Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

ALEKSI RENFORS

Automaation ajansäästö regressio-testauksessa

Opinnäytetyö

TIETOJENKÄSITTELY
2023

Tekijä(t)
Renfors, Aleksi

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
4/2023

Sivumäärä
29 sivua

Julkaisun kieli
Suomi

Julkaisun nimi
Automaation ajansäästö regressiotestauksessa

Tutkinto-ohjelma
Tietojenkäsittely

Tiivistelmä
Opinnäytetyössä tutkittiin Visma Real Estate Oy:n web-käyttöliittymän regressiotestauksen automaation taloudellisuutta, sekä tehokkuutta tarkastelemalla, kuinka paljon testien suorittaminen automaation avulla säästää aikaa verrattuna niiden suorittamiseen manuaalisesti. Tämän lisäksi tutkittiin, mitä asioita tulisi ottaa huomioon automaatiota suunniteltaessa, jotta saavutettaisiin mahdollisimman laadukas lopputulos.

Työn tuloksena saavutettiin parempi ymmärrys pääkohdista, joita tulee ottaa huomioon automaation suunnitteluvaiheessa, sekä siitä, kuinka taloudellisia Robot Frameworkilla toteutetut web-käyttöliittymän automaatiotestit käytännössä ovat, kun otetaan huomioon automaation luomiseen ja ylläpitoon käytetty aika.

Asiasanat haetaan asiasanaluettelosta, mutta siihen ei tehdä linkitystä
Robot Framework, testiautomaatio, regressiotestaus

Author(s) Renfors, Aleksi	Type of Publication Bachelor’s thesis	Date 4/2023
	Number of pages 29 pages	Language of publication: Finnish
Title of publication Time saved by automating regression testing		
Degree programme Business Information Systems		

Abstract

In the thesis, the time saved by automating regression testing of Visma Real Estate Oy's web user interface was studied, as well as its efficiency by examining how much time is saved by performing the tests with automation compared to performing them manually. In addition, it was investigated what factors should be considered when designing automation in order to achieve the highest possible quality outcome.

As a result of the work, a better understanding of the main points to be considered in the design phase of automation was achieved, as well as how cost-effective the Robot Framework-implemented web user interface automation tests are in practice, considering the time spent on creating and maintaining the automation.

Key words search from key word list but not link

Robot Framework, test automation, regression testing

SISÄLLYS

LÄHTEET

1 JOHDANTO

Tässä opinnäytetyössä käsitellään testauksen automatisointiin liittyviä asioita testiautomaation ajansäästön puolesta, kuten mitä asioita tulee ottaa huomioon, jotta testiautomaatio saadaan toteutettua järkevästi, sekä mahdollisimman tehokkaasti. Tämän lisäksi pyritään selvittämään kuinka paljon Visma Real Estaten tuotteisiin tehdyt web-käyttöliittymän automaatiotestit käytännössä säästävät aikaa ja millaisella aikavälillä testit maksavat itsensä takaisin.

Opinnäytetyön aihe valikoitui omasta mielenkiinnosta testiautomaatiota kohtaan. Automaatiotestit ovat suuressa osassa testauksessa Visma Real Estatella, joten niiden säästämä aika käytännössä tuntui erittäin mielenkiintoiselta tutkimusaiheelta. Testiautomaatio on yleisestikin hyvin keskeisessä osassa nykyajan ohjelmistotuotannossa. Kun ohjelmistojen koko kasvaa, kasvaa myös testattava kuorma. Jotta kaikki tarvittavat asiat saataisiin testattua, vaadittaisiin päivittäin ajettavien testien suorittamiseen ohjelmiston koosta riippuen kymmeniä, ellei jopa satoja testaajia ilman automaatiota. Testauksen automatisointi mahdollistaa erittäin kattavan säännöllisen testauksen järkevällä resurssienkäytöllä.

2 TESTIAUTOMAATION HYÖDYT

Testaamisen tarkoituksena on varmistaa sekä laatukriteerit että ohjelmiston toiminen määritelmien mukaan (Kestikievari n.d.). Testaamista voi suorittaa manuaalisesti tai automaation avulla. Manuaalisessa testauksessa käytännössä kehittäjän, tai testaajan toimesta testataan kehitettyä työtä käsin. Tuotteen testaaminen manuaalisesti vie kuitenkin vähintään yhden ihmisen kaikki resurssit siltä aikaa, kun tuotetta testataan. Tämän lisäksi pelkästään uuden/ korjatun ominaisuuden testaaminen ei riitä, vaan tulisi aina tehdä regressiotestausta, eli vanhojen testien suorittamista, jotta saadaan selville, ettei uusi päivitys ole rikkonut vanhoja ominaisuuksia.

2.1 Rahan ja ajan säästäminen

Jos kaikki tarvittava testaus tapahtuisi ihmisten toimesta, tarvitsisi siihen käyttää erittäin suuria määriä rahaa, jotta saataisiin palkattua/ ulkoistettua testaajia suorittamaan edes välttämättömät testit. Automatisoitu testaaminen eli testausautomaatio onkin aivan ehdoton tarve nykypäivän ohjelmistoalalla ajan ja rahan säästämiseksi.

Käytännössä testausautomaation avulla voidaan hoitaa suurilta osin esimerkiksi regressiotestaus, sekä ajastettu testaus (Kestikievari n.d.). Mihin testausautomaatiota kannattaa käyttää)? Jos regressiotestaus saadaan suoritettua robotin avulla, tarvitsee testaajan testata vain uuden ominaisuuden toiminta. Ajastettu testaus saadaan myös hoidettua erittäin hyvin robotin avulla, koska yleensä parhaat ajat laajempaan testaukseen ovat vähäisen käyttökuorman vuoksi työajan ulkopuolella, esim. yöllä, jolloin on mukavampi nukkua kuin tehdä töitä.

2.2 Testien kattavuus ja skaalautuvuus

Testikattavuus voidaan saada suuremmaksi automaation avulla. Robotin suorittaessa testejä voidaan saada katettua paremmin erilaiset käyttötapaukset, kuin mitä manuaalisesti testaamalla olisi mahdollista (Mulders 2019). Sen sijaan, että joutuisi joka kerta tekemään mutkikkaita ja pitkiä testitapauksia, voi kirjoittaa siitä yksittäisen robottitestin, jonka pystyy ajamaan niin monta kertaa kuin tarpeellista.

Mutkikkaiden ja epämieluisien testitapauksien lisäksi automaatio auttaa suuresti skaalautuvuuteen. Kun ohjelman koko kasvaa, kasvaa samalla myös testattava kuorma. Kun testit on automatisoitu, pystytään helposti lisäämään uusia testitapauksia täydentämään vanhoja testiskriptejä, jolloin kaikki uudet testitapaukset tulevat automaatiotestauksen pariin. Testien suorittaminen eri ympäristöjen välillä helpottuu myöskin huomattavasti, kun pystytään ajamaan sama testitapaus napin painalluksella missä tahansa ympäristössä.

2.3 Testien tarkkuus ja luotettavuus

Testien tarkkuus saadaan sataprosenttisen tarkaksi robottitestien avulla. (Mulders 2019) Vaikka ihminen olisi kuinka tarkka, tulee välillä silti tehtyä virheitä. Varsinkin silloin, kun työpäivä on kestänyt yli kahdeksan tuntia ja yöunia ehti kertymään vajaat kuusi tuntia. Robotti tekee juuri sen, mitä sen on käsketty tekemään, eli käytännössä esim. ihmisille tuttua ajatusvirhettä ei voi koskaan tapahtua.

2.4 Itseään toistavien tehtävien eliminointi

Automaation avulla voidaan laadunvarmistuksessa työskentelevien testaajien arjesta ottaa pois itseään toistavaa, monestikin tylsää manuaalista testausta, joka voi lisätä työn mielekkyyttä, sekä vähentää koko laadunvarmistustiimin stressiä vähentyneen kuorman ansiosta. (Mulders 2019) Sen sijaan, että testaaja joutuisi päivästä toiseen toistamaan samoja manuaalisia testejä aamusta iltaan, voi aikaa jäädä enemmän esimerkiksi testausautomaation kehittämiselle ja sen ylläpitämiselle. Testaajien aikaa saadaan siis käytettyä sellaisiin tehtäviin, mitä ei automaatiolla voida hoitaa. Tällä tavoin voidaan keventää testaajien henkistä taakkaa itseään toistavilta tehtäviltä. Voidaankin ajatella, että testiautomaation perimmäisenä ideana on vähentää tarve ihmisten ajankäytölle testien suorittamiseen niin pieneksi, kuin mahdollista.

2.5 Tesiten suorittamisen nopeus ja virheiden löytyminen

Bugien löytymistä voidaan nopeuttaa automaation kautta huomattavasti yksikkötestien avulla (Bose 2021). Kun esimerkiksi kehittäjä on saanut uuden ominaisuuden valmiiksi ja se vaikuttaa toimivan hänen itsensä suorittaman testauksen mukaan oikein, voidaan testiautomaation avulla varmistaa, ettei luomansa uusi ominaisuus riko mitään vanhaa, jos vanhoille ominaisuuksille on yksikkötestit tehtynä. (Yksikkötestillä tarkoitetaan koodin yksittäiselle pienimmälle mahdolliselle osalle, esimerkiksi funktiolle tehtävää testiä) Parhaimmassa tapauksessa tämä tapahtuu yhden napin painalluksella, jonka jälkeen kehittäjä voi turvallisimmin mielin tehdä muutoksia ja lisätä uusia ominaisuuksia koodiin.

3 TESTIAUTOMAATIOON LIITTYVÄT ONGELMAT

Vaikka testauksen automatisoinnista on paljon hyötyä, on siinäkin asioita, joita tulee ottaa huomioon ennen kuin alkaa laskemaan sen avulla säästettyä aikaa.

3.1 Vaadittu osaaminen

Testien automatisointi ei tapahdu itsestään, eli toisin sanoen tarvitaan henkilö, joka osaa käyttää testauksen automatisointiin valittuja ohjelmointikieliä, frameworkeja, yms. työkaluja. Se taas tarkoittaa sitä, että jos esimerkiksi testaajat huolehtivat testien automatisoinnista, tarvitsee kehittäjien lisäksi myös testaajilla olla jonkin asteista osaamista ohjelmoinnista. Ja vaikka osaamista löytyisi, tarvitaan testien automatisointiin myöskin aikaa. (Safonova 2022.)

Testitapauskohtaisesti myöskin testien tekeminen voi olla hankalaa. Tämä saattaa esim. näkyä, kun pyritään tekemään mahdollisimman ylläpidettävää, mutta samalla myöskin luotettavaa testiautomaatiota. Koska testattaviin ominaisuuksiin voidaan tehdä, ja yleensä tehdäänkin jatkuvasti päivityksiä, tulisi testiautomaation olla helposti muokattavissa takaisin toimivaksi näiden päivitysten sattuessa. Samalla testien tulisi myös olla tarpeeksi monimutkaisia, jotta kaikki tarvittava asia tulisi testattua. Tästä johtuen, kun pyritään tekemään mahdollisimman täydellistä testiautomaatiota, vaatii se harkittua suunnittelua, sekä ohjelmistokehityksen taitoja. (Patt 2021.)

3.2 Automaation toteuttaminen ja ylläpito

Vaikka testiautomaation päämääränä on ajan säästäminen, joudutaan kuitenkin käyttämään aikaa testiautomaation luomiseen. Tuleekin ottaa huomioon, että järkevästi toteutetun automaation tekeminen ja suunnittelu saattavat viedä huomattavan määrän aikaa sekä rahaa (Da Silva 2022).

Vaikka aika ja taito riittäisivätkin testiautomaation luomiseen, tulee myöskin pohtia, mitä testejä tulisi automatisoida. Kaikkia testitapauksia ei lähtökohtaisesti ole kannattavaa lähteä automatisoimaan, joten ennen testien tekemistä tulisi selvittää, mistä ohjelman osa-alueiden testauksen automatisoinnista olisi eniten hyötyä, kun otetaan huomioon tekemiseen käytettävä aika, sekä testien ylläpito (Safonova 2022). Tulee myöskin harkita, kuinka tarkasti testattavan ominaisuuden testausta tullaan automatisoimaan. Mitä tarkempia ja mitä useampia testitapauksia tehdään, sitä suuremman ylläpitokuorman ja tekemiseen käytettävän ajan se vaatii.

Testiautomaation tekemiseen liittyvien ongelmien lisäksi yhtenä pulmana on se, kun automaatiotestit palauttavat ns. False positiveja. False positiveilla tarkoitetaan sitä, kun automaatiotestien tulokset näyttävät virhettä, mutta todellisuudessa virhettä ei oikeasti ole. Tällaisia tilanteita voi esim. syntyä, kun ei ole osattu huomioida kaikkia mahdollisia tilanteita, joita testien ajon aikana voi syntyä, tai jos ollaan testaamassa nettisivujen käyttöliittymää ja nettiyhteydessä sattuu olemaan pieniä katkoksia (Da Silva 2022). Näitä ongelmia ei periaatteessa koskaan saa kitkettyä kokonaan, mutta niiden määrä voidaan saada minimiin hyvällä suunnittelulla.

4 AUTOMAATION SUUNNITTELU

Kun pyritään säästämään aikaa automaation avulla, tulisi suunnitelma automaation toteuttamiseksi olla mahdollisimman pitkälle mietittynä ennen sen toteuttamista mahdollisimman hyvän lopputuloksen saavuttamiseksi (Hedge 2022). Samoin kuin monessa muussakin asiassa, ilman järkevää suunnitelmaa, ei lopputulos yleensä ole toivotun mukainen, eikä ikäviltä yllätyksiltä vältytä tekovaiheessa. Mitä pidemmälle työ saadaan suunniteltua, sitä varmemmin se saadaan onnistuneesti valmiiksi. (Kushnir 2022.) Kattavan suunnitelman avulla pienennetään huomattavasti esimerkiksi umpikujaan ajautumisen riskiä kehitysvaiheessa, ja pienennetään samalla myös riskiä joutua myöhemmin kirjoittamaan koko automaation logiikka uudelleen, jos testitapauksia tarvitseekin laajentaa, tai työkalut eivät ole laajennuksien kanssa yhteensopivia. (Hedge 2022.) Tuleekin siis ottaa huomioon, että automaation kokonaisuudessaan säästämä aika pohjautuu suurelta osin automaation suunnitelman laatuun.

4.1 Tavoitteiden määrittäminen

Testiautomaatiota suunniteltaessa tulisi aluksi pohtia sille asetettuja tavoitteita. Kun ryhdytään luomaan suunnitelmaa testiautomaation toteuttamiseksi, tulisi ensimmäiseksi miettiä mitkä ovat automaation tavoitteet. Tavoitteiden avulla saadaan parempi ymmärrys siitä, mitä automaatiolla käytännössä halutaan saavuttaa, minkä avulla taas saadaan varmistettua, että automaatio voidaan toteuttaa vastaamaan siltä odotettua arvoa. (Hedge 2022.) Tavoitteiden olisi suotavaa huomioida ainakin seuraavat kohdat:

- Miksi testiautomaatio halutaan toteuttaa?
- Minkä arvon testiautomaatio tuo projektille?
- Mikä on korkean tason suunnitelma automaation toteutukselle?
- Miten automaation toteutus jaotellaan käytössä olevalle ohjelmistokehityksen mallille?

Jos saadaan vastaus kaikkiin edellä oleviin kysymyksiin, voidaan niiden pohjalta paremmin suunnitella automaatio vastaamaan sille asetettuja tarpeita. Jos taas selkeitä tavoitteita ei ole annettu, ei pystytä myöskään tekemään tarkkaa suunnitelmaa projektin toteuttamiselle. Tällaisessa tapauksessa voidaan joutua tilanteeseen, jossa toteutettu automaatio ei tuo siltä odotettua liikearvoa. Hyvällä suunnittelulla saadaan projekti varmemmin onnistumaan (Jotform 2023).

4.2 Automatisoitavien testien valitseminen

Kaikkea mahdollista ei ole järkevää automatisoida. Jos testi ajetaan vain muutaman kerran, tai testattava asia on kyseisellä hetkellä jatkuvan kehityksen alla, ei ole välttämättä järkevää toteuttaa sille automaatiota, koska itse automaation suunnittelu, ylläpito ja toteutus saattavat viedä huomattavan suuren ajan verrattuna siihen, mitä automaatio tulisi säästämään manuaalitestaukselta. Toisaalta taas, vaikka testi olisikin hyvin pienellä vaivalla tehtävissä manuaalisesti, voi kyseisen testin automatisointi silti olla kannattavaa, jos sitä joudutaan suorittamaan usein eri ympäristöissä (SmartBear 2023).

Pohdittaessa tulisiko jokin testi automatisoida, tulee miettiä ainakin seuraavia asioita:

- Kuinka usein kyseinen testitapaus tarvitsee toistaa?
- Kuinka suuri ylläpitokuorma testin automatisoinnista tulisi?
- Kuinka vaativaa automaation toteutus kyseiselle testille on manuaalitestaukseen verrattuna?

Jos testitapausta tarvitsee toistaa usein, voi sen automatisointi olla järkevää. Kun testiä toistetaan esimerkiksi kerran päivässä ja testin suorittaminen vie ihmiseltä viisi minuuttia, tulee tästäkin viisipäiväisen työviikon aikana jo 25 minuuttia. Jos ajatellaan, että vuodessa olisi 40 työviikkoa, tulee pelkästään tästä yhdestä viiden minuutin testin tekemisestä joka vuosi 1000 minuuttia, eli 16,67 tuntia, mikä taas vastaa yli kahta työpäivää. Eikä tässä laskelmassa edes oteta huomioon kuluvaa aikaa, tai haittavaikutuksia, kun siirrytään erityyppisten tehtävien välillä. Kun hypitään eri asioiden välillä, kuluu aikaa, ennen kuin pystyy täysin keskittymään uuteen tehtävään. Jos joudutaan lyhyin väliajoin hyppimään tehtävästä toiseen, voi se vaikuttaa negatiivisesti keskittymiskykyyn ja luovuuteen (Raeburn 2022).

4.3 Automaatioon käytettävät työkalut

Automaation tavoitteiden ja automatisoitavien testien ollessa selvillä, voidaan näiden pohjalta valita kyseiseen tapaukseen sopivat työkalut. Testauksen automatisointiin on nykypäivänä olemassa hyvin paljon erilaisia työkaluja, joiden avulla voidaan toteuttaa erityppisten testien automatisointi. Web-käyttöliittymän testaukseen tarkoitettuja työkaluja ovat esimerkiksi Robot Framework, Selenium, sekä WebdriverIO. Pohdittaessa mikä työkalu on oikea omiin tarpeisiin, on suotavaa miettiä seuraavia asioita:

- Vaativatko automaatioon valitut työkalut erillistä koulutusta?
- Kuinka hyvin valitut työkalut sopivat yhteen muihin ohjelmistossa käytettyjen työkalujen kanssa?
- Kuinka laadukasta tukea on saatavilla valittujen työkalujen käyttöön?
- Kuinka hyvin työkaluilla tehtyjä testejä voidaan tulevaisuudessa laajentaa?
- Onko työkalut kehitetty tukeman alati kehittyviä teknologioita?
- Mitkä ovat työkalujen käytön kustannukset?

4.3.1 Avoimen lähdekoodin työkalut

Avoimen lähdekoodin työkalut ovat erittäin hyvä ratkaisu, jos niiden ominaisuudet vastaavat omia tarpeita. Niiden käyttö ja opetusmateriaalit ovat ilmaisia ja niin kauan, kun niiden ympärillä on aktiivinen yhteisö, ne kehittyvät jatkuvasti ja tukea ongelmiin voi saada ilmaiseksi nopeallakin vastausajalla. Toisaalta taas huonona puolena on juuri se, että ne kukoistavat vain niin kauan, kun aktiivinen yhteisö on olemassa. Yhteisön hiipuessa myös kehitys ja tuki avoimen lähdekoodin järjestelmille hiipuu. (Shvets n.d.) Esimerkkejä avoimen lähdekoodin testiautomaation työkaluista: WebdriverIO, Selenium ja Robot Framework. Jos avoimen lähdekoodin työkalut eivät täysin sovi omiin tarpeisiin, voidaan niiden pohjalta kehittää omia tarpeita vastaava kustomoitu toteutus. Tähän kuitenkin vaaditaan osaamista, josta käytännössä joudutaan maksamaan joko ulkoiselle henkilölle, tai sisäisesti työtuntien muodossa. (Shvets n.d.)

4.3.2 Maksulliset työkalut

Maksullisilta työkaluilta usein löytyy erittäin laaja valikoima ominaisuuksia ja varma tuki. Niistä saattaa kuitenkin joutua maksamaan suhteellisen korkean hinnan ilmaisiin avoimen lähdekoodin vastaaviin työkaluihin verrattuna. (Reverchuk 2023.) Tästä johtuen onkin suositeltavaa testata maksullisia työkaluja kokeiluversion muodossa ennen, kuin ostaa itselleen koko tuotetta. Maksullisia testiautomaation työkaluja ovat esimerkiksi Cucumber, Avo Assure ja ZeuZ.

5 TUTKIMUSONGELMA

Tässä osassa esitellään tutkimusongelman tuloksia, eli web-käyttöliittymän automaatiotestauksen ajansäästöä regressiotestauksessa Visma Real Estaten tuotteessa.

5.1 Käsiteltävän alueen rajaaminen

Tarkoituksena opinnäytetyöllä oli kartoittaa web-käyttöliittymään automaation avulla suoritetun regressiotestauksen ajansäästöä. Vertailukohteeksi tähän valittiin Visma Real Estaten taloushallintotuotteen web-käyttöliittymään Robot Frameworkin avulla toteutetut robottitestit. Testejä on tehty eri tuotteiden välillä tuhansia, mutta koska kaikkien testien läpikäyminen ei olisi siihen kuluvan ajan puolesta järkevää, käytetään vain yhteen tuotteeseen tehtyjä testejä vertailukohteena. Kaikki vastaavat testit on tehty samoilla työkaluilla ja niissä on käytetty samoja tekniikoita, kuin valituissa testeissä, joten lopputuloksen pitäisi täten olla verrattavissa myös muihin testeihin.

5.2 Tutkimusongelma

Visma Real Estaten tuotteiden web-käyttöliittymän manuaalisia testisuunnitelmia on laajalti tuotu automaatiotestauksen pariin ja automaatiota kehitetään jatkuvasti. Näin ollen halutaan saada parempi kuva siitä, kuinka paljon aikaa regressiotestien automatisointi käytännössä säästää.

Jotta saadaan ymmärrys automaation ajansäästöstä, tulisi saada selvyys seuraaviin kysymyksiin:

- Kuinka tehokkaita ajankäytön puolesta automaatiolla suoritetut testit ovat manuaalisesti suoritettuihin verrattuna?
- Onko testeistä mahdollista saada tehokkaampia?
- Kuinka paljon automaation toteuttamiseen käytetty aika vaikuttaa sen taloudellisuuteen?
- Kuinka paljon automaation ylläpitoon käytetty aika vaikuttaa sen taloudellisuuteen?

5.3 Tavoite

Tavoitteena on saada kartoitettua testeihin kuluvia aikoja manuaalisesti, sekä automaation avulla, ja tarkastella, kuinka paljon ja minkä kokoisessa aikaikkunassa testien au-

tomatisointi tulee maksamaan itsensä takaisin otettaessa huomioon niiden toteuttamiseen ja ylläpitoon kuluva aika. Tämän lisäksi pyritään selvittämään, onko mahdollista saada nykyisistä automaatiotesteistä tehokkaampia eri työkaluilla.

6 KONKRETIA

Tässä osassa esitetään tuloksia testien suoritusten ajoista, niiden tehokkuuteen liittyvistä asioista, sekä käytännön esimerkkejä testien toteuttamiseen ja ylläpitoon käytetyn ajan vaikutuksista automaation taloudellisuuteen.

6.1 Testien suoritukseen kuluva aika

Jotta voidaan kartoittaa regressiotestauksen automatisoinnin säästämää aikaa, tarvitsee ensiksi ymmärtää, kuinka paljon automaatiotestit itsessään säästävät aikaa verrattuna siihen, että ne suoritettaisiin manuaalisesti. Tätä varten valittiin yhden tuotteen kaikki Robot Frameworkilla tehdyt web-käyttöliittymän smoke testit vertailukohteeksi, joita on yhteensä 62 kappaletta. Kyseiset testit suoritettiin manuaalisesti, sekä automaation avulla, jotta saatiin selvitettyä aika, joka niiden suorittamiseen käytännössä kului.

6.2 Robot Framework lyhyesti

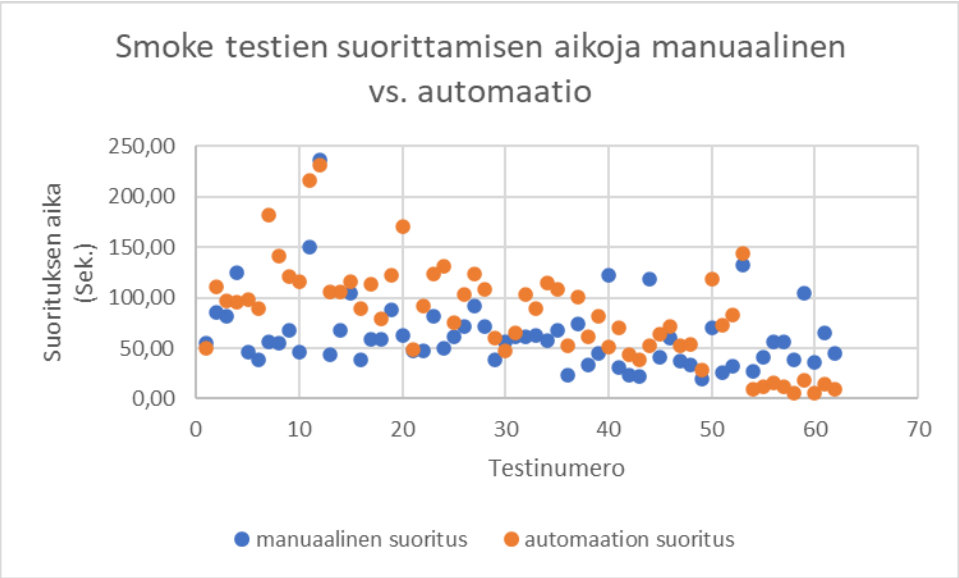
Robot Framework on suomalaisen Pekka Klärckin diplomityöhön pohjautuva testausautomaatioon suunnattu, pythonilla kirjoitettu geneerinen avoimen lähdekoodin framework, jolle voidaan halutessa kirjoittaa mm. Python, sekä Java –pohjaisia kirjastoja toiminnallisuuden laajentamiseksi (Klärck 2006). Robot Framework käyttää omaa syntaksiaan, jonka tarkoituksena on olla mahdollisimman helposti ymmärrettävää. Robot Frameworkille on saatavilla ulkoisia kirjastoja, mm SeleniumLibrary, sekä Browser Library selainpohjaista käyttöliittymätestausta varten (Robot Framework 2023).

6.2.1 Testien suoritukseen kuluvan ajan mittauksia

Tässä esitetään Visma Real Estaten taloushallintajärjestelmän web-käyttöliittymään tehtyjen smoke testien suorittamisen aikoja manuaalisesti ja automaation avulla. Testit tehdään aina oman testaussuunnitelmansa mukaan, joten itse testit on suoritettu täysin saman kaavan mukaan manuaalisesti ja automaation avulla. Kaikki testit ja niissä testattavat asiat eroavat toisistaan. Testit ovat kooltaan suunnilleen samankokoisia, tosin testattavasta asiasta riippuen niissä on toiminnallisuuden ja kompleksisuuden puolesta käytännön eroja.

Manuaalitestauksen suoritin itse kellottamalla jokaisen testin erikseen. Testien selaimena Google Chrome. Ainoana erona manuaalisesti ja automaation avulla suoritettujen testien kulussa on se, että automaatiolla suoritetuissa testeissä avataan aina ensiksi selain, jonka jälkeen siirrytään oikeaan URL-osoitteeseen suorittamaan testiä. Manuaalisissa suorituksissa selain oli lähtökohtaisesti jo avattu ja oltiin oikeassa URL-osoitteessa. Tämä johtuu siitä, että pyrittiin simuloimaan mahdollisimman tarkasti todellista tilannetta, jossa selainta ei välttämättä suljeta testien välillä.

Kuviossa 1 on nähtävillä 62 testin suorituksiin kuluneet ajat manuaalisesti tehtyinä, sekä automaation avulla. Kuten kuviosta näkyy, keskimäärin yksittäisen testin suorittaminen onnistui nopeammin manuaalisesti. Tulee kuitenkin ottaa huomioon, että testit ja testattavat ominaisuudet olivat minulle tuttuja. Jos testit suorittaisi henkilö, jolle ohjelma ei ole tuttu, veisi niiden suorittaminen suurella varmuudella huomattavasti enemmän aikaa. Tämän lisäksi manuaalisen suorituksen tehokkuuteen voi liittyä inhimillisiä tekijöitä, kuten ihmisille tyypilliset virheet ja ulkopuoliset häiriöt. Testin tulos voi olla täysin epävalidi, jos esimerkiksi unohtaa yhdenkin kymmenistä tai sadoista eri testivaiheista.



Kuvio 1. Manuaalisesti ja automaation avulla suoritettuihin smoke testeihin kuluva aika.

Taulukko 1. Manuaalisesti ja automaation avulla tehtyjen testien kulunut aika yhteensä.

Manuaali yhteensä (sekuntia)	Automaatio yhteensä (sekuntia)
3917,94	5194,89

Smoke testien manuaaliseen suorittamiseen kului yhteensä ~65 minuuttia (3917,94/60 = 65,3 min.), kun taas automaatiolla vastaava aika oli ~87 minuuttia (5194,89/60 = 86,6 min.). Automaatio siis suoritti samat testit ~33 % (87/65 = 1,33) pidemmässä ajassa. Jos yksittäisten testien suoritusten ajat summataan yhteen, voidaan siis todeta, että tästä näkökulmasta kyseisten testien tapauksessa, automaation avulla suoritetut testit eivät ole yhtä tehokkaita, kuin manuaalisesti tehtyinä. Tämä ei kuitenkaan kerro automaation tehokkuudesta käytännössä, koska tässä ei oteta huomioon esimerkiksi sitä, että automaatiotestit voidaan ajaa rinnakkain, jolloin useampaa testiä suoritetaan saman aikaisesti. Tarkastellaan seuraavaksi tarkemmin mihin asioihin automaatiotesteissä aikaa kului ja miten eri tekniikat mahdollistavat paremman taloudellisuuden automaatiotestaukselle siihen kuluvan ajankäytön puolesta.

6.2.2 Mihin ylimääräinen aika kului automaatiotesteissä?

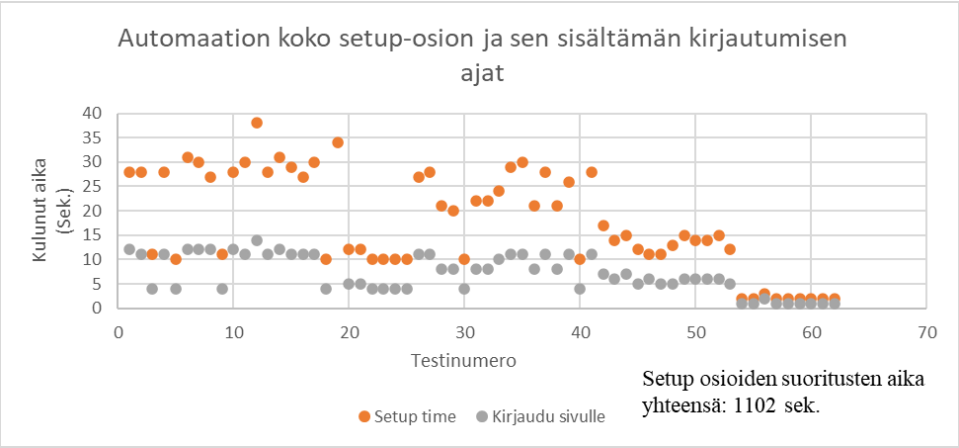
Suoritetuissa testeissä on käytetty kahta eri Robot Frameworkin web-käyttöliittymän testiautomaatioon suunnattua kirjastoa: SeleniumLibrarya, sekä Browser librarya. Nämä molemmat pohjautuvat eri selaimen hallintaan perustuviin teknologioihin. SeleniumLibrary käyttää Selenium Webdriver-moduulia, kun taas Browser Library pohjautuu Playwright-kirjastoon (Browser Library 2023; SeleniumLibrary 2023). Molemmilla kirjastoilla pystytään käytännössä tekemään samoja asioita, mutta koska ne ovat sisäisesti rakennettu eri tavoin, voi niiden tehokkuudessa ajankäytön osalta olla eroja.

Käytännössä kaikki automaation avulla suoritetut testit koostuvat kahdesta eri kohdasta: testin alustus osio, eli setup-osio, sekä itse testin suoritus. Manuaalisesti tehdyissä testeissä ei avattu selainta testien välissä erikseen. Tässä asiassa manuaalisesti suoritettujen testien ja automaation kulku eroaa toisistaan. Kun selain avataan joka testin välissä uudelleen, tarvitsee selaimen avautumisen jälkeen siirtyä oikeaan URL-osoitteeseen kirjautumissivulle, jonka jälkeen kirjaudutaan palveluun. Nämä kohdat on sisällytetty jokaisen automaatiotestin setup-osioon.

6.2.2.1 Automaatiotestien setup-osio

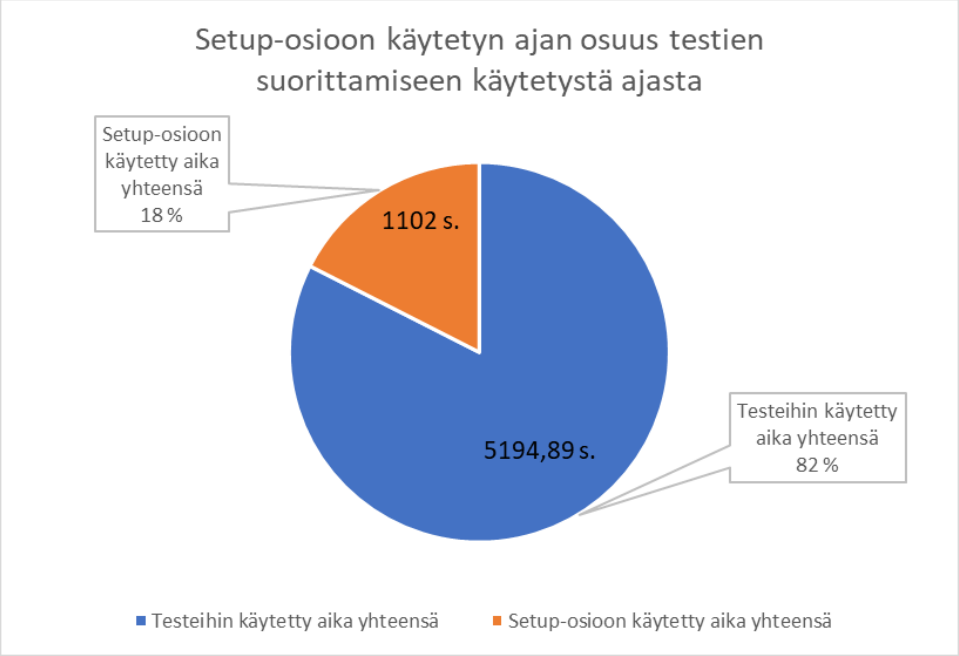
Automaatiotestien setup-osiossa pääkohtina ovat selaimen avaus oikeaan URL-osoitteeseen, sekä sisäänkirjautuminen. Näiden lisäksi voidaan tapauksen mukaan avata nettisivulta testattava toiminto.

Kun tarkastellaan kuviossa 2 olevien automaation avulla suoritettujen smoke testien setup-osioiden aikoja, voidaan huomata, että korkeimmillaan ylletään lähes 40 sekuntiin. Lyhyimmillään, pääosin loppupään testeissä, setuppien aika taas näyttää noin kahta sekuntia. Ero on huomattava, kun vertaa viimeisimpien testien setuppien aikoja alkupään osioon. Käytännössä tämä johtuu testeissä käytettyjen kirjastojen tehokkuuden eroista, jota tutkitaan tarkemmin tulevassa kappaleessa. Kokonaisuudessaan setup-osioon kulutettiin kaikkien smoke testien osalta yhteensä 1102 sekuntia.



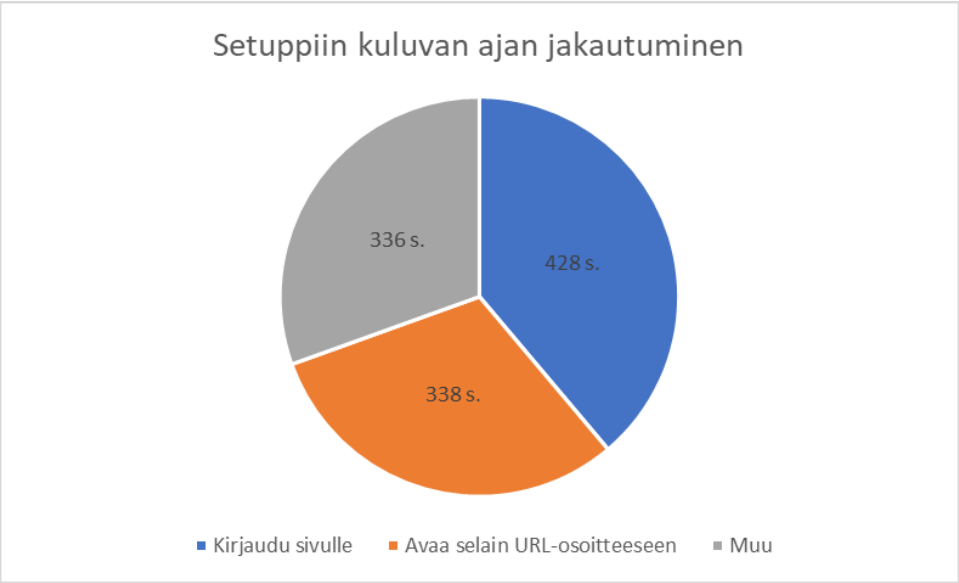
Kuvio 2. Automaation avulla suoritettujen smoke testien setup-osion ja setup-osioon sisältyvän kirjautumisvaiheeseen kuluvat ajat jokaisen testin osalta.

Kuviosta 3. nähdään, että setup-osio vei kokonaisuudessaan ~18 % testien suoritusajasta. Setup-osio saattaa kuitenkin testistä riippuen sisältää testikohtaisia vaiheita, joten setup-osioon kuluva aika tulee ensiksi erotella, jotta voidaan tarkemmin määritellä testistä riippumaton setupin osuus.



Kuvio 3. Setup-osioon kuluva aika suhteutettuna koko testin suorittamisen aikaan.

Kuten kuviosta 4. nähdään, kirjautuminen, sekä selaimen avaus oikeaan URL-osoitteeseen veivät suurimman osan käytetystä ajasta. Kun ei oteta huomioon Muu-osioon käytettyä aikaa setup-osion ajassa, saadaan tulokseksi 766 = (428 + 338). Kokonaisuudessaan kirjautumiseen ja selaimen avaukseen käytettiin siis 766 sekuntia.



Kuvio 4. Setuppiin käytetyn ajan jakautuminen.

Kappaleessa 6.2.1 todettiin automaatiotestien suorituksen kestäneen keskimäärin 33 % kauemmin manuaaliseen suoritukseen verrattuna. Manuaalisesti suoritettuna testeissä ei kuitenkaan tehty automaatiotestien sisältämän setup-osion asioita, joten otetaan tämä aika pois automaatiotestien kestoista ja lasketaan tulos uudelleen. Automaatiotestien suoritusaika ilman selaimen avausta, url-osoitteeseen siirtymistä ja sisäänkirjautumista on siis 4428,89 sekuntia (5194,89–766). Manuaalisen suorituksen kokonaiskesto on 3917,94 sekuntia. Kun verrataan automaation suoritusaikaa manuaaliseen siten, että molemmissa tapauksissa suoritetaan täysin samat asiat, on automaation avulla suoritettujen testien kokonaisaika enää 13 % (4428,89/3917,94 = 1,13) pidempi manuaalisesti suoritettuihin verrattuna. Tällä tavoin katsottuna automaatiolla suoritetut testit ovat yksittäinkin suoritettuina lähes yhtä tehokkaita, kuin manuaalisesti tehtyinä.

6.2.3 Selaimen testiautomaatioon tarkoitettujen kirjastojen vaikutus automaatiotestien suoritusaikoihin

Aiemmassa kappaleessa kerrottiin, että automaatiotesteissä on käytetty kahta eri kirjastoa selaimen hallintaan. Nämä kaksi kirjastoa ovat siis SeleniumLibary, sekä Browser Library.

6.2.3.1 SeleniumLibrary

SeleniumLibrary on selainpohjaiseen testaukseen tarkoitettu ulkoinen kirjasto Robot Frameworkille, joka käyttää Selenium WebDriver moduuleita selaimen hallintaan (GitHub 2021). PyPi:n julkaisuhistorian (PyPi 2021) mukaan ensimmäinen versio SeleniumLibrarystä julkaistiin vuonna 2011. Koska kirjastoa on kehitetty huomattavan pitkään, se sisältää suhteellisen laajan määrän kustomoituja ominaisuuksia. Tosin, uudemman vastaavaan tarkoitukseen suunnatun kirjaston, Browser Libraryn kehityksen johdosta SeleniumLibraryn tuki saattaa heiketä tulevaisuudessa, koska SeleniumLibraryn ylläpitäjä kuuluu Browser Librarya kehittävään tiimiin.

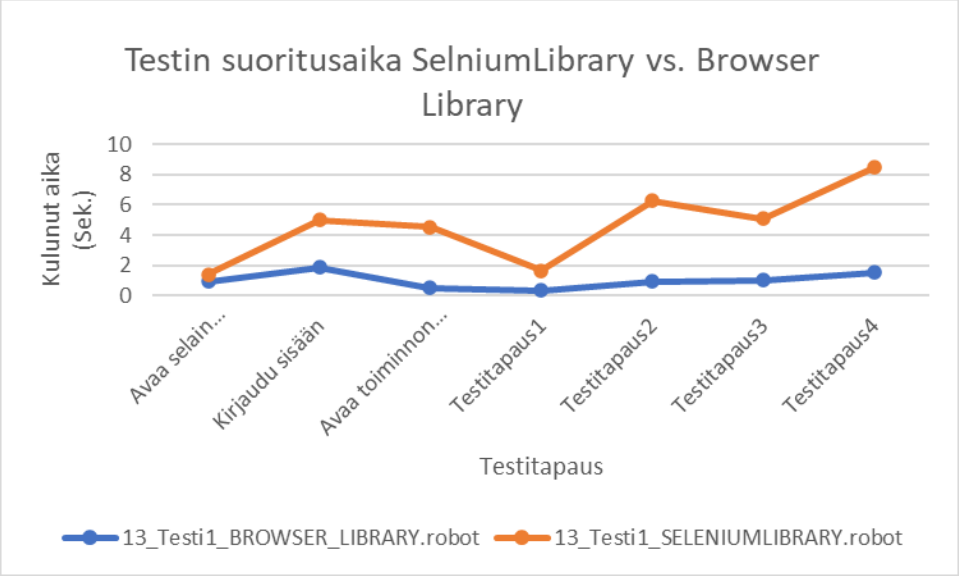
6.2.3.2 Browser Library

Browser library on suhteellisen uusi Robot Frameworkille suunnattu kirjasto selaimen testauksen automatisointiin. Se on SeleniumLibraryn vanhenevan teknologian johdosta kehitetty tarjoamaan tukea myös uusille web-teknologioille. Browser Library käyttää Playwright–kirjastoa selaimen hallintaan. Koska Playwright on tarkoituksella suunniteltu tarjoamaan tukea moderneille web-teknologioille, on se varteenotettava valinta SeleniumLibraryn sijasta tulevaisuutta ajatellen (Playwright 2023). Käytännössä SeleniumLibraryyn verrattuna Browser Library tarjoaa mm. huomattavasti paremman tuen shadow domeille ja erityyppisten web-selectorien ketjuttamiselle, sekä mahdollisuuden käyttää WebKittiä testeissä käytettävänä selaimena. (Browser Library 2022).

6.2.3.3 SeleniumLibraryn ja Browser Libraryn vaikutus testin suoritusaikaan käytännössä

Yhteensä 62 automaation avulla ajetusta testistä yhdeksän käytti SeleniumLibraryn sijaan Browser Librarya. Koska Browser Libraryn avulla tehtyjen testien osuus oli niin pieni, päädyin kirjoittamaan identtisen testin molempia kirjastoja käyttäen. Testi koostui setup-osiosta, mikä sisältää selaimen avaamisen, ja sisäänkirjautumisen, sekä neljästä eri testitapauksesta, joissa käytännössä klikataan auki linkkejä ja tarkistetaan, että ne toimivat.

Kuviosta 5. voidaan todeta, että Browser Librarya käyttäen testin suoritus oli huomattavasti nopeampaa, kuin SeleniumLibrarylla tehty vastaava. Tulokseen voi toki vaikuttaa se, mitä asioita testissä tehdään, mutta, Browser Library oli jokaisessa testikohdassa suoritusajaltaan tehokkaampi.



Kuvio 5. Identtisen automaatiotestin vaiheiden suoritusajat Browser Librarya, sekä SeleniumLibrarya käyttäen.

Jos testien suoritusaika olisi tärkeää saada mahdollisimman pieneksi, olisi tämän perusteella kannattavaa suosia Browser Librarya selainpohjaisten automaatiotestien luomisessa SeleniumLibraryn sijaan. Tällä ei kuitenkaan ole merkittävää vaikutusta, jos automaatiotestien läpimenoaika ei ole pullonkaulana. Pääasia käytännössä on usein se,

että testit saadaan suoritettua automaation avulla ilman, että niihin tarvitsee käyttää ihmisvoimaa. Jos yhdenkään henkilön ei tarvitse käyttää omaa aikaansa automaatiotestien ajamisen seurantaan, ei sillä välttämättä ole merkitystä tulevatko testit automaation avulla suoritetuiksi 30 tai 60 minuutissa.

6.2.4 Automaatiotestien rinnakkaissuorittamisen vaikutus testien suoritusaikoihin

Yhtenä suurimpana testiautomaation etuna on mahdollisuus suorittaa useampaa testiä samanaikaisesti. Robot Frameworkille tähän on olemassa työkalu nimeltä Pabot. Pabotin avulla pystytään yhdellä koneella ajamaan testejä rinnakkain omissa prosesseissaan. Rinnakkaisajoon voidaan asettaa jokainen testisarja omaan prosessiinsa, tai yhden testisarjan testit omiin prosesseihin. (Robot Frameworkin dokumentaatio 2023.)

Tällä osa-alueella automaatio päihittää tehokkuudellaan manuaalisesti tehdyn testauksen täysin. Esimerkiksi tilanteessa, jossa suoritettavien testien määrä on 10 ja kuvitellaan, että ihminen suorittaisi jokaisen testin yhtä nopeasti kuin automaatio, voisi robotti suorittaa jokaista testistä samaan aikaan, jolloin automaation avulla saataisiin periaatteessa kaikki 10 testiä suoritettua samassa ajassa, jossa ihminen sai suoritettua pisimmän kymmenestä testistä. Testien rinnakkaisajoon liittyen tulee kuitenkin ottaa huomioon, että jokainen rinnakkaisajoon lisätty testi vie tietokoneen resursseja. Tästä syystä testejä ei käytännössä voi ajaa äärettömiä määriä saman aikaisesti rinnakkain.

Kuviosta 6. nähdään, kuinka kauan testien suorittaminen vei aikaa, kun testejä ajettiin rinnakkain 20 kappaletta samaan aikaan ja kun testit suoritettiin yksitellen. Rinnakkain ajettujen testien suoritusajaksi tuli 14 minuuttia, kun taas yksittäin ajettuina samojen testien suoritus kesti 87 minuuttia. Käytännössä tämä tarkoittaa sitä, että jos haluttaisiin ihmisvoimin suorittaa testit yhtä tehokkaasti, kuin mitä tässä tapauksessa rinnakkain saatiin suoritettua automaation avulla, vaatisi se 20 ihmistä suorittamaan testejä saman aikaisesti. Automaatiolla saadaan tässä tapauksessa suoritettua testit samalla tehokkuudella, mikä vaatisi 20 ihmisen työpanoksen 14 minuutin ajaksi ilman yhdenkään ihmisen työpanosta. 20 ihmisen työpanos 14 minuutin ajalta tarkoittaa käytännössä 280 minuuttia (20*14=280), mikä taas vastaa 4,67 tunnin (280/60=4,67) aikaa.

Voidaan siis ajatella, että joka kerta, kun testit suoritetaan automaation avulla rinnakkain, säästetään 4,67 tunnin edestä työaikaa.

Kuvio 6. Testien suoritusaika, kun 20 testiä ajettiin rinnakkain samaan aikaan vs. yksittäin ajettuina.

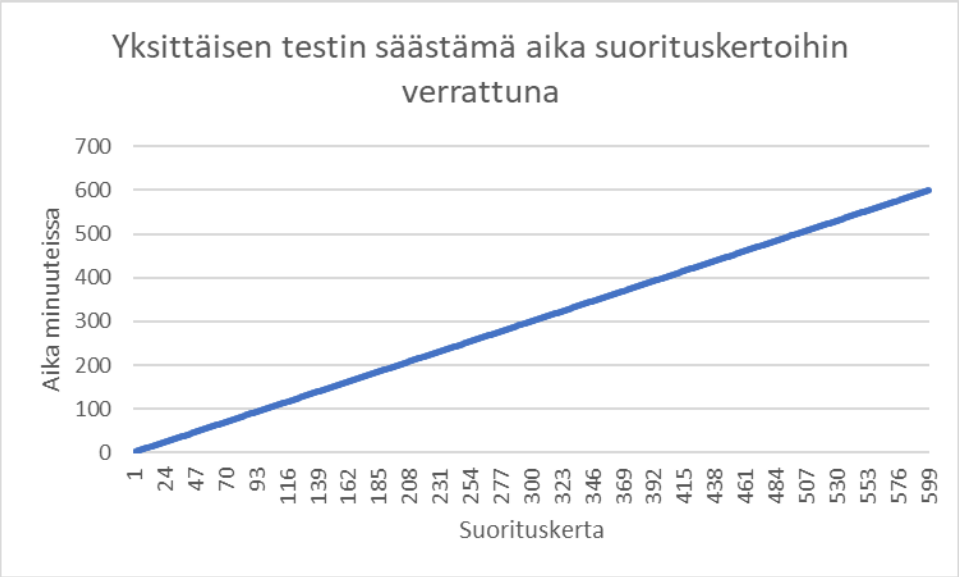
6.3 Testien toteuttamiseen ja ylläpitoon kuluvan ajan vaikutus testiautomaation ajansäästöön

Testien suoritusaikojen lisäksi tulee automaation ajansäästössä ottaa huomioon myös automaation luomiseen kuluva aika. Koska käytännössä jokainen automatisoitava testi eroaa toisistaan jollakin tavalla, on mahdotonta laskea aikaa, joka testiautomaation luomiseen, tai ylläpitoon tulee kulumaan. Voidaan kuitenkin periaate tasolla laskea, milloin ja missä tapauksissa automaation luomiseen ja ylläpitoon käytettävä aika tulisi maksamaan itsensä takaisin.

6.3.1 Automaatiotestien toteuttamisen ja ylläpitoon käytetyn ajan vaikutukset testiautomaatiolla säästettyyn aikaan

Kuviossa 7. on esiteltynä aika, jonka yksi minuutin kestävä automaatiotesti olisi säästänyt tietyn määrän suorituskertojen jälkeen verrattuna siihen, että testi olisi tehty manuaalisesti samassa ajassa. Jos kuvan 8. tapauksessa testin tekoon olisi käytetty 10

tuntia aikaa, maksaisi se itsensä takaisin 600 suorituskerran jälkeen. Tässä tapauksessa tulisi testi suorittaa vähintään 600 kertaa, jotta sen tekemiseen käytetty aika olisi taloudellisesti kannattavaa. Käytännössä testien automatisointi tulee siis maksamaan itsensä takaisin automaation luomiseen käytetyn ajan takia vasta useiden suorituskertojen jälkeen. Tästä johtuen tulisi ennen automaation toteuttamista selvittää kuinka usein testejä tulisi suorittaa, sekä kuinka pitkään niitä tulisi suorittaa. Nämä kaksi tekijää määrittävät käytännössä ylläpidon ohella sen, onko automaatio kyseisessä tilanteessa taloudellisesti kannattavaa.



Kuvio 7. Testiautomaation säästämä aika suorituskertoihin verrattuna.

6.3.2 Automaation ylläpitoon kuluvan ajan vaikutus testiautomaatiolla säästettyyn aikaan

Lopuksi automaation ajansäästöä laskettaessa tulee ottaa huomioon automaation ylläpitoon kuluva aika. Niin kauan, kuin automaation avulla testattavia asioita kehitetään, tarvitsee niille luotuja testisuunnitelmia ja niiden pohjalta luotuja automaatiotestejä päivittää uusien ohjelmapäivitysten mukaiseksi. Samoin kuin automaation luomiseen, myöskään automaation ylläpitoon kuluvaa aikaa ei käytännössä voi laskea ennalta. Ylläpitoon käytettävä aika voidaan kuitenkin saada mahdollisimman pieneksi, kun automaatio toteutetaan laadukkaasti (Grasberger 2020).

Kuvassa 8. havainnollistetaan, kuinka monta suorituskertaa yhden minuutin kestävä automaatiotesti, jonka ylläpitoon on varattu 15 minuuttia sataa testin suorituskertaa kohden, vaatii säästääkseen 10 tunnin edestä aikaa. Tässä tapauksessa testi vaatisi yli 700 suorituskertaa, ennen kuin 10 tuntia aikaa olisi sen avulla saatu säästettyä.

Kuvio 8. Yksittäisen minuutin kestävän automaatiotestin ajansäästö verrattuna suorituskertoihin, kun ylläpitoon kuluvaksi ajaksi on asetettu 15 minuuttia 100 testin suorituskertaa kohden.

7 LOPPUTULOS JA POHDINTA

Testien suorittamiseen kuluvan ajan mittausten tuloksena saatiin pätevä katsaus regressiotestauksen automatisoinnin säästämästä ajasta. Tulosten pohjalta saatiin myös hyvä katsaus siitä, kuinka paljon web-käyttöliittymän regressiotestauksen automatisointi käytännössä säästää opinnäytetyössä vertailukohteena käytettävien testien osalta aikaa, ja miten Robot Frameworkilla automatisoiduissa testeissä käytettävät kirjastot eroavat tehokkuudeltaan. Tämän lisäksi mallinnettiin testien automaation toteuttamiseen ja ylläpitoon kuluvan ajan vaikutuksia automaation taloudellisuuteen. Tulosten

pohjalta pystytään jo automaation suunnitteluvaiheessa paremmin määrittelemään, onko automaation toteuttaminen kyseisessä tilanteessa kannattavaa.

Testien automatisointiin käytettyjen kirjastojen eroista olin lukenut ennenkin ja tiesin, että erot voivat olla suhteellisen suuria, mutta olin silti yllättynyt siitä, kuinka paljon saman testin suorittamiseen kuluvassa ajassa oli eroja SeleniumLibraryn ja Browser Libraryn välillä. Tämän tiedon pohjalta pystytään tulevaisuudessa paremmin arvioimaan aikaa, jonka saadaan säästettyä, jos yksittäinen testi refaktoroidaan käyttämään SeleniumLibraryn sijasta Browser Librarya.

Oli hyvin mielenkiintoista myös nähdä, kuinka paljon automaatio käytännössä säästi aikaa manuaaliseen testaukseen verrattuna. Kun automaatio on luotu ja testit ajetaan automaattisesti, ei tule aina mietittyä, kuinka paljon se säästää itseään toistavalta työltä. Tässä opinnäytetyössä vertailukohteeksi otetut testit olivat kuitenkin vain yhdeltä tuotteelta, mutta testien manuaaliseen suorittamiseen kului silti huomattava määrä aikaa ja työ oli hyvin puuduttavaa. Tämä antoi hyvää perspektiiviä sille, kuinka tärkeässä roolissa automaatio nykyaikana on testauksen työkaluna.

LÄHTEET

Browser Library (2023). Introduction. https://robotframework-browser.org/#introduction

Da Silva, M. (2022). Pros and cons of automated testing. Uilicious. https://uilicious.com/blog/pros-cons-automated-testing/

Grasberger, M. (25.08.2020). 4 ways to minimize test automation maintenance. TechTarget. https://www.techtarget.com/searchsoftwarequality/tip/4-ways-to-minimize-test-automation-maintenance

Hedge, A. (02.09.2022). How to create Test Automation Strategy: Best Practices. BrowerStack. https://www.browserstack.com/guide/how-to-create-test-automation-strategy

Jotform. (28.03.2023). Why is project planning important? https://www.jotform.com/blog/importance-of-project-planning/

Kestikievari, M. (n.d). Yksi testausautomaatio, kiitos. Itse wiki. https://www.itewiki.fi/p/yksi-testausautomaatio-kiitos

Kushnir, A. (05.05.2022). Software Development Planning: Why It Is Important. Bamboo Agile. https://bambooagile.eu/insights/software-development-planning/

Klärck, P. (2006). Data-Driven and Keyword-Driven Test Automation Frameworks. [Pro gradu -tutkielma, Teknillinen korkeakoulu]. Eliga. http://eliga.fi/writings.html

Mulders, M. (2019). Test Automation Benefits: 12 Reasons to Automate in 2020. Testim.io. https://www.testim.io/blog/test-automation-benefits/

Patt, A. (27.05.2021). The Pros and Cons of Automated Testing. Test.io. https://test.io/resources/blog/the-pros-and-cons-of-automated-testing

Raeburn, A (26.10.2022). Context switching is killing your productivity. https://asana.com/resources/context-switching

Reverchuk. L. (13.01.2023). COMMERCIAL VS OPEN SOURCE SOFTWARE: BENEFITS AND DRAWBACKS. https://echoua.com/commercial-vs-open-source-software-the-benefits-and-drawbacks/

Robot Framework (2023). Introduction. https://robotframework.org/

Robot Framework dokumentaatio. (2023). Running tests in parallel. https://docs.robotframework.org/docs/parallel

Safonova, E. (28.03.2022). Automation Testing: Pros & Cons. Sumatosoft. https://sumatosoft.com/blog/automation-testing-pros-cons

SeleniumLibrary (2023). Introduction. https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html

Shvets, D. (n.d). Test automation framework. Mindk. https://www.mindk.com/blog/test-automation-framework/