin läpimenoa.

Tässä testiajossa Selenium IDE havaittiin hyväksi testiautomaatio työkaluksi. Testausvälinettä tutkittiin erilaisilla ominaisuuksilla (katso kohdat 6 ja 7) ja se osoittautui hyväksi työkaluksi. Lisäksi tarvittiin teoria osuutta(mm. katso kohdat 1.3, 4.1.4, 4.1.5, 4.1.6, 4.1.7 ja 5.2. Muista teoria-osuuksista tarvittiin osia testiajon testaamisen varten.) testiajon suunnittelutyöhön.

7.3.4 Arviointi

Opinnäytetyössä testattiin Selenium IDE automaation työkalua testiajossa. Testiajoja tehtiin useita ja erilaisia. Kuvissa 9 ja 10 on kahden eri testiajon tulokset.
Ne on tehty Selenium IDE:llä. Testiajot tehtiin, jotta käyttäjille hahmottuisi kuva, miltä näyttää onnistunut ja ei onnistunut testiajo automaatio työkalulla.

Havaitsin testiajoa tehdessä mm- Selenium IDE:n käyttäjäystävällisyyden. Sitä on helppo käyttää, vaikka ei koskaan olisi tehnyt testiajoa aikaisemmin. Näyttö on selkeä ja toiminnot sijoitettu järkevästi. Käyttäjän on helppoa erottaa toimintoja, koska toiminnot on laitettu symboleilla ja eri väreillä työkaluun. Testiajoa tehtäessä, liikkuminen eri toimintojen välillä on helppoa ja mutkatonta. Työkalun käyttö ei vaadi erikoisosaamista.

Selenium:in kotisivuilta löytyy paljon tietoa ja keskustelupalstoja, mistä voi käydä hakemassa apua tai muuten tutustua testiautomaation ja työkaluun. Keskustelupalstat toimivat nopeasti, kattavasti ja selkeästi. Kotisivut on hyvin rakennettu, ne sisältävät paljon tietoa, käytännön ohjeita ja ovat helposti käytettävissä.

Hankaluuksia oli muuttaa error-koodi toimivaksi (katso kuva 10). Ensiksi oli vaikea hahmottaa ongelman kohtaa, että missä kohtaa skriptiä on tullut tehtyä virhe. Automaatio työkalu näyttää virheen kohdan, mutta skripti täytyy avata ja tutkia tarkemmin, että missä on vika. Selenium:in kotisivuilta löytyvä ongelmanratkaisu osio ja keskustelupalsta auttoivat hahmottamaan vian syyn. Error-koodi saatiin muutettua onnistuneeksi kääntämällä ajettu skripti toiseen muotoon kääntäjän avulla. Ongelmana oli, että testityökalu ei tunnistanut sisään ajettua skriptiä, josta tuli error-ilmoitus vaan tarvitsi avuksi kääntäjää, joka käänsi skriptin toiselle kielelle. Työkalun kääntäjä kääntää kielen toiselle kielelle, jonka työkalu tunnistaa, jos se on työkalun kanssa yhteensopiva kieli (katso kohta 7.1.). Kielen kääntämisen jälkeen testiautomaatio jatkaa samasta kohdasta mihin se oli pysähtynyt. Jos testiautomaatio ei lähde eteenpäin, kääntämisestä huolimatta, niin testiautomaatio täytyy aloittaa alusta ja käydä läpi, että missä on vika.

Laajempaa testiajoa tehtäessä olisi tarvittu enemmän teoriapohjaista tietoa (katso kohdat 4.1.1.-4.1.3., 5.3. ja 5.4) ja käytännön testausta. Opinnäytetyössä käydyt asiat on hallittava, jotta pystyy tekemään testiautomaatio kokonaisuuksia. Teorian hallinta on osattava, jotta pystyy suunnittelemaan testiajon sisältöä, korjaamaan virheitä ja ymmärtämään testauksen merkityksen projektille. Tässä testiajossa tarvittava teoria osuus on käyty teoria osuudessa läpi.

8 Testaus osana muita toimintoja

Testaus on yksi tärkeä osa liiketoiminnassa. Testausalaan liittyy monia eri osa-alueita, jotka vaikuttavat testauksen toteutumiseen. Eri osa-alueiden on toimittava keskenään, jotta projekti onnistuisi ja tuotettaisiin voittoa.

Kommunikointi ja tiedon jakaminen projektissa auttavat pääsemään tavoitteeseen. Testaaja joutuu olemaan tekemisissä erilaisten ihmisten kanssa. Sujuva kommunikointi on osa toimivaa kokonaisuutta.

Testauksen merkitystä ei aina ymmärretä organisaatiossa. Testaus on kuitenkin tärkeä vaihe ohjelmiston kehittämisessä. Ilman testauksen hyötyä, yritys voi tuottaa tappiota.

8.1 Aikataulut

Projektilla on aikataulu. Se määrittelee milloin tuotteen on oltava valmis. Projektin johto määrittelee kokonaisuuden, johon aikataulut sidotaan. Aikatauluihin vaikuttaa mm. projektin laajuus, eri sidosryhmät, prosessien onnistumiset, komponenttien ja muiden rakenteiden toimivuus yms.

Jokaisella prosessilla on omat aikataulunsa, joka on sidoksissa isompaan kokonaisuuteen. Jopa pienet viiveet jollain tasolla saattavat myöhästyttää projektin valmistumista. Mitä kauemmin prosesseissa kestää, sitä kauemmin tuote viivästyy markkinoilta. Se on erittäin kallista projektille.

Projektin aikataulussa pysyminen on erittäin tärkeää. Eri osa-alueiden myöhästyminen saattaa vaikuttaa muiden osa-alueiden myöhästymiseen. Tästä seuraa viivettä, joka saattaa vaikuttaa projektin kehittymiseen. Pahimmassa tapauksessa projekti saattaa myöhästyä, mikä aiheuttaa tappiota yritykselle.

8.2 Eri sidosryhmät ja mahdolliset ongelmat

Testiautomaatiota suunnitellessa on mietittävä eri sidosryhmien vaikutusta projektin kokonaisuuteen. Eri sidosryhmiä voivat olla esimerkiksi projektin johto, markkinointi, talous-osasto, koodaajat, insinöörit, asiakas, testaajat, rahoittajat ja viranomaiset. Sidosryhmiin voi kuulua paljon erilaisia tahoja, ammattikuntia ja erikoisalan tuntijoita. Vuorovaikutus ja joustava kommunikointi projektin onnistumisen kannalta ovat tärkeitä tekijöitä. Jokaisessa projektissa on oma resurssin tarve, joka määrätään liiketoimintamallin sallimissa rajoissa. Projektiin otetaan vain tarvittavat sidosryhmät.

Projekteissa eteen voi tulla monenlaisia ongelmia Ongelmina voidaan pitää monia asioita. Tällaisia asioita ovat mm. kiire, resurssipula, kyvyttömyys hoitaa asioita, tiedonkulun ongelmat, virheet ohjelmistossa, kapasiteettiongelmat, luonnonkatastrofit, tuotannolliset ja taloudelliset asiat yms. projektiin vaikuttavat asiat.
Ongelmat saattavat hidastaa tai pahemmassa tapauksessa estää projektin läpiviennin. Se tuottaa tappiota. Ongelmien ratkaiseminen tavoiteajassa voi olla haastavaa ja stressaavaa.

8.3 Testauksen merkitys liiketoiminnassa

Jotta voitaisiin tehdä laadukasta tuotetta, tarvitaan toimiva rahoitus ja liiketoimintamalli. Laadukas ja toimiva sovellus on erittäin tärkeä osa toimivaa liiketoimintaa. Jos ajatellaan esimerkiksi yritystä, jonka ydinliiketoimintaa on ohjelmistokehitys, niin toimimattomien sovellusten merkitys korostuu, koska ne tuottavat tappiota toimimattomuudellaan. Vaikutus voi heijastua koko liiketoimintaan.

Testaajat ovat oma ammattikuntansa. He ovat erittäin laatutietoisia ja osaavat soveltaa ammatti-osaamistaan eri projekteissa. Heidän asiantuntija tietämys on tärkeä osa toimivissa sovelluksessa.

Testauksen merkitys toimivien sovellusten kohdalla on merkittävä hyöty. Testaajat tekevät kaikkensa, jotta sovellus toimisi. Toimivan sovelluksen läpi vienti yrityksessä tuottaa voittoa.

9 Yhteenveto ja loppupohdinta

Onnistuneen testiajon läpivieminen on hyvin laaja kokonaisuus, joka pitää sisällään monta eri osa-aluetta. Eri osa-alueiden hallinta ja kokonaisuus näkymä projektissa vaatii hyvää hahmottamiskykyä, pitkäjänteisyyttä, hyvää paineensietokykyä, yhteistyötä eri sidosryhmien kanssa ja alan tuntemusta. Automaatio työkalujen tehtävänä on automatisoida osia testausprosessissa, joka helpottaa projektin läpivientiä. Muuten projektit saattaisivat kestää hyvin kauan, joka hidastuttaa sovelluksen pääsyä markkinoille. Tämä hidastaa prosessia ja sitä kautta tuottaa tappiota liiketoimelle.

Tutustuminen Selenium IDE- työkaluun oli hyvin mielenkiintoista. En ole ennen tehnyt näin laajaa projektia ja se opetti paljon hahmottamaan eri asioita mitä tulee ottaa huomioon tehdessä testiajoja. Kokonaisuudessaan projekti oli haastava, koska opittavaa oli paljon. Tiedon sisäistäminen vaatii aikaa ja käytännön kokemusta.

Selenium IDE:n käyttöönotto ja käyttö itsessään olivat melko helppoja. Testiautomaation suunnittelu oli vaikein kohta, koska se sisältää niin paljon erityisosaamista, monen asian samanaikaista hallintaa ja alan tuntemusta. Minua ohjasi testausalaan perehtynyt,

koodauspohjainen, tietoturva-alan ja laatukonsultointialan asiantuntija. Hän opasti, neuvoi ja tuki projektin läpiviemisessä. Hänen tietämyksensä ja ohjauksensa alasta sai minut hahmottamaan kokonaisuuden.

Koko testausala on hyvin mielenkiintoinen ja laaja kokonaisuus. Se sisältää valtavasti tietoa, erikoisosaamista ja laajaa tuntemusta eri tekniikoista. Opin paljon projektin aikana ja toivon pystyväni hyödyntämään osaamistani tulevaisuudessa.

Lähteet

Kirja lähteet:

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2010. Tutki ja Kirjoita. 15-16. painos. Helsinki: Tammi.

Kaner, C., Bach, J. & Pettichord, B. 2001. Lessons learned in software testing, ac-ontext-Driven Approach. 1st edition, New York, United States of America: Wiley Computer Publishing, John Wiley & Sons Inc.

Kaner, C. , Falk, J. & Hung, N.1999. Testing computer software. Second edition. New York, United States of America: Wiley Computer Publishing, John Wiley & Sons Inc.

Pressman, R., S. ,Adapted by Ince, Darrel. 2000 Software Engineering, a Practitioner's approach, Europian Adapation. Fifth Edition.England: McGraw-Hill International.

Ruuska, K. 2013. Pidä projekti hallinnassa. 7.painos. Helsinki: Talentum

Sähköiset lähteet:

Agile 2014. Viitattu 20.03.2014
http://agiletesting.com.au/

Alkoholilaki. Luettu 22.04.2014
http://www.finlex.fi/fi/laki/ajantasa/1994/19941344

Boundary value analysis. Viitattu 22.04.2014
http://istqbexamcertification.com/what-is-boundary-value-analysis-in-software-testing/

Bugiraportti. Viitattu 25.05.2014
https://bugzilla.mozilla.org/show_bug.cgi?id=931817

Eguivalence partitioning. Viitattu 22.04.2014
http://istqbexamcertification.com/what-is-equivalence-partitioning-in-software-testing/

Honolulu 2013. Viitattu 06.10.2013.
Philip Johnson, collaborative software development laborotary, information and computer science, university of Hawaii, Honolulu 96822 .
http://www.youtube.com/watch?v=9hz9aQztdrw

ISTQB, Vesiputousmalli 2014. Viitattu 30.03.2014.
http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/

ISTQB V-malli 2013. Viitattu 30.09.2013
http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/

ISTQB:n sanasto 2014. Viitattu 20.03.2014.
http://fistb.ttlry.mearra.com/sites/fistb.ttlry.mearra.com/files/istqb_sanasto_2007-12-20_fi_en.pdf.

ISTQB, Testing techniques 2014. Viitattu 20.03.2014
http://istqbexamcertification.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques/

Selenium IDE 2013. Viitattu 28.09.2013
http://www.Selenium IDEhq.org/download/

Selenium IDE demo. Luettu 30.09.2013.
http://www.youtube.com/watch?v=gsHyDIyA3dg

Tutorialspoint 2014. Viitattu 20.03.2014
http://www.tutorialspoint.com/software_testing/testing_methods.htm

V-malli, Software business competence 2014. Viitattu 04.04.2014.
http://www.oamk.fi/sbc/testaus/testaustasot.htm

Testaus, Talvio. Luettu.28.10.2013
http://www.cs.tut.fi/tapahtumat/testaus04/talvio.pdf

Julkaisemattomat lähteet:

Laatukonsultti. 2014.
Teemu Vesala, asiantuntija-apu ja konsultointi.

Kuvat