Deltagon Group Oy:n testausautomaation kehittäminen

Risto Hirvo

2019 Laurea

Laurea-ammattikorkeakoulu

Deltagon Group Oy:n testausautomaation kehittäminen

Risto Hirvo
Tradenomi, tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Toukokuu, 20192019

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tradenomi, tietojenkäsittelyn koulutusohjelma
Tietojenkäsittelyn koulutusohjelma
Tradenomi (AMK)

Risto Hirvo Risto Hirvo

Deltagon Group Oy:n testausautomaation kehittäminen

Vuosi 2019 2019 Sivumäärä 28 + 3 liitesivua

Tämän opinnäytetyön tavoitteena oli Deltagon Group Oy:n testausautomaation kehittäminen. Deltagon Group Oy:ssä testausautomaatiota käytetään osana ohjelmistotestausta. Opinnäytetyön tavoitteena oli luoda tehokas ja luotettava testausautomaatiotyökalu kehittämällä olemassa olevaa testausautomaatioympäristöä ja järjestämällä koulutus testausautomaatiosta Deltagon Group Oy:n kehitystiimin jäsenille. Testausautomaatioympäristön jatkokehittämisen tarkoituksena oli päivittää aikaisemmat automaattiset testit uuden ohjelmistopäivityksen kanssa yhteensopivaksi ja korjata olemassa olevia luotettavuusongelmia. Kehitystiimin jäsenille pidettävän testausautomaatiokoulutuksen tarkoituksena oli laajentaa testausautomaatiota koskemaan koko kehitystiimiä, jotta jatkossa jokaisella olisi mahdollisuus kirjoittaa automatisoituja testejä uusille tuotteiden ominaisuuksille tai ohjelmistomuutoksille.

Opinnäytetyö on toiminnallinen koostuen teoreettisesta raporttiosuudesta, testausautomaation kehittämisprosessin ja kehitystiimin jäsenille pidetystä testausautomaatiokoulutuksen kuvauksesta. Teoreettisessa raporttiosuudessa käsitellään ohjelmistotestausta, testausautomaatiota, Robot Frameworkia ja Selenium2Librarya. Testausautomaation kehittämisprosessissa on hyödynnetty konstruktiivista tutkimusmenetelmää (eng. Design Science Research). Testausautomaatiokoulutuksen kuvauksessa käsitellään koulutuksen yleiset käytänteet sekä koulutuksen sisältöä.

Opinnäytetyön tuloksena syntyi tehokas ja luotettava testausautomaatiotyökalu, sekä paranneltu testausautomaation kehittämisenprosessi. Kehitystyöstä ja koulutuksesta saatu palaute oli positiivista ja toi esille useita jatkokehitysmahdollisuuksia testausautomaatioprosessien kehittämisestä entistä paremmaksi.

Asiasanat: Robot Framework, testausautomaatio, ohjelmistotestaus

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

Risto HirvoRisto Hirvo

Development of test automation for Deltagon Group Oy

Year 2019 2019 Pages 28 + 3 pages in appendices

The purpose of this thesis was to develop test automation for Deltagon Group Oy. In Deltagon Group test automation is used as part of the software testing process. The main goal for this thesis was to achieve efficient and reliable test automation by further developing the existing test automation environment and providing test automation training for Deltagon Group's development team. The purpose of further developing the existing test automation environment was to update the current automated tests to be compatible with new software releases and fix old unreliability problems. The purpose of test automation training was to involve the whole development team in test automation development.

The thesis is a functional thesis and consists of three sections: a theoretical section, a description of the test automation development process and a description of test automation training for the development team. The theoretical section consists of theory on software testing, test automation, Robot Framework and Selenium2Library. The test automation development section of the thesis contains a step-by-step description of the test automation development process implementing Design Science Research method. The test automation training section of the thesis contains a description of the training session.

The result of the thesis was an efficient and reliable test automation tool with improved development process. The feedback from Deltagon Group's development team on test automation development and test automation training was positive and provided interesting future development ideas which could further enhance the company's test automation processes.

Keywords: Robot Framework, test automation, software testing

Sisällysluettelo

1 Johdanto

Suoritin opintoihin liittyvän työharjoitteluni vuoden 2018 alussa Deltagon Group Oy:ssä. Tehtäväni harjoittelun aikana oli kehittää yrityksen tuotteille käyttöliittymän testausautomaatiota. Deltagon Groupin testausautomaatio kehitettiin käyttäen Robot Framework testiautomaatioviitekehystä, joka hyödyntää Selenium2library verkkotestaus -kirjastoa. Työharjoittelun aikana kirjoitin testejä yrityksen neljälle eri tuotteelle: Sec@GW salattuun sähköpostiratkaisuun, secureForms turvalliseen lomakeratkaisuun, collabRoom turvalliselle tiedonjako alustalle ja secSigned sähköiseen allekirjoitusratkaisuun.

Nykyisin työskentelen Deltagon Group Oy:lle kehitystiimin web-developerina. Syksyllä 2018 Deltagon päätti julkaista tuotteista uudet versiot vuoden 2019 ensimmäisellä vuosineljänneksellä. Tarkoittaen sitä, että kaikki edellisen julkaisun jälkeen tehdyt parannukset, korjaukset ja lisäykset tuotteisiin julkaistaisiin asiakkaiden käyttöön. Tähän liittyen ajankohtaiseksi nousi myös testausautomaation kehitys yhteensopivaksi tulevien ohjelmistoverisoiden kanssa, jotta olisi mahdollista testata automaattisesti myös uusia ominaisuuksia ja muita muutoksia ohjelmistossa. Samalla esimieheni ilmoitti kehityskeskustelussa, että muille kehitystiimin jäsenille pitäisi järjestää koulutus siitä, miten testausautomaatiotestejä kirjoitetaan.

Opinnäytetyön tarkoituksena on testausautomaation kehittäminen Deltagon Group Oy:lle. Deltagon Group Oy on suomalainen tietoturva-alan yritys. Deltagon Group Oy tuottaa monipuolisia ohjelmistoratkaisuja, joiden keskiössä on tietoturvallisuus. Deltagon Group toimii osana Suomen Erillisverkot -konsernia.

Opinnäytetyön aiheeksi valikoitui testausautomaation kehittäminen, sillä hankeyritys on julkaisemassa uuden päivitetyn version tuotteistaan. Tämän lisäksi yrityksen testausautomaation kehitystä käydään läpi kehitystiimin jäsenille pidettävässä koulutuksessa. Opinnäytetyöprosessi käynnistyi syksyllä 2018. Opinnäytetyö on toiminnallinen ja koostuu teoreettisesta raporttiosuudesta, sekä testausautomaation kehittämisprosessin kuvauksesta.

2 Lähtökohdat ja tarve

2.1 Deltagon Group Oy

Kehitystyö tehdään Deltagon Group Oy:lle. Deltagon Group Oy on vuonna 1999 perustettu suomalainen tietoturva-alan yritys, joka toimii osana Suomen Erillisverkot -konsernia. Yrityksellä on toimipisteet Espoossa, Oslossa ja Tukholmassa. Yrityksen arvoja ovat yhtenäisyys, ketterä kehitys, tuottavuus sekä asiantuntijuus. Yrityksen tuotteita ovat Sec@GW -

suojattusähköpostiratkaisu, secureForms -turvallinen lomakealusta, collabRoom -turvallinen tiedostonjako ja viestintäratkaisu ja secSigned -sähköinen allekirjoitusratkaisu. (Deltagon Group Oy, 2019.)

2.2 Opinnäytetyön tavoite

Opinnäytetyön tavoitteena on luotettavan ja toimivan testausautomaation kehittäminen Deltagon Group Oy:lle. Kehittämistyön tarkoitus on jatkokehittää yrityksen testausautomaatiosta yhteensopiva tulevan ohjelmistopäivityksen kanssa ja järjestää koulutus kehitystiimin jäsenille testausautomaation kehittämisestä. Koska nykyiset automaattiset testit ovat suunniteltu testaamaan nykyisten tuotteiden versioita, sitä ei voida luotettavasti käyttää tulevan version testaukseen. Kehitystiimin koulutus puolestaan monipuolistaa ja nopeuttaa testausautomaation kehitystä kasvattamalla yrityksen testausautomaation kehittäjien määrää.

Testausautomaation jatkokehityksen tarkoituksena on kehittää nykyistä testausautomaatiota edelleen yhteensopivaksi tulevien versiopäivitysten kanssa. Aikaisempiin testeihin on korjattava tuotteiden ominaisuuksien muutokset, sillä ulkoasu ja/tai toiminnalliset muutokset tuotteiden eri ominaisuuksiin ovat mitä todennäköisimmin rikkoneet aikaisemmin tehtyjä testejä. Lisäksi automaattisiin testeihin on myös lisättävä testitapauksia, jotka testaavat tuotteiden uudet ominaisuudet ja muut lisäykset. Opinnäytetyö rajataan käyttöliittymän testaukseen loppukäyttäjän näkökulmasta, sekä aikaisempaan testausautomaatiotyökaluun tehtäviin tarvittaviin muutoksiin ja uusien ominaisuuksien testien lisäämiseen. Tuotteista on poistunut tai muutettu tiettyjä ominaisuuksia, joita aikaisemmat testit käyttivät. Tämän vuoksi kehitystyön yksi tavoite on selvittää, mikä on muuttunut, jotta testit voidaan muuttaa toimimaan oikein. Nykyisissä testeissä on myös muutamia vikoja ja/tai puutteita. Niitä ajaessa on tullut vastaan virheitä, jotka johtuvat testeistä, eivätkä tuotteista. Tämän tyyppiset virheet tekevät testeistä epävarmoja ja tuotteista saattaa jäädä useita ominaisuuksia testaamatta. Sen vuoksi tarkoituksena onkin muokata ja lisätä testejä, jotka varmistavat, etteivät virheet johtuisi itse testeistä, vaan tuotteesta löytyneestä virheestä.

Testausautomaatiokoulutuksen tarkoitus on lisätä yrityksessä testausautomaation kehittäjien määrää, jolloin testien kehittäminen tehostuu ja yrityksen sisäinen testausautomaatio-osaaminen laajenee. Koulutuksen keskeinen tavoite on saada jokainen kehitystiimin jäsen kirjoittamaan omat testausautomaatiotestit tekemilleen muutoksille yrityksen tuotteisiin. Myös aikaa säästyy, kun jokainen kehitystiimin jäsen pystyy kirjoittamaan omat automaattiset testit tekemilleen koodimuutoksille, eikä toisen kehittäjän tarvitse tutkia koodimuutoksen toiminnallisuutta testien suunnittelua varten. Koodimuutoksen alkuperäinen tekijä tuntee tekemänsä muutoksen muita paremmin, jonka vuoksi hän pystyy kirjoittamaan perinpohjaisempia automaattisia testejä.

Oma oppimistavoite opinnäytetyön aikana on kehittää omaa testausautomaation kehitystaitoja. Olen työharjoittelun jälkeen päässyt itse kehittämään yrityksen eri tuotteita ja olen oppinut syvällisemmin, miten tuotteet toimivat ja miten niiden pitäisi toimia. Tavoitteenani onkin oppia kehittämään laadukkaampia testejä.

3 Ohjelmistotestaus

Ohjelmistotestaus on olennainen osa ohjelmistokehitystä. Ohjelmistotestauksen tarkoitus ohjelmistokehityksessä on varmistaa kehitettävän ohjelmiston toiminnallisuus. Ohjelmistotestaus saattaa eri ohjelmistoissa painottua eri alueisiin, kuten ohjelmiston käyttäjäystävällisyyteen, ulkoasuun tai toimintavarmuuteen. Keskeinen tarkoitus testaamisessa on kuitenkin mahdollisten virheiden ja vikojen löytäminen. Jos ohjelmistokehityksessä ei ole toteutettu testausta tai testaus on puutteellista, voivat seuraukset johtaa suuriin taloudellisiin ja/tai maineen menetyksiin. Ohjelmistotestauksessa on myös hyvä huomioida, että testaus on jatkuva prosessi ja sen pitäisi kytkeytyä jokaiseen ohjelmistokehityksen eri vaiheeseen. (Kasurinen, 2013, 10-19.)

Ohjelmistotestaus voidaan jakaa kolmeen testaustasoon, eli yksikkötestaus, integraatiotestaus, järjestelmätestaus. Näiden jälkeen voidaan lopuksi siirtyä hyväksymistestaukseen. (Kasurinen 2013, 50.) Yksikkötestauksella tarkoitetaan vaihetta, jossa testataan kehittäjän tekemää yksittäistä komponenttia, mikä ei ole vielä kytköksissä ohjelmiston muihin komponentteihin (Kasurinen 2013, 51). Integraatiotestaus tarkoittaa testausvaihetta, kun yksittäistä komponenttia ollaan liittämässä osaksi järjestelmää. Toisin sanoen kyseisessä vaiheessa testataan yksittäisten komponenttien toiminnallisuutta keskenään. (Kasurinen 2013, 54.) Järjestelmätestaus kattaa koko järjestelmän testauksen kokonaisuutena. Tässä vaiheessa vuorossa on siis tuotteen laajamittainen kokonaisuuden testaaminen. (Kasurinen 2013, 56.) Hyväksymistestauksen tarkoitus on osoittaa, että lopullinen tuote on niiden vaatimusten mukainen, jotka sille on vaatimusmäärittelyssä asetettu. Tässä vaiheessa ei enää suoranaisesti testata etsien virheitä, vaan kyseessä on enemmänkin viimeinen tarkistusvaihe ennen tuotteen siirtymistä asiakkaan käyttöön. (Kasurinen 2013, 57.)

3.1 Testausautomaatio

Testausautomaatiolla tarkoitetaan automaattisia testejä, jotka on kehitetty erillisellä testausohjelmistolla. Testausautomaatiota voidaan käyttää osana ohjelmistotestausprosessia. Automaattisilla testeillä ei kuitenkaan voida korvata manuaalista testausta, vaan automaattisia testejä voidaan käyttää manuaalisen testauksen tukena. (CrossBrowserTesting 2019.)

Automaattinen testaus ei itsestään riitä ohjelmiston tarvittavan toiminallisuuden testaamiseen, koska automaattisista testeistä puuttuu ihmisen näkemys ja tunteet. Esimerkiksi käyttöliittymän ulkoasun testaaminen, mihin tarvitaan usein toisen ihmisen näkemys siitä, että miltä ohjelman käyttäminen tuntuu tai näyttää. (Rachel 2017; Bartlett 2017.) Testauksessa voidaan käyttää automaattisia testejä esimerkiksi testatakseen suurempia kokonaisuuksia tai helpottaa toistuvien testien testausta (CrossBrowserTesting 2019). Automaattinen testaus soveltuu hyvin regressiotestaukseen ohjelmistokehityksessä, eli laajempaan testaukseen, jossa varmistetaan uuden ominaisuuden tai muutoksen yhteensopivuutta aiempien ominaisuuksien kanssa (Rachel 2017; Bartlett 2017). Samalla voidaan myös testata uuden ominaisuuden toimivuus itsestään, jos siihen on lisätty tarvittavat testit.

Deltagon Groupissa käytetäänkin testausautomaatiota tämän tyyppisesti, jossa automaattisia testejä ajetaan viikoittain. Näin varmistetaan, että mahdollisten virhekorjausten tai muiden kriittisten muutosten yhteydessä ohjelmisto ei ole menettänyt toimivuuttaan. Tämän opinnäytetyön kehittämistyö tuleekin olemaan osa laajempaa yrityksen testausautomaatiotyökalun edelleen kehittämistä, jossa tullaan lisäämään uusia testejä uusia ominaisuuksia varten ja tarvittaessa muokkaamaan aikaisempia testejä.

3.2 Robot Framework

Robot Framework on avoimen lähdekoodin, Python pohjainen testausautomaatioviitekehys. Robot Frameworkin on kehittänyt Pekka Klärck vuonna 2005 osana diplomityötään. (Bisht 2013, 26.) Robot Frameworkiin on sisäänrakennettu useita eri kirjastoja, joiden ominaisuuksia voidaan käyttää kutsumalla testiskripteissä niitä ennalta määritellyillä avainsanoilla (Bisht 2013, 66). Tässä opinnäytetyössä keskitytään kuitenkin vain Robot Frameworkin selainpohjaisen käyttöliittymän automaattiseen testaamisen.

Robot Frameworkin selainpohjaiseen käyttöliittymän testaamiseen käytetään Selenium selainpohjaista testausautomaatiotyökalua. Selenium työkalulla voidaan testata monipuolisesti verkkosivun eri ominaisuuksia. Tätä varten Robot Frameworkiin tarvitaan Robot Framework Selenium -kirjasto, jolla voidaan liittää Selenium WebDriver Robot Frameworkiin. Siten voidaan käyttää Seleniumin omia komentoja suoraan Robot Frameworkin testitiedostoista. (Bisht 2013, 77.)

3.3 Selenium2Library/SeleniumLibrary

Selenium2Library (3.0 version jälkeen SeleniumLibrary) on Robot Frameworkin kirjasto, joka käyttää Selenium testausautomaatiotyökalua sisäisesti (GitHub 2019b). Selenium2Library sisältää suurimman osan avainsanoista, mitä Robot Frameworkin selainpohjaiseen testaukseen tarvitaan. Avainsanat, jotka löytyvät Selenium2Library -kirjastosta tarvitsevat useimmiten eri argumentteja toimiakseen, kuten paikanninarvoja (locator). Näillä määritellään, mihin verkkosivun elementtiin pyritään avainsanalla vaikuttamaan. Kyseisiä paikanninarvoja voidaan määritellä avainsanoihin HTML-attribuuteilla, CSS-valitsimilla tai XPath määritelmillä. (Selenium2Library 2019.)

4 Testien kirjoittaminen Robot Framework testausautomaatioviitekehykseen

Robot Frameworkin testit koostuvat avainsanaosiosta (*** Keywords ***), asetusosiosta (*** Settings ***), muuttujatosiosta (*** Variables ***) ja testitapausosiosta (*** Test Cases ***). (GitHub 2019a.) Robot Frameworkin testien havainnollistamiseksi tein esimerkki testin, jossa näkyy tässä osiossa läpikäytävät Robot Frameworkin testien kirjoittamisen osa-alueet (kuva 1). Avainsanoilla rakennetaan testien toiminnallisuus. Avainsanat voivat olla eri kirjastojen valmiiksi määritellyt avainsanat tai käyttäjän itse kirjoittamat avainsanat. Avainsanoja käyttääkseen pitää joihinkin avainsanoihin määritellä argumentteja, kuten HTML-elementtien paikanninarvoja, syötteitä, muuttujia tai muita avainsanoja. On myös avainsanoja, johon ei tarvitse antaa argumentteja toimiakseen kuten ”Close Browser” avainsana. (GitHub 2019a.)

```
*** Settings ***          Asetusosio
Library                   SeleniumLibrary

*** Variables ***         Muuttujat osio
${LOGIN URL}              example.com
${BROWSER}                Chrome

*** Test Cases ***        Testitapausosio
Testi
    Opens doto-list and hides it
    Adds new item todo-list without value

*** Keywords ***          Avainsanaosio
Opens doto-list and hides it
    Open Browser  ${LOGIN URL}  ${BROWSER}
    Element Should Be Visible  xpath=//input[@name="newitem" and @id="newitem"]
    Click Button  button
    Element Should Not Be Visible  xpath=//input[@name="newitem" and @id="newitem"]
    [Teardown]    Close Browser
```

Kuva 1 Robot Framework testin rakenne

Kirjastojen määrittelemät avainsanat sisältävät monipuolisen valikoiman valmiita avainsanoja, joita voi käyttää rakentaakseen testitapauksia. Eri avainsanakirjastoja on mahdollista

ottaa käyttöön lisäämällä asetusosioon ”Library” -avainsana ja käytettävän kirjaston nimi. Eri kirjastoista ja kirjastojen käytettävissä olevista avainsanoista löytyy kattava dokumentaatio Robot Frameworkin verkkosivuilta. (GitHub 2019a.) Esimerkkitestissä (Kuva 2) näkyy, miten kirjaston avainsanoja on käytetty asetusosiossa, tuodakseen kirjaston ”SeleniumLibrary” testin käyttöön ja miten kirjaston avainsanoja on käytetty oman avainsanan luomiseksi.

```
*** Settings ***
Library           SeleniumLibrary

*** Variables ***
${LOGIN URL}      example.com
${BROWSER}        Chrome

*** Test Cases ***
Testi
    Opens doto-list and hides it
    Adds new item todo-list without value

*** Keywords ***
Opens doto-list and hides it
    Open Browser  ${LOGIN URL}  ${BROWSER}
    Element Should Be Visible  xpath=//input[@name="newitem" and @id="newitem"]
    Click Button  button
    Element Should Not Be Visible  xpath=//input[@name="newitem" and @id="newitem"]
    [Teardown]    Close Browser
```

Kirjaston määrittelemät avainsanat

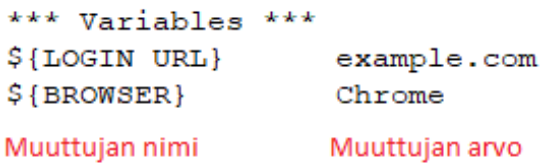
Kuva 2 Esimerkkitestin kirjaston määrittelemät avainsanat

Käyttäjän luoma avainsana

```
*** Keywords ***
Opens doto-list and hides it
    Open Browser  ${LOGIN URL}  ${BROWSER}
    Element Should Be Visible  xpath=//input[@name="newitem" and @id="newitem"]
    Click Button  button
    Element Should Not Be Visible  xpath=//input[@name="newitem" and @id="newitem"]
    [Teardown]    Close Browser
```

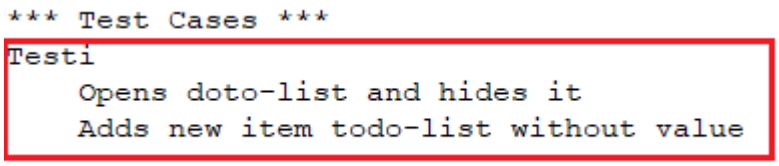
Kuva 3 Esimerkkitestin käyttäjän luoma avainsana

Käyttäjien itse luomat avainsanat puolestaan kirjoitetaan testitiedoston avainsanat osioon, käyttäjien itse kirjoittamiin avainsanoihin on mahdollista käyttää eri kirjastojen avainsanoja tai käyttäjän aikaisemmin luomia avainsanoja. Käyttäjän kirjoittaessa oman avainsanan, on ensimmäiseksi kirjoitettava avainsanalle nimi. Tällä nimellä sitä voidaan kutsua, joko testitapauksissa tai muissa avainsanoissa. (GitHub 2019a.) Esimerkkitestiin (Kuva 3) on käyttäjä luonut ” Opens doto-list and hides it” nimisen avainsanan, johon on koottuna eri avainsanoista koostuva toiminnallisuus. Asetusosioon voidaan lisätä testiin tarvittavat kirjastot, resurssitiedostot, mahdollinen dokumentaatio, testien aloitus/lopetus toiminnot ja niin edelleen. Jotta asetusosioon voidaan lisätä toiminnallisuuksia, pitää ensin kirjoittaa toiminnallisuuden avainsana ja sen jälkeen argumentti. (GitHub 2019a.) Esimerkkitestissä asetusosioon (Kuva 1) on lisätty kirjasto (Library) SeleniumLibrary.

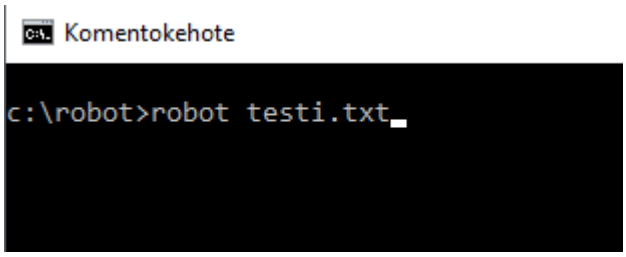


Kuva 4 Esimerkkitestin muuttujat osio

Muuttujat osioon voidaan määritellä avainsanoissa argumentteina käytettäviä muuttujia. Muuttujia määritellään muuttujat osioon antamalla sille ensiksi nimi muodossa ${NIMI} ja sitten arvo, kuten ${USER_NAME} testi. (GitHub 2019a.) Esimerkkitestin muuttujat osioon (Kuva 4) on määriteltynä ${LOGIN URL} ja ${BROWSER} -muuttujat, eli verkkosivunosoite, missä testattava kohde sijaitsee sekä selainohjelmisto, jolla testi suoritetaan. Testitapausosioon kootaan eri avainsanoista koostuvat testit, joilla testataan ominaisuuksia. Testitapaukset kirjoitetaan ensiksi kirjoittamalla testitapauksen nimi ja sen jälkeen avainsanoja. (GitHub 2019a.) Esimerkkitestiin (kuva 5) on tehty yksi testitapaus, joka on nimetty Testiksi, ja siinä käytetään kahta käyttäjän kirjoittamaa avainsanaa.



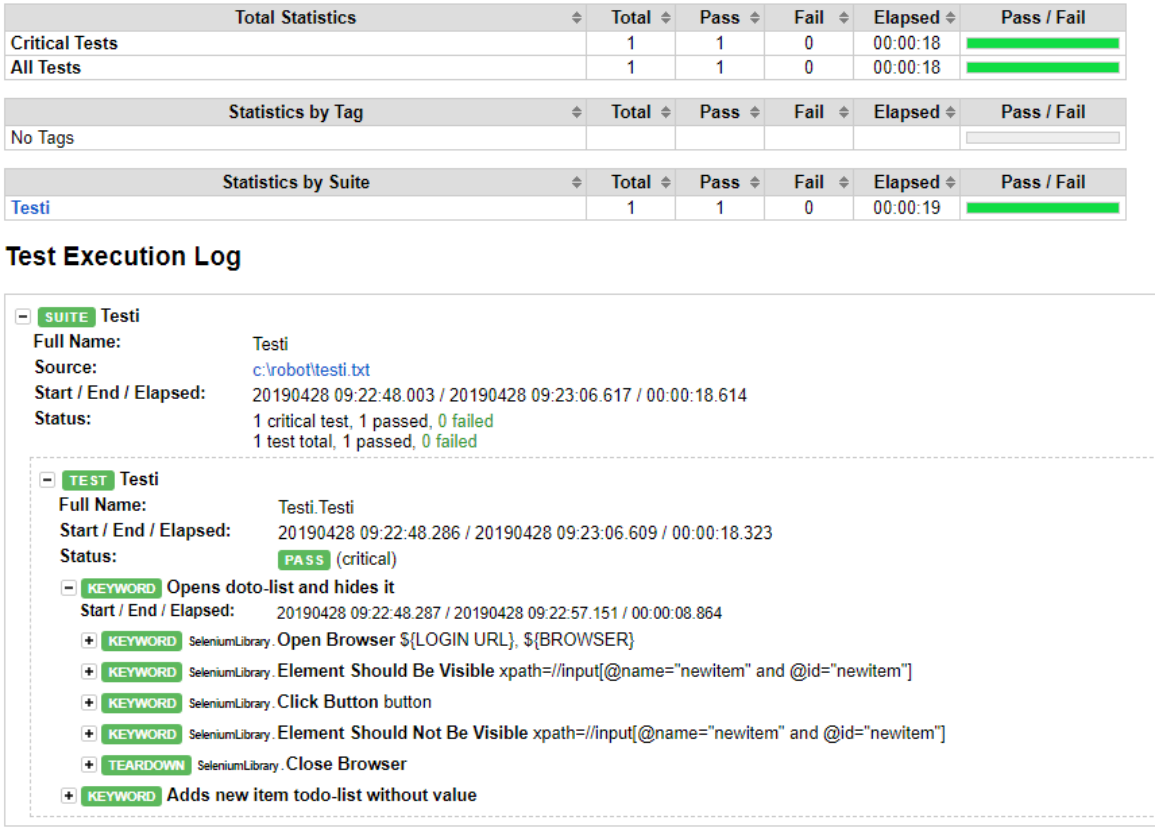
Kuva 5 Esimerkkitestin testitapaus



Kuva 6 Esimerkkitestin suorittaminen

Kun testi on kirjoitettu, voidaan suorittaa testi ajamalla komentokehotteessa komento ”robot testitiedostonnimi.tiedostonmuoto” (GitHub 2019a). Esimerkkitestin tapauksessa robot testi.txt (Kuva 6). Testien suorituksen aikana pystyy komentokehotteesta seuraamaan testien etenemistä. Kun testit on suoritettu Robot Framework luo testien ajettuaan ”log.html”, ”output.xml” ja ”report.html” -tiedostot kansioon, missä testit sijaitsevat. Näistä tiedostoista pystyy yksityiskohtaisesti seuraamaan, miten testit on ajettu ja mitä on missäkin vaiheessa tehty. (GitHub 2019a.) Esimerkkitestin tapauksessa käytetään log.html tiedostoa, jossa näkyy

onnistuneen testin tulokset (kuva 7). Jos testissä olisi virheitä (kuva 8) näkyy raportista, missä kohdassa testiä virhe on tapahtunut (”Adds new item todo-list without value” avainsana) sekä selitys, mistä virhe on johtunut (”UnexpectedAlertPresentException: Alert Text: None” virhe).



Kuva 7 Esimerkkitestin onnistunut log.html raportti

Kuva 8 Esimerkkitestin virheet

5 Toiminnallinen opinnäytetyö

Opinnäytetyö on toiminnallinen. Opinnäytetyö koostuu teoreettisesta raporttiosuudesta ja testausautomaation kehittämisprosessin kuvauksesta. Yksi yhtenäinen tekijä toiminnallisilla opinnäytetöillä on niiden tuloksena syntyvä tuotos, joka voi olla esimerkiksi portfolio tai jokin tapahtuma. (Vilkka & Airaksinen, 2003, 51.) Tämän opinnäytetyön tuotoksena syntyy testausautomaatiotyökalu ja siihen liittyvä koulutus.

Toiminnallisissa opinnäytetöissä tutkimusmenetelmien rooli jää vähäisemmäksi kuin tutkimuksellisissa opinnäytetöissä. Toiminnallisissa opinnäytetöissä tiedonkeruu kuitenkin tapahtuu samoin menetelmin, kuin tutkimuksellisissa opinnäytetöissä. (Vilkka & Airaksinen, 2003, 56-58, 63.) Laadullista ja määrällistä tutkimusmenetelmää on vaikeaa täysin erottaa toisistaan. Opinnäytetyössäni on osittain piirteitä molemmista tutkimusmenetelmistä. Toiminnallisessa opinnäytetyössä laadullinen tutkimusmenetelmä nousee esiin esimerkiksi asiantuntijoiden konsultaatioina ja määrällinen tutkimusmenetelmä nousee esiin ohjelmiston testausvaiheessa. Testausvaiheessa aikaisempia testejä ajaessa ohjelmisto nostaa esiin testaustuloksia, joiden perusteella testausautomaatiota kehitetään. Toiminnallisen opinnäytetyön raporttiosuuden tarkoituksena on, että lukija voi sen perusteella arvioida, miten onnistunut opinnäytetyö on. Opinnäytetyön raporttiosuudessa tuodaan esiin opinnäytetyöprosessia, jossa kuvataan sitä, mitä on tehty, miksi on tehty ja miten on tehty. (Vilkka & Airaksinen, 2003, 65.) Opinnäytetyön raporttiosuudessa arvioidaan myös toiminnallisen opinnäytetyön tuotosta eli testausautomaatiotyökalua.

5.1 Opinnäytetyöpäiväkirja

Opinnäytetyöpäiväkirja on dokumentti, johon voidaan kirjata opinnäytetyöprosessin eri vaiheita, ideoita, muistiinpanoja, lähes mitä vain. Opinnäytetyöpäiväkirja on henkilökohtainen ja dokumentointitavaksi kukin voi valita omaan käyttöön parhaiten soveltuvan tavan. Opinnäytetyö voi olla perinteisessä sanallisessa muodossa, auditiivisessa muodossa tai vaikkapa kuvallisessa muodossa. (Vilkka & Airaksinen 2003, 19.) Dokumentointitavasta riippumatta säännöllisesti dokumentoimalla opinnäytetyöpäiväkirjasta voi saada suurimman hyödyn. (Vilkka & Airaksinen 2003, 22.)

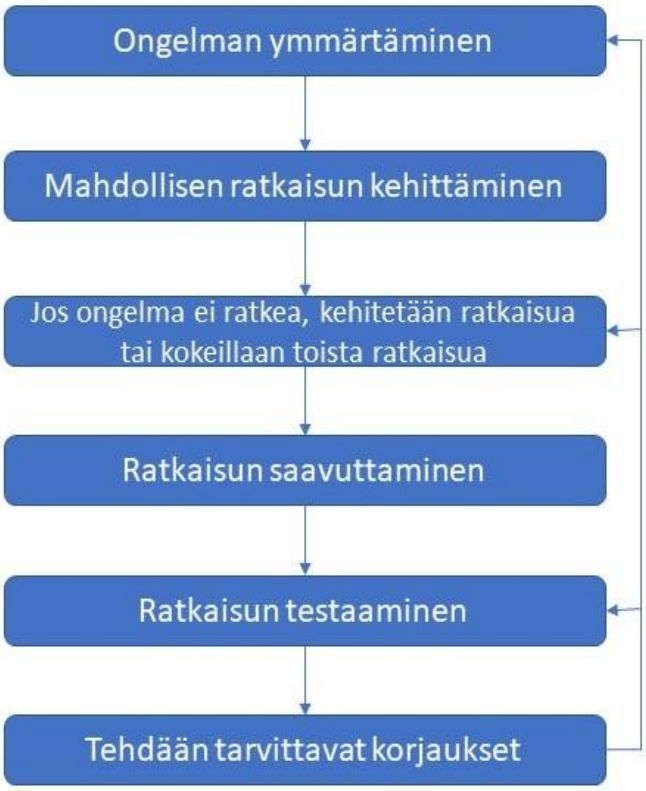
Opinnäytetyöprosessia aloittaessani päätin kirjoittaa opinnäytetyöpäiväkirjaa, joka toimisi dokumentointivälineenä, sekä muistin tukena opinnäytetyöprosessin edetessä. Opinnäytetyöpäiväkirjaan dokumentoin muun muassa esimiehen konsultoinnit; kirjasin säännöllisesti ylös vaiheittain, mitä olen tehnyt; esiin nousseet huomiot sekä omaa pohdintaa.

Opinnäytetyöpäiväkirjaa voin siten hyödyntää myös opinnäytetyön arvioinnissa, sillä opinnäytetyöpäiväkirjan avulla on mahdollista palata opinnäytetyön eri vaiheisiin (Vilkka & Airaksinen 2003, 19-22).

6 Testausautomaatiotyökalun kehitysprosessin kuvaus

Itse opinnäytetyön tuotoksena tulee olemaan testausautomaation kehitys. Tätä kehitän hyödyntäen konstruktiivista tutkimusmenetelmää, jossa käytän tiedonkeruumenetelminä esimiehen konsultointeja, testituloksia ja niiden läpikäymistä sekä tutkimista. Esimiehen konsultointia tarvitsin saadakseni tietoa muutoksista eri tuotteisiin liittyen ja tarvittaessa neuvoa mahdollisiin ongelmatilanteisiin. Testituloksia tutkimalla sain arvokasta tietoa siitä, mikä testeissä tai testattavassa ominaisuudessa toimii tai ei toimi. Sen perusteella suoritin tarvittavat muutokset ja/tai korjaukset. Edellä mainittujen tiedonkeruu menetelmien avulla pystyin kehittämään luotettavan ja toimintavarman testausautomaatiotyökalun.

Opinnäytetyön automaatiotyökalun kehittämisprosessi toteutettiin noudattamalla konstruktiivista tutkimusmenetelmää. Konstruktiivinen tutkimus (eng. Design Science Research) on tutkimus menetelmä, jossa tutkimuksen kohteena on jokin ihmisen luoma keinotekoinen asia (Dresch, Lacerda & Antunes Jr. 2015, 47). Konstruktiivisen tutkimusmenetelmän keskeinen tarkoitus on ratkaista tutkittavan kohteen ongelma (Dresch ym. 2015, 68). Konstruktiivisessa tutkimuksessa on useita eri menetelmiä ja keinoja löytää ratkaisu tutkittavaan kohteeseen. Tässä opinnäytetyössä keskiössä on Bungen kehittämä menetelmä, joka soveltuu parhaiten opinnäytetyön kehitystyön kuvaamiseen (Dresch ym. 2015, 72). Bungen kehittämän tutkimusmenetelmän keskeinen tarkoitus on mahdollistaa se, että tutkijan on mahdollista löytää tutkittavasta kohteesta tiettyjä ilmiöitä ja kehittää sitä. Bungen kehittämä menetelmä etenee kuudessa eri vaiheessa, jotka ovat: ongelman ymmärtäminen; mahdollisen ratkaisun kehittäminen; mikäli ongelma ei ratkea, kehitetään uutta ratkaisua; ratkaisun saavuttaminen; ratkaisun testaaminen ja lopuksi mahdollisten korjausten tekeminen. (Dresch ym. 2015, 72-73.) Kehittämisprosessista tehty kuvio on kuvattuna seuraavalla sivulla.



Kuva 9: Testausautomaatiotyökalun kehittämisprosessi Bungen konstruktiivista mallia mukaillen (Dresch, Lacerda & Antunes Jr. 2015, 73).

Bungen menetelmän ensimmäisen vaiheen tarkoituksena on ongelman toteamisen jälkeen oppia ymmärtämään, mikä tutkittavan kohteen ongelma on (Dresch ym. 2015, 72). Testausautomaation kehitystyössä todettiin ongelmaksi se, että aikaisemmat testit eivät olleet enää toiminnallisia uusien tuoteversioiden kanssa. Tämän lisäksi testeistä puuttui täysin uusien ominaisuuksien testaaminen. Tämä voitiin todeta ajamalla vanhat testit uusia versioita vasten. Testien tuloksia analysoimalla huomattiin, että suurin osa testeistä palauttivat virheen. Esiin nousseet virheet eivät johtuneet tuotteiden virheistä, vaan uusista muutoksista ja virheellisistä testeistä, jotka rikkoivat testien toiminallisuuden. Kuten esimerkiksi HTML-elementtien ulkoasumuutokset. Uusien ominaisuuksien testaamista varten kerättiin tehtävienhallintaohjelmistosta kaikkien uusien ominaisuuksien tiketit ja karsittiin niistä pois tiketit, joita ei ollut mahdollista testata nykyisellä testausautomaatio-ohjelmistolla tai niitä ei ollut enää olennaista testata. Tiketillä tarkoitetaan tässä kontekstissa kehitystiimin työtehtävää, joka on kirjattu tehtävienhallintaohjelmistoon. Tiketti voi olla esimerkiksi ohjelmiston kehitysehdotus, selvityspyyntö liittyen ohjelmiston toimintaan, korjauspyyntö johonkin haavoittuvuuteen tai ohjelmistovirheeseen. Uusien ominaisuuksien testaamisessa konsultoin esimiestäni, jonka seurauksena saimme lopuksi kerättyä tiketeistä uudet ominaisuudet, joille on kirjoitettava omat testit.

Bungen menetelmän toisessa vaiheessa voidaan yrittää ratkaista ongelmaa olemassa olevan tietoperustan pohjalta, kun on ymmärretty tutkittavan kohteen ongelma (Dresch ym. 2015, 72). Testausautomaation kehitystyössä toinen vaihe tapahtui siten, että havaitut muutostarpeet ja uudet testit pyrittiin tekemään ensisijaisesti olemassa olevien avainsanojen avulla ja vain pienillä muutoksilla avainsanoihin. Bungen menetelmän kolmannessa vaiheessa - jos edellisen vaiheen ratkaisu ei toiminut- voidaan kehittää edellistä ratkaisua edelleen tai kehittää kokonaan uusi keino ongelman ratkaisemiseksi (Dresch ym. 2015, 73). Vanhojen testien päivityksessä ei erityisemmin tullut vastaan tilanteita, joissa olisi pitänyt kehittää aikaisempia avainsanoja tai luoda uusia avainsanoja. Muutamana poikkeuksena oli kehitettävä selaimen avaukseen liittyviä avainsanoja ja sähköpostikansion käsittelyyn liittyviä avainsanoja. Uusiin testeihin puolestaan oli tehtävä useampia uusia avainsanoja, koska uudet ominaisuudet toivat tuotteisiin uusia toiminnallisuuksia, joita vanhoja avainsanoja hyödyntämällä ei ollut mahdollista testata.

Neljännessä vaiheessa Bungen menetelmää mukaillen, vastaan tulee ratkaisun saavuttaminen. Eli aiemmissa vaiheissa toteutettujen toimenpiteiden jälkeen on päästy siihen lopputulokseen, että esillä olevaan ongelmaan on löydetty mahdollinen ratkaisu. (Dresch ym. 2015, 73.) Testausautomaation kehitystyössä neljäs vaihe saavutettiin kaikissa vanhojen testien päivityksessä. Kuitenkin muutamissa uusien ominaisuuksien testeissä nousi esiin tilanteita, joissa ei ollut mahdollista kehittää toimivia testejä tiettyihin ominaisuuksiin. Esimerkiksi joissakin tuotteiden ominaisuuksissa oli mahdollista ladata tiedostoja omalle tietokoneelle. Kuitenkin koska Robot Frameworkin testit toimivat selainpohjaisena, ei ollut mahdollista varmistaa, että tiedosto oli oikeasti latautunut tietokoneelle. Tästä johtuen eräät uusia ominaisuuksia sisältävät testit pelkistettiin esimerkiksi jättämällä pois ominaisuuden testistä itse lataus toiminto.

Viidennessä Bungen menetelmän vaiheessa on löydetty mahdollinen ratkaisu ongelmaan. Tässä vaiheessa on testattava, toimiiko ratkaisu halutulla tavalla (Dresch ym. 2015, 73). Testien testaus itsessään suoritettiin useampaan otteeseen, kun johonkin ominaisuuteen oli kirjoitettu tai korjattu testi. Testi suoritettiin ja varmistettiin, että se toimii. Ennen testien viemistä testipalvelimelle suoritettiin vielä tuotteen kaikki testit kokonaisuutena. Myös testipalvelimella kaikkien tuotteiden testit suoritettiin kertaalleen kokonaisuutena. Bungen menetelmän viimeisessä vaiheessa voidaan arvioida, onko tarpeen tehdä ratkaisuun parannuksia. Jotta parannuksia ratkaisuun voidaan tehdä, on käytävä läpi aikaisempia vaiheita. (Dresch ym. 2015, 73.) Testeihin tehtiin tarvittaessa parannuksia kunkin testausvaiheen yhteydessä. Tuolloin tutkittiin, mikä aiheutti virheen uusissa testeissä, tehtiin tarvittavat muutokset ja toiminnallisuus testattiin uudelleen.

7 Toteutus

Tässä osiossa esiin on nostettuna muutamia esimerkkejä eri tuotteiden testausautomaatiotestien kehittämisen huomioista, joita käsitellään vain pintapuolisesti. Opinnäytetyössä ei voida käsitellä tarkasti eri tuotteissa ja kehitystyössä esiin nousseita asioita, kuten haavoittuvuuksia tai ohjelmointivirheitä. Nämä ovat salassa pidettävää tietoa, jonka vuoksi näitä ei voida julkisesti käsitellä ja analysoida opinnäytetyössä. Testausautomaatiotyökalun kehittämisen loppuvaiheessa järjestin pienimuotoisen koulutuksen kehitystiimille esimieheni pyynnöstä.

7.1 Testausautomaatiotyökalun päivitys

Testausautomaation päivitys toteutettiin käytännössä kahdessa vaiheessa: vanhojen testien korjaus uusien tuotteiden kanssa yhteensopivaksi ja uusien ominaisuuksien testien kirjoittaminen. Kehitystyöhön kului yhteensä aikaa noin kuukausi. Testausskriptejä on tehty kaikkiin Deltagonin tarjoamiin tuotteisiin, eli Sec@GW salattuun sähköpostiratkaisuun, secureForms turvalliseen lomakeratkaisuun, CollabRoom turvalliselle tiedonjakoalustalle, secSigned sähköinen allekirjoitusratkaisuun ja tuotteiden hallintotyökaluun. Ensimmäinen vaihe testausautomaation kehityksessä oli saada vanhat testiskriptit toimimaan uusien tuoteversioiden kanssa. Toinen vaihe testausautomaation kehityksessä oli kerätä tietoa uusista ominaisuuksista, joihin on kirjoitettava uudet testit, jonka jälkeen niihin kirjoitettiin uudet testit.

Ensimmäinen vaihe testausautomaation kehityksessä oli saada vanhat testiskriptit toimimaan uusien tuoteversioiden kanssa. Ensimmäinen tehtävä vanhojen testien päivittämiseksi oli suorittaa vanhat testit uusia tuoteversioita vasten. Kun vanhat testit oli suoritettu, pystyttiin testiraportista selvittämään, mitkä testit palauttivat virheen. Seuraavaksi oli käytävä läpi testiraportista selvinneet virheelliset testit ja selvittää, mistä virheilmoitukset johtuivat. Tämä toteutettiin tulkitsemalla testiraportin tuloksia ja toistamalla virheelliseen testiin liittyviä toimintoja. Kun pystyttiin toteamaan, ettei virhe johtunut tuotteesta vaan testin toiminnallisuudesta, pystyttiin muuttamaan vanhan testin toiminnallisuutta. Kuitenkin jos kyseessä oli tuotteesta löydetty ohjelmointivirhe, oli siitä tehtävä tehtävienhallintaohjelmistoon tiketti. Lopuksi suoritettiin vielä kaikki testit toiminnallisuuden varmistamiseksi.

Toinen vaihe testausautomaation kehityksessä oli kerätä tietoa uusista ominaisuuksista, joihin on kirjoitettava uudet testit. Tämän jälkeen niihin kirjoitettiin uudet testit. Tietoa uusista ominaisuuksista kerättiin tehtävienhallintaohjelmistoon tehdyistä tiketeistä. Nämä käytiin läpi esimiehen kanssa ja karsittiin pois ne uudet ominaisuudet, joihin ei pystyisi tekemään testejä. Kun oli selvitettynä mihin uusiin ominaisuuksiin tulee kirjoittaa testit, oli seuraavaksi suunniteltava testien toiminnallisuus. Testien toiminnallisuuden suunnittelemiseksi oli tutustuttava tuotteiden uusien ominaisuuksien toiminnallisuuksiin, joiden pohjalta luotiin

suunnitelmat testien toiminnasta. Kun testejä varten oli luotu suunnitelmat, oli mahdollista aloittaa uusien testien kirjoittaminen. Testien kirjoitusvaiheessa suoritettiin testejä useampaan otteeseen eri vaiheissa testien toimivuuden varmistamiseksi. Testien lisäyksen jälkeen suoritettiin kaikki testit varmistaakseen niiden toiminnallisuus.

Suurimpia muutoksia testausautomaation päivityksessä olivat selaimen avaukseen lisätty ehdollinen avainsana ja sähköpostikansion sivunlataus avainsana, jotka lisättiin kaikkien tuotteiden testeihin. Samoin kaikkiin tuotteisiin oli tehty muutoksia HTML-elementteihin, kuten sivulla näkyviin nappeihin, kenttiin, valikkoihin sekä myös HTML-elementtien attribuutteihin oli tehty muutoksia (kuten id, class, name). Robot Framework selectorit hakevat sivulta useimmiten juurikin HTML -atribuuttien mukaan eri elementtejä (id, class, value jne.). Tämän vuoksi piti kaikkien tuotteiden testeihin päivittää muuttuneiden HTML-elementtien uudet arvot tai tunnisteet. Selaimen avauksen ehdollinen avainsana lisättiin, koska testejä suoritettaessa Robot Framework ei päässyt halutulle nettisivulle, vaan selain palautti virhesanoman. Tätä pyrittiin alkuun korjata päivittämällä Robot Framework ja selaimen ajuri, mutta ongelma ei korjaantunut. Totesimme kuitenkin, että toista kertaa selainta avatessa Robot Framework pääsi halutulle verkkosivulle. Sen vuoksi lisättiin kaikkiin selaimen avainsanoihin ehtoavainsana, joka selaimen avauksen epäonnistuessaan pyrki avaamaan selaimen uudestaan, kunnes selaimen avaus onnistui tai viidennen yrityksen jälkeen palautti virheen.

Sähköpostikansion sivunlatausavainsana lisättiin kaikkiin sähköpostia käyttäviin tuotteiden testeihin, koska testit aika ajoin palauttivat virheen, jos sähköpostia ei löytynyt saapuneet kansiosta. Tämä johtui siitä, että Robot Framework yritti etsiä sähköpostia, joka ei ollut vielä saapunut sähköpostikansioon. Ratkaisuksi tähän kehitettiin ehtoavainsana, joka yritti etsiä saapunutta sähköpostia. Mikäli sitä ei löytynyt, ajettiin avainsana, joka päivitti selaimen enintään kolme kertaa ja jokaisen päivityskerran jälkeen tarkisti, onko sähköposti saapunut. Jos sähköpostia ei kolmen sivun latauksen jälkeen löytynyt saapuneet kansiosta, testi palautti virheen. Lisäksi kaikkiin sähköpostia käyttäviin testeihin lisättiin lopuksi sähköpostilaatikon tyhjennys, eli poistettiin kaikki testin luomat sähköpostit.

7.2 Testausautomaatiokoulutus

Itse testausautomaation jatkokehityksen lisäksi sain esimieheltäni tehtäväkseni myös pitää kehitystiimin jäsenille muutaman tunnin koulutuksen testausautomaatiosta. Näin jokainen kehitystiimin jäsen voisi jatkossa kirjoittaa omat testit itse tekemilleen muutoksille tai uusiin ominaisuuksiin. Koulutusta edeltävänä päivänä ohjeistin kehitystiimin jäseniä asentamaan Robot Frameworkin työtietokoneilleen ja ajamaan asennusohjeiden mukana tulleen yksinkertaisen testiskriptin, jolla varmistettiin asennuksen onnistuminen. Itse koulutusta varten olin valmistellut ytimekkään PowerPoint-esityksen (Liite 1), jossa on läpikäytynä keskeisimmät asiat

testien kirjoittamiseen. Esityksen tukena käytin myös aikaisemmin tekemiäni testejä, joita käytin malliesimerkkeinä. Koulutuksessa käytiin läpi testausautomaation yleiset käytänteet. Koulutuksessa toin esiin testausautomaation kehittämisprosessissa esiin nousseita huomioita ja oivalluksia. Koulutuksessa kävimme läpi myös hyödyllisemmät testaukseen tarvittavat dokumentaatiot, kuten Selenium2Libraryin eri kirjastot, jossa kuvaillaan eri kirjastojen avainsanoja ja niiden toiminnallisuus. Koulutuksen päätteeksi annoin kehitystiimin jäsenille tehtävän kirjoittaa yksinkertainen testi, jossa he pääsivät kokeilemaan koulutuksessa käytyjä asioita. Samalla kuin kehitystiimin jäsenet kirjoittivat testiä, olin itse heidän apunaan ja neuvoin tarvittaessa. Kaikki kehitystiimin jäsenet saivat esimerkki testin tehtyä ja toimimaan oikein.

7.3 Testausautomaatiokoulutuksen sisältö

Testausautomaatiokoulutuksen sisältöä on tarkemmin avattuna tässä osiossa, jotta koulutuksesta ja sen sisällöstä saisi selkeämmän käsityksen. Esille nostettu testausautomaation kehittämisen prosessi ei ole ainoa tai kaikissa tilanteissa toimivin tapa. Tämän koin toimivan parhaiten kehitettäessä Deltagon Groupille testausautomaatiota. Koulutuksessa pyrin myös painottamaan uusien keinojen olevan tervetulleita, mikäli kehitystiimin jäsenet löytävät paremman tavan toimia tai tehostaa nykyistä testausautomaatioprosessia. Kuitenkaan testausautomaation luotettavuudesta ja toimintavarmuudesta tinkimättä.

Tässä osiossa on kohtakohdalta tuotu esiin suorittamaani testausautomaation kehittämisen prosessia, joka toimi runkona järjestetylle koulutukselle. Jotta testausautomaatiota voidaan lähteä kehittämään, on tiedettävä missä muodossa testikokonaisuudet on alun perin kirjoitettu. Kehitystyön testausautomaatioympäristössä on jokaisella testikokonaisuudella kaksi eri tiedostoa, joihin kirjoitetaan testit. Ensimmäinen tiedosto sisältää testitapaukset, joilla kutsutaan eri avainsanoja. Toinen tiedosto sisältää käyttäjän kirjoittamat avainsanat ja muuttujat. Robot Frameworkilla on mahdollista kirjoittaa kaikki testeihin tarvittava yhteen tiedostoon, mutta päädyimme kahteen erilliseen tiedostoon, jotta olisi helpompi erottaa avainsanojen toiminnallisuus itse testitapauksista.

Ennen kuin aloitetaan testien kirjoittaminen, on hyvä suunnitella mitä kaikkea aiotaan testata. Suunnittelun voi tehdä yhteydessä, kun itse testaa tekemänsä muutoksen ennen sen siirtämistä hyväksymistestausjonoon. Tässä vaiheessa ei ole kannattavaa tehdä itse testejä, koska muutos tai uusi ominaisuus saattaa vielä muuttua. Suunnitteluvaiheessa olisi hyvä jo miettiä valmiita testitapausten otsikoita, jotka kuvaavat testattavaa ominaisuutta.
Testejä tehtäessä on hyvä myös pyrkiä kirjoittamaan toistuville toiminnoille mahdollisimman vähän, mutta monikäyttöisiä avainsanoja. Jotta näitä olisi mahdollista käyttää uudestaan eri testitapauksissa, kuten esimerkiksi avainsanoja: sisäänkirjautuminen, selaimen avaus, tietyn napin painaminen tai sivun lataus toiminto. Tällöin kun kyseiseen toiminnallisuuteen tulee

koodimuutoksia ja avainsana ei enää toimi, on tehokkaampaa tehdä korjaus muutamaan avainsanaan. Verraten tilanteeseen, jossa on lähdettävä etsimään useamman tuhannen rivin seasta, missä kaikkialla kyseistä toimintoa käytetään.

Avainsanoissa käytettävät muuttujat ja toistuvat arvot on myös suositeltava lisätä resource-tiedoston variables-osioon, jotta niitä olisi ketterämpi käyttää uusia avainsanoja luodessa ja muuttaa tarvittaessa. Itse tiedostoon, johon kirjoitetaan testitapaukset, on hyvä antaa jokaiselle tekemälleen testitapaukselle yksilöllinen ja mahdollisimman kuvaava nimi, kuten ”Kirjaudu sisään pääkäyttäjällä testatakseen viestin lähetystä yhdellä vastaanottajalla”. Itse testitapaukset ovat tiedostossa jaoteltuna kommentoituihin osioihin, missä kommentoituna on osio, mitä testit käsittelevät, kuten ”#viestinlähetystestit”, tällöin jos tulee uusia ominaisuuksia, voidaan lisätä uudet testit suoraan samaan osioon. Jokaisen tuotteen testikokonaisuuteen on lisättävä aloitusosio, joka varmistaa testattavan kohteen oletusasetukset ja testien loppuun lisätään oletusasetusten palautus ja luotujen tietojen poisto. Kirjoittaessa testejä on hyvä välillä ajaa keskeneräisiä testitapauksia, jotta mahdolliset virheet tulevat korjatuksi ennen kuin testitapauksia on liikaa. Muutoin kaikkien testitapausten ajaminen voi muuttua aikaa vieväksi. Uusien testien valmiiksi kirjoittamisen jälkeen on tehdyt testit suoritettava ja varmistettava niiden toiminnallisuus. Lopuksi on suoritettava kaikki vanhat, sekä uudet testit kokonaisuudessaan.

7.4 Eettinen tarkastelu

Tämän opinnäytetyön lähteinä on käytetty kirjallisuutta, artikkeleita ja dokumentaatiota koskien testausautomaatiota, Robot Frameworkia ja SeleniumLibrarya. Opinnäytetyötä tehdessä ja lähteiden käytössä on pyritty noudattamaan Tutkimuseettisen neuvottelukunnan (2012, 6) ohjeistusta hyvästä tieteellisestä käytännöstä. TENK:in ohjeistuksen mukaisesti hyvään tieteelliseen käytäntöön kuuluu keskeisesti eettisyys, huolellisuus, tarkkuus ja rehellisyys tieteellisen työn kaikissa vaiheissa. Muiden tutkijoiden töitä lähteenä hyödyntäessä viitataan heidän töihinsä asianmukaisesti. (TENK 2012, 6.)

Tutkimuseettisen neuvottelukunnan antaman ohjeistuksen lisäksi opinnäytetyön teossa on pyritty noudattamaan myös tietotekniikka alan eettisiä käytänteitä, jotka tässä tapauksessa seuraavat Code of Ethics for IT Professionals käytänteitä. (Code of Ethics for IT Professionals 2011.) Tietotekniikka alan eettisissä käytännöissä keskeisiä käytänteitä ovat esimerkiksi lupien ja sääntöjen varmistaminen yhteistyöyrityksen kanssa ennen tutkimuksen aloittamista tai tutkimuskohteen testaamista. Tietotekniikka alalla erityisen merkittävää on myös yrityksen yksityisten tietojen käsittelyssä noudatettava velvollisuus huolehtia, ettei tietoja päädy ulkopuolisille tahoille. Huolehdittava on myös, ettei omalla toiminnallaan aiheuta uusia haavoittuvuuksia tai haavoita kohteen tietoturvallisuutta. Tutkimuksen aikana ilmenevistä

haavoittuvuuksista, parannusehdotuksista tai muista huomioista on ilmoitettava yrityksen tietoturvasta vastaavalle henkilölle. (Code of Ethics for IT Professionals 2011.) Edellä mainittujen merkitys korostuu etenkin kehittäessäni testausautomaatiota Deltagon Group Oy:lle.

8 Arviointi

Opinnäytetyön arviointi koostuu kolmesta osuudesta: esimiehen arviointi tehdystä työstä, kehitystiimin arviointi pidetystä koulutuksesta ja itsearvioinnista. Esimiehen kanssa arviointi toteutettiin haastattelun muodossa, jossa päällimmäisenä tarkoituksena oli saada palautetta tehdyn työn onnistumisesta, ajankäytöstä, työn hyödyistä yritykselle ja jatkokehityksestä. Kehitystiimin arviointi toteutettiin kyselyn muodossa, jossa pyrittiin selvittämään, oliko koulutuksesta hyötyä, onko tarvetta lisäkoulutukselle ja mitä voisi tehdä paremmin. Itsearvioinnissa puolestaan analysoin opinnäytetyön tavoitteita, saavutettiinko tavoitteet ja lisäksi arvioin omaa oppimistani ja ammatillista toimintaa.

8.1 Työelämäyhteistyökumppanin arviointi

Esimiehen kanssa käytävää arviointihaastattelua varten olin laatinut kysymyksiä liittyen kehittämistyön onnistumiseen, yrityksen kehittämistyöstä saamaan hyötyyn, ajankäyttöön sekä jatkokehittämismahdollisuuksista. Kaiken kaikkiaan kehittämistyö arvioitiin onnistuneeksi. Kehittämistyön myötä yrityksellä on käytössä kattavat automaattiset testit, jotka ovat hyödyksi erityisesti kehittäessä tukea uudelle käyttöjärjestelmälle, jolloin kaikki uudet ominaisuudet on testattava uudella alustalla. Vanhat ja uudet automaattiset testit saatiin luotettaviksi, joka näkyy virheraporttien vähentymisenä. Aiemmin automaattisissa testeissä oli luotettavuusongelmaa niiden palauttaessa virheitä, vaikka itse sovellus oli virheetön. Kehittämistyölle varattu aika arvioitiin sopivaksi suhteessa yrityksen saamaan hyötyyn, vaikka tämä aika olikin poissa muusta työstä. Testausautomaation avulla varmistetaan ja parannellaan yrityksen tuotteiden laatua. Automaattisten testien avulla on mahdollista toistaa suuri määrä testejä säännöllisin väliajoin. Resurssien vuoksi näiden tekeminen käsin ei olisi mahdollista. Säännöllisesti testejä toistamalla voidaan huomata mahdolliset virheet hyvissä ajoin. Samalla myös vanhoja ominaisuuksia testaamalla voidaan tarkistaa, etteivät nämä ole menneet rikki muutoksien yhteydessä. Kehittämistyön aikana nousi esiin erilaisia huomioita. Yhtenä oleellisena huomiona se, että erilaiset ympäristöt saattavat aiheuttaa ongelmia, kun testejä kehitetään yhdessä ympäristössä ja testit ajetaan toisenlaisessa. Ensimmäisen testausautomaation kanssa tämä aiheutti useita ongelmia, mutta kehittämisprosessin edetessä tämä osattiin ottaa huomioon aikaisemman kokemuksen ansiosta. Kehittämisehdotuksena, automaattisten testien kirjoittamisen vahvistaminen koko muun kehitystiimin tasolla. Myös koko prosessin automatisoiminen on tulevaisuudessa tehtävälistalla, toistaiseksi automaattiset testit on käynnistettävä käsin.

Laadin pienimuotoisen kyselyn testausautomaatiokoulutukseen osallistuneille kehitystiimin jäsenille. Kyselyn avulla oli tarkoituksena saada palautetta erityisesti, koulutuksen hyödyllisyydestä, lisäkoulutuksen tarpeesta, sekä ajatuksia siitä, mitä voisi tehdä paremmin. Kehitystiimin jäsenten antaman arvioinnin ja palautteen perusteella koulutus koettiin hyödylliseksi ja koulutukselle varattu aika arvioitiin riittäväksi. Koulutus toi lisätietoa Robot Frameworkista ja koulutuksen aikana oli mahdollista käydä läpi kysymyksiä testausautomaatioon liittyen. Erityisesti aiheeseen liittyvä harjoitustehtävä koettiin hyödylliseksi osallistujien päästessä itse tekemään ja kokeilemaan. Koulutuksen myötä osallistujat arvioivat pystyvänsä jatkossa kehittämään automaattisia testejä itsenäisesti.

8.2 Itsearviointi ja oman oppimisen arviointi

Opinnäytetyön tavoitteena oli toimivan ja luotettavan testausautomaation kehittäminen. Tavoite saavutettiin testausautomaatiotyökalun jatkokehittämisellä ja järjestämällä koulutus kehitystiimin jäsenille.

Testausautomaation jatkokehitykseen kuului aikaisempien testien kehittäminen uusien tuoteversioiden kanssa yhteen sopivaksi ja tuotteiden uusia ominaisuuksia varten uusien testien kirjoittaminen. Testausautomaation jatkokehityksen arvioin onnistuneeksi, sillä automaattiset testit eivät enää palauttaneet testeistä johtuvia virheitä. Testien suorittamista ei kuitenkaan pystytty nopeuttamaan ajallisesti, sillä tuotteiden testeihin lisättiin satoja uusia testitapauksia. Kehitystyön aikana kokeiltiin lyhentää testien suoritusaikaa vähentämällä testitapausten välistä aikaa. Toisin sanoen yhden testitapauksen loppumisen jälkeen tulevan odotusajan ennen toiseen testitapaukseen siirtymistä. Kehitystyön valmistumisen jälkeen suoritetut testit palauttivat virheitä johtuen ohjelmointivirheistä tai muutoksista tuotteisiin, mikä olikin toivottu lopputulos.

Testausautomaatiokoulutuksen järjestämisen päämäärä oli opettaa kehitystiimin jäsenille tarvittavat taidot, jotta jokainen kehittäjä pystyisi kirjoittamaan omat automaattiset testit tekemilleen koodimuutoksilleen. Koulutuksen koin onnistuneeksi osallistujilta saadun palautteen perusteella. Jokainen osallistuja koki oppineensa riittävästi testausautomaatiosta osatakseen kirjoittaa itsenäisesti omia automaattisia testejä. Osa kehittäjistä onkin jo lisännyt kirjoittamiaan testejään testausautomaatiotyökaluun.

Opinnäytetyöprosessia aloittaessani päätin kirjoittaa opinnäytetyöpäiväkirjaa, jotta saisin kootusti kirjattua opinnäytetyön vaiheita, muistiinpanoja, huomioita ynnä muuta. Opinnäytetyöpäiväkirja toimi myös itsearvioinnin välineenä. Opinnäytetyöpäiväkirjan pitämisestä oli myös todella paljon hyötyä testausautomaatiokoulutuksen järjestämisessä, missä pystyin tuomaan esiin opinnäytetyöpäiväkirjaan merkitsemiäni huomioita ja ratkaisuja testausautomaation kehitykseen liittyen.

Oman oppimisen kannalta koin opinnäytetyön kehitystyön ja itse opinnäytetyöraportin kirjoittamisen olevan todella antoisaa. Kehitystyössä huomasin, miten omat taidot ovat kehittyneet. Opinnäytetyötä varten olin tutustunut testausautomaation teoriaan, jonka myötä osaan arvostaa testausautomaatiota täysin uudella tavalla. Aikaisemmin testausautomaation kehitys oli enemmänkin ylimääräistä työtä kehitystyön ohessa, koen nyt ymmärtäväni testausautomaation olevan olennainen osa ohjelmistotestausta. Kehitetyn testausautomaatiotyökalun tuloksetkin ovat osoittaneet automaattisten testien hyödyllisyyden, sillä päivitettyjen testien tulokset ovat tuoneet esille useita virheitä, käsin testaamalla ei olisi ollut mahdollista havaita. Teoriaan perehtyminen ja kehitystyön tekeminen helpottivat myös huomattavasti testausautomaatio koulutuksen pitämistä. Seurauksena pystyin hahmottamaan paremmin, mihin koulutuksessa pitäisi pyrkiä ja mitkä ovat olennaisia asioita, jotka kehittäjien olisi hyvä oppia tuottaakseen kattavia ja luotettavia automaattisia testejä.

Kuten esimiehen arvioinnista tuli esille, on tulevaisuudessa tarkoitus kehittää entisestään testausautomaatio-osaamista koko kehitystiimin tasolla ja automatisoida testien suoritusprosessi. Testausautomaation jatkokehityksenä voisi olla automaattisten testien kehitysprosessin integroimista vahvemmin mukaan itse kehitysprosessiin. Toisin sanoen, kun tehdyt koodimuutokset ovat menneet kehitystiimin sisäisen testauksen läpi ja parannusehdotukset on lisätty, pitäisi sen jälkeen lisätä automaattiset testit. Tällöin ei myöhemmin tarvitse selvittää, mihin kaikkiin muutoksiin on kirjoitettava testit. Koodimuutoksen ollessa vielä tuoreessa muistissa, ei toiminnallisuuden selvittämiseen kulu yhtä paljon aikaa.

9 Pohdinta

Tämän opinnäytetyön tarkoituksena oli kehittää Deltagon Group Oy:n testausautomaatiota, kehittämällä itse testausautomaatiotyökalusta luotettavampi ja kattavampi, sekä järjestää koulutus kehitystiimin kaikille jäsenille automaattisten testien kehittämisestä. Testausautomaatiotyökalua ja testausautomaatiokoulutusta arvioitiin yhdessä esimiehen ja kehitystiimin jäsenten kanssa. Opinnäytetyön tuloksena saatiin kehitettyä yrityksen testausautomaatiotyökalusta toimiva ja luotettava apuväline ohjelmistotestaukseen. Kuten työelämäyhteistyökumppanin arvioinnissa nousi esille ennen kehittämistyön aloittamista, testausautomaatiotyökalu palautti usein virheitä johtuen testeissä olevista virheistä. Uusia testejä suorittamalla huomattiin heti, miten testausautomaatiotyökalu oli parantunut. Testejä pystyttiin luotettavasti suorittamaan useampaan otteeseen ilman virheiden palautumista. Testien palauttaessa virheitä, johtuivat ne ohjelmistovirheistä tai muutoksista, ei testien toiminnallisuuksista. Tärkeimpiä muutoksia testausautomaatiotyökaluun olivatkin uudet ehtoavainsanat, jotka parantavat testien toimintavarmuutta huomattavasti. Testausautomaatiokoulutuksen myötä

puolestaan saatiin automaattisten testien kehitysprosessista ketterämpi ja tehokkaampi. Testausautomaatiokoulutuksen seurauksena automaattisia testejä on kehittämässä useampi kehitystiimin jäsen. Koulutuksen ansiosta kehitystiimin jäsenet ovat jo kirjoittaneet ensimmäiset testinsä testausautomaatiotyökaluun.

Koko opinnäytetyöprosessin ja testausautomaatiotyökalun kehittämisen aikana oli kiinnitettävä erityistä huomiota eettisyyteen. Kehittämisprosessin aikana muutoksia ja korjauksia tehtiin niin testeihin kuin itse tuotteisiin. Opinnäytetyön raporttia kirjoittaessa oli huomioitava eettiset kysymykset yrityksen tietoturvan kannalta. Sen vuoksi opinnäytetyöraportissa ei ole nostettu esiin yksityiskohtaisia tietoja, sillä testeihin ja tuotteisiin tehdyt muutokset ovat luottamuksellisia ja siksi salassa pidettävää tietoa.

Opinnäytetyön kehitystyön ja siitä saadun palautteen myötä on noussut esiin myös useita arvokkaita jatkokehitysideoita, kuten testien suorituksen automatisointi ja automaattisten testien kehityksen lisääminen osaksi tuotekehitysprosessia.

10 Lähteet

Painetut

Bisht, S. 2013. Robot Framework Test Automation. Birmingham: Packt Publishing Ltd.

Dresch, A., Lacerda, D. & Antunes Jr., J. 2015. Design Science Research. A Method for Science and Technology Advancement. Switzerland: Springer International Publishing.

Kasurinen, J. 2013. Ohjelmistotestauksen käsikirja. 1., painos. Jyväskylä: Docendo Oy.

Toikko, T. & Rantanen, T. 2009. Tutkimuksellinen kehittämistoiminta -näkökulmia kehittämisprosessiin, osallistamiseen ja tiedontuotantoon. 3., korjattu painos. Tampere: Tampereen Yliopistopaino Oy – Juvenes Print.

Vilkka, H. & Airaksinen, T. 2003. Toiminnallinen opinnäytetyö. Helsinki: Tammi.

Sähköiset

Bartlett, J. 2017. Why Automated Testing Will Never Replace Manual Testing. Viitattu 3.1.2019. https://blog.testlodge.com/why-automated-testing-will-never-replace-manual-testing/

Code of Ethics for IT Professionals. 2011. Illinois Institute of Technology. Viitattu 9.1.2019. http://ethics.iit.edu/ecodes/node/3115

CrossBrowserTesting. 2019. Why Begin Test Automation- A Faster, More Scalable Test Strategy. Viitattu 6.1.2019. https://crossbrowsertesting.com/resources/why-automate-testing

Deltagon Group Oy. 2019. Viitattu 3.1.2019. https://www.deltagon.com/company

GitHub. 2019a. Robot Framework Quick Start Guide. Viitattu 24.4.2019. https://github.com/robotframework/QuickStartGuide/blob/master/QuickStart.rst

GitHub. 2019b. Selenium2Library. Viitattu 11.1.2019. https://github.com/robotframework/Selenium2Library

Rachel, L. 2017. Test Automation Can't Replace Manual Testing. Viitattu 3.1.2019. https://dzone.com/articles/why-test-automation-cannot-replace-manual-testing

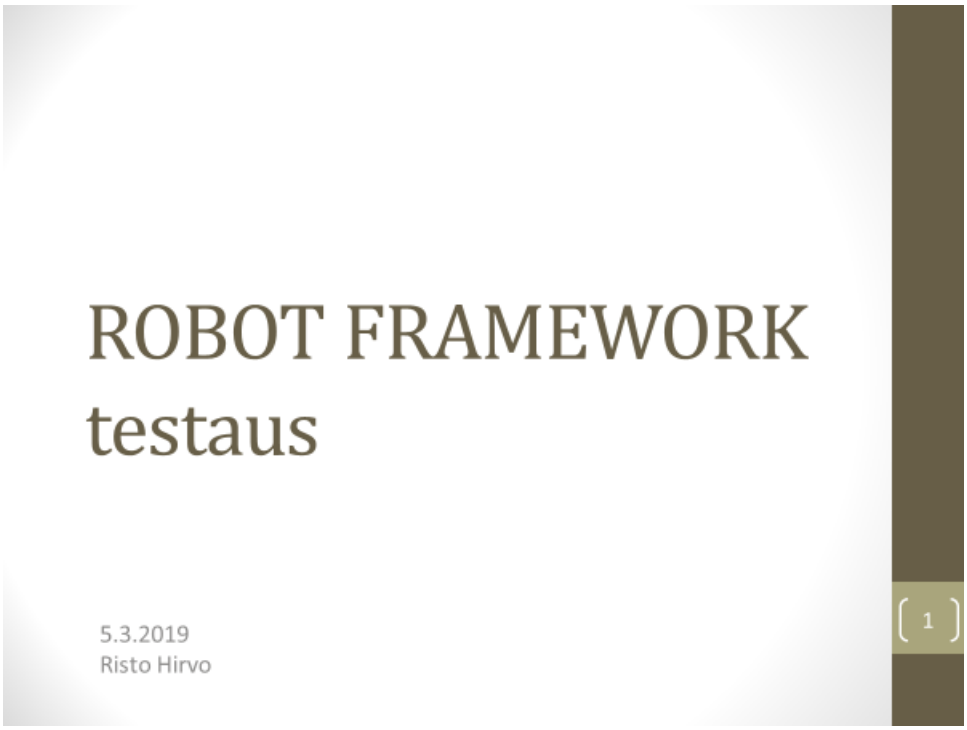
Selenium2Library. 2017. Viitattu 11.1.2019. http://robotframework.org/Selenium2Library/Selenium2Library.html

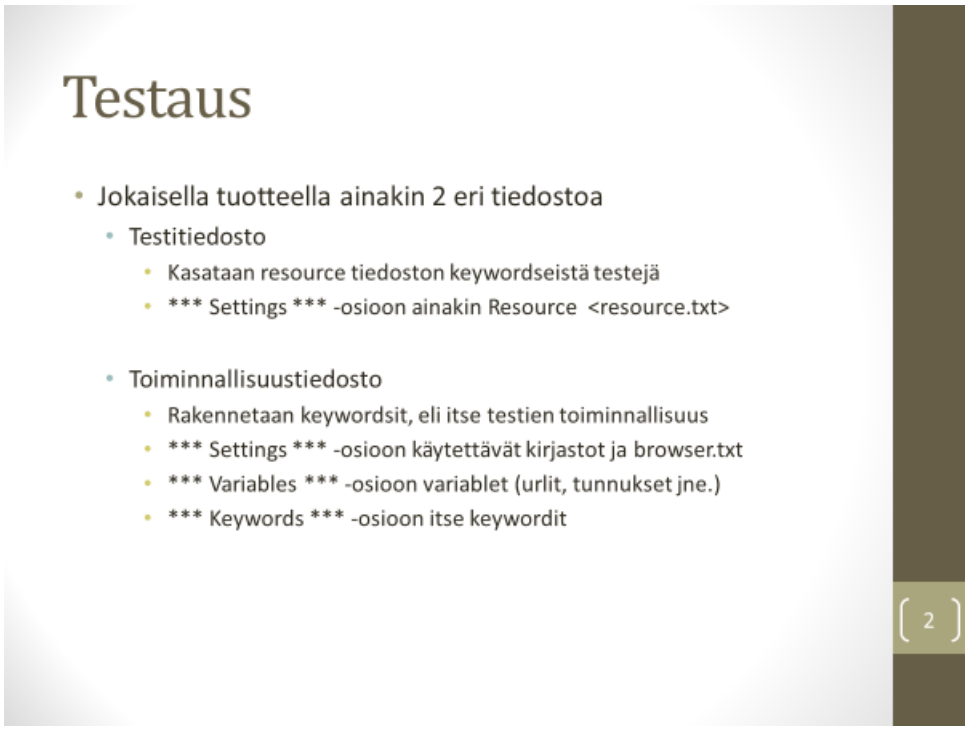
Tutkimuseettinen neuvottelukunta. 2012. Hyvä tieteellinen käytäntö ja sen loukkausepäilyjen käsitteleminen Suomessa. Viitattu 9.1.2019. https://www.tenk.fi/sites/tenk.fi/files/HTK_ohje_2012.pdf

11 Kuviot

12 Liitteet

Liite 1: Testausautomaatiokoulutuksen powerpoint-diat





Testaus

- Robotilla kehitys ja tuotanto versiot, eli kehitysversion ominaisuuksien testit Robootin kehitysversioon ja tuotantoversion ominaisuudet Robootin tuotantoversioon
- Ennen testien kirjoittamista kannattaa mitettiä/suunnitella mitä testaa
- Testejä kirjoittaessa kannattaa välillä ajaa testejä, että näkee toimiiko
- Ennen lisäystä testipalvelimelle, aja tekemäsi testit ja varmista, että toimii

3

Testaus

- Selaimen avaukseen tehty unless/looppi
- Tiedostojen lisäykseen käytetään javascriptii
- Mailboxi tyhjäks aina testin jälkee
- Käytä samaa resource filuu ja valmiita keywordseja jos löytyy
- Testien alkuun aina oletusasetusskripti
- Testien lopuks aina 'close browser'/'close all browsers'
- Robootti osaa vaihtaa selaimen ikkunoita: 'switch browser'keywordilla

4

asennus ja muut ohjeet:

- keywordsit: http://robotframework.org/Selenium2Library/Selenium2Library.html
- iffit unlessit, muut: http://robotframework.org/robotframework/latest/libraries/BuiltIn.html
- inputtien muuntelua: http://robotframework.org/robotframework/latest/libraries/String.html
- päivämäärät muut: http://robotframework.org/robotframework/latest/libraries/DateTime.html
- xpath cheatsheetti: https://devhints.io/xpath
- w3 xpathi: https://www.w3schools.com/xml/xpath_syntax.asp