Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Sulautetut ohjelmistot

2017

Petri Linja-aho

TESTAUSAUTOMAATIO ROBOT FRAMEWORKILLÄ

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut ohjelmistot

2017 | 30

Petri Linja-aho

TESTAUSAUTOMAATIO ROBOT FRAMEWORKILLÄ

Ohjelmistotestaus on tärkeä osa ohjelmistokehitystä, ja testausautomaatio tuo testaukseen tarkkuutta ja toistettavuutta. Testausautomaation tarjoamat hyödyt ovat houkuttelevia erityisesti jatkuvaa testausta vaativissa ohjelmistoprojekteissa. Sen implementointi vaatii resursseja, mutta pitkällä aikavälillä voidaan saavuttaa merkittäviä säästöjä.

Tämän työn tavoitteena oli pilotoida Robot Frameworkin käyttöä kansainväliseen PerkinElmer-teknologiakonserniin kuuluvan Wallac Oy:n ohjelmistotuote LifeCycle 5.1:n testauksessa. Robot Framework on hyväksyntätestaukseen tarkoitettu ohjelmistokehys, jonka avulla testattava ohjelma LifeCycle 5.1 on raskausajan seulontalaboratorioihin tarkoitettu datanhallintatyökalu.

Työn toteutus aloitettiin tutustumalla Robot Frameworkin käyttöön ja asentamalla paikallinen testiympäristö. Toteutuksessa hyödynnettiin verkkoselaimen ohjauksen mahdollistavaa testikirjasto Selenium2Libraryä yhdessä Robot Frameworkin kanssa. Pilotointi toteutettiin kirjoittamalla LifeCycle 5.1:n käyttöliittymää testaavia testitapauksia, jotka hyödyntävät Robot Frameworkin erilaisia ominaisuuksia. Yhdeksi testitapaukseksi määritettiin myös potilasdatan syöttö dataa generoivasta Excel-tiedostosta, joka toteutettiin luomalla datahallintatyökalu pandasia hyödyntävä testikirjasto.

Työn tuloksena saatiin erilaisia testitapauksia, joita voidaan hyödyntää jatkokehityksessä ja pilotoinnin laajennusvaiheessa. Toteutettu Excel-tiedostosta testidataa lukeva testitapaus on lähes sellaisenaan mahdollista ottaa käyttöön ja lisätä jatkuvan integraation järjestelmään. Robot Frameworkin havaittiin olevan tehokas testausautomaatioprosessin luomiseen soveltuva työkalu ja sen syntaksin mahdollistavan selkolukuisten testitapausten kirjoittamisen melko vaivattomasti.

Työlle asetetut tavoitteet toteutuivat, mutta testausautomaatioprosessin luonnin kannalta tärkeä testitapausten lisääminen jatkuvan integroinnin järjestelmään jätettiin pois sen vaatiman laajuuden vuoksi.

ASIASANAT:

Robot Framework, ohjelmistokehitys, testaus, automaatio

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Software

2017 | 30

Petri Linja-aho

TEST AUTOMATION WITH ROBOT FRAMEWORK

Software testing is an important part of software development and the use test automation provides test cases with accuracy and repeatability. The benefits of test automation are appealing especially for software projects that require continuous testing. The implementation of a test automation process requires resources but significant savings can be achieved in the long run.

The purpose of this thesis was to pilot the use of Robot Framework by utilising it for test automation with the LifeCycle 5.1 software. Robot Framework is a test automation framework for acceptance testing. LifeCycle 5.1 is made by Wallac which is a part of multinational corporation PerkinElmer.

The thesis execution was started by familiarising with the use of Robot Framework and creating a local development environment. The Robot Framework library Selenium2Library which enables the controlling of web browsers was used in the execution. The pilot was executed by writing test cases for testing the user interface of LifeCycle 5.1 that utilise the different expandable properties of Robot Framework. A specified test case to extract patient data from an Excel file was written by creating a library that utilises the *pandas* data management library toolkit.

The results of this thesis were different test cases that can be utilised in the development process of future test cases and the continuation of the test automation pilot project. The test case for inputting data from the Excel file can be almost directly used for testing when Robot Framework is implemented to a continuous integration system. Robot Framework was found to be a powerful and a suitable tool for the creation of a test automation process. Its syntax provides easy readability and test cases can be written quite effortlessly.

The objectives of this thesis were achieved although the continuous integration implementation, while being an important part of a test automation process creation, was left out of this thesis work due to its needed extensiveness.

KEYWORDS:

Robot Framework, software development, testing, automation

SISÄLTÖ

KUVAT

OHJELMAT

TAULUKOT

1 JOHDANTO

Tämän opinnäytetyön aiheena on Robot Frameworkillä toteutettava testausautomaatio. Testusautomaatio tuo ohjelmistotestaukseen tarkkuuta ja toistettavuutta. Sen tärkeys realisoituu henkilöstöresurssien ja rahan säästämisessä. Testausautomaation hyödyt ohjelmistoprojekteissa ovat olleet tiedossa jo pitkään. Aihe pysyy ajankohtaisena teknologian ja tuntemuksen kehittyessä yhdessä ohjelmistojen määrän kasvun ja yhä laajamittaisempien ohjelmistoprojektien mukana.

Ohjelmistotestauksesta ja testausautomaatiosta sekä yksittäisistä testityökaluista löytyy runsaasti materiaalia verkosta ja kirjallisuudesta. Lähdemateriaalia tulee tarkastella objektiivisesti, sillä erityisesti testausautomaation konsultointia tarjoavien yritysten verkkosivustojen materiaalit ovat yksinomaan testausautomaatiota ylistäviä. Painetuista julkaisuista ja mm. testausautomaatiota työkseen tekevien henkilöiden ylläpitämistä blogeista voidaan ammentaa tietoa käytännöllisestä lähestysmistavasta testausautomaation eri osa-alueisiin. Tällaisia ovat esimerkiksi vaadittavat resurssit testausautomaation toteuttamiseen yrityksen sisällä.

Tämän työn tarkoituksena on selvittää, mitä hyötyjä Robot Frameworkillä voidaan saavuttaa PerkinElmer-teknologiakonserniin kuuluvan Wallac Oy:n raskausajan seulontaan suunnatun ohjelmistotuotteen LifeCycle 5.1 testuksessa. Valitun testausautomaatiotyökalun pilotoinnin tarkoituksena on saada kokemusta testityökalun käytöstä ennen sen implementointia ja varmistaa, että sen avulla saavutetaan tarvittavia hyötyjä (Hass 2008, 368–369). Selvitys luodaan pilotoimalla Robot Frameworkin käyttöä kirjoittamalla automatisoituja testejä, jotka testaavat ohjelman selainpohjaista käyttöliittymää. Robot Frameworkin tuntemuksen syventämiseksi kokeillaan myös erilaisten ominaisuuksien ja laajennettavuuksien käyttöä.

Työn alussa käsitellään ohjelmistotestauksen aiheita, jotka ovat Robot Frameworkin kannalta oleellisia eli testausautomaatiota ja avainsanapohjaista testausta. Työssä käytetyt työkalut esitellään ja tarkastellaan testattavaa ohjelmaa ja sen käyttöä sekä ominaisuuksia. Vaatimusmäärittelyn kautta siirrytään testien kirjoituksen työvaiheisiin ja niistä saatuihin tuloksiin. Työn tuloksissa esitetään keskeisiä havaintoja pilotoinnin jatkamisen ja työkalun integroimisvaiheeseen siirtymisen osalta.

2 OHJELMISTOTESTAUS

Ohjelmistotestaus on tärkeä osa ohjelmistokehitystä. Ohjelmiston testauksella pyritään lisäämään ohjelman laatua ja luotettavuutta. Ohjelmistotestauksen tarkoituksena on lähteä olettamuksesta, että testattava ohjelma sisältää virheitä. Ohjelmistotestauksella on mahdotonta löytää ohjelmiston kaikkia virheitä siihen tarvittavien resurssien takia, joten testauksen määrittely ja toteutus perustuvat riskin arvioimiseen. Siksi testaus onkin vaikeaa mutta välttämätöntä. Koska testauksen tarkoituksena ei ole todentaa ohjelman virheetöntä toimintaa, voidaan testitapausta pitää onnistuneena, jos siinä havaitaan virheitä. (Meyers ym. 2012, 5–8.)

Ohjelmistotestauksen päämäärät vaihtelevat ohjelmiston eliniästä riippuen. Ohjelmiston suunnitteluvaiheessa pyritään etsimään mahdollisimman paljon virheitä ja vikoja, jotta saadaan tuotua mahdollisimman laadukas ohjelmisto markkinoille. Ohjelmiston jatkokehitysvaiheessa testauksella pyritään varmistamaan sen olemassa olevien osien virheetön toiminta, kun uusien toiminnallisuuksia kehitetään ja integroidaan. (Holmes 2013, 9–10).

2.1 Testausautomaatio

Automaattisen testauksen tarkoituksena on saada mahdollisimman monia useaan kertaan toistuvia ja testaajalta luovuutta vaatimattomia testauksen osia automatisoiduksi. Automaatiolla pyritään säästämään rahaa ja henkilöstön testaukseen käyttämää aikaa automatisoimalla manuaalisesti haastavia tai hitaasti tehtäviä töitä. Testin eri osia automatisoivat testityökalut mahdollistavat myös suurien datamäärien varastoinnin ja hallinnan. Testityökaluiden hankkiminen on usein kallista, koska niiden ylläpito ja implementointi vaativat resursseja. Testityökaluista saatavat rahalliset hyödyt syntyvät säästöistä joita saavutetaan, kun testauksen eri toimintoja voidaan tehdä nopeammin ja vähemmillä virheillä. (Hass 2008, 361–364.)

Testausautomaatiotyökalujen käyttöön liittyy myös monia riskejä. Niiden tarjoamien hyötyjen saavuttamiseen tarvittavaa vaivaa saatetaan aliarvioida, koska niiden käyttö vaatii koulutusta ja testien ylläpito luo työkuormaa testauksesta vastaaville kehittäjille. Työkaluista saadut hyödyt ovat myös vaarassa jäädä saavuttamatta, jos niiden rajoitteita ei ole tunnistettu tai niiden implementointi ei ole ollenkaan mahdollista muun

testiympäristön riippuvuuksien takia. Niiden voidaan myös havaita olevan kyvyttömiä saavuttamaan tarvittavia tehokkuus-, luotettavuus- ja laatutavoitteita. (Holmes 2013, 284–285.)

Testausautomaatioprosessia luotaessa on tärkeä selvittää, mitkä testit kannattaa automatisoida ja mitkä testien osat suorittaa manuaalisesti. Testausautomaation käyttöä kannattaa välttää, kunnes mahdolliset ohjelmaan suunnitellut suuret muutokset on implementoitu. Testityökalun ei kannata myöskään olettaa korvaavaan kaikkea manuaalista testausta ja ratkaisevan kaikkia testauksessa mahdollisesti olevia ongelmia.

2.2 Avainsanapohjainen testaus

Avainsanoihin perustuvan testauksen tarkoituksena on määritellä testiskriptejä vastaavia avainsanoja, jotka yhdistävänä linkkinä testityökalu toimii. Tämä mahdollistaa parametrien syöttämisen avainsanojen avulla ilman mahdollisesti hyvinkin monimutkaisten skriptien muuttamista. Avainsanat piilottavat testien teknisyyttä ja monimutkaisuutta, joten testien ajamisen voi tapahtua korkeammalla abstraktiotasolla sellaisten ihmisten toimesta, joilla ei ole teknistä ohjelmointitietämystä. (Hass 2008, 377–378.)

Testitapausten ambiguiteetin eli vaikealukuisuuden vähentymistä voidaan pitää avainsanapohjaisen testauksen hyötynä. Avainsanapohjainen lähestymistapa ei kuitenkaan vähennä testitapauksien implementointiin tarvittavaa työmäärää, vaan vaatii kehittäjiltä ja muilta testauksessa mukana olevilta henkilöiltä yhteistyötä, jotta uusien tasojen lisääntyminen testattavan tuotteen ja testien ajamisen välille toimii yhtenevästi ja sen ylläpidettävyys on mahdollista (Hass 2008, 379). Avainsanoja tulee hyödyntää johdonmukaisesti ja yhteisesti sovitulla tavalla kehittäjäpiirin sisällä.

2.3 Testityökalun pilotointi

Testityökalun pilotoinnin tarkoituksena on sen tuntemuksen syventäminen, käyttö pienessä mittakaavassa ja soveltuvuuden arviointi. Testityökalun pilotointivaiheessa on myös tarkoitus törmätä ongelmiin, joiden ratkaisuja voidaan käyttää myöhemmin työkalun implementointivaiheessa. (ISTQB Exam Certification 2017.)

Pilottia toteutettaessa voidaan tarkentaa arviota testityökalun implementoinnin todellisista kuluista ja saavutetuista hyödyistä. Samalla voidaan tunnistaa testiprosessiin ja testityökaluun mahdollisesti tarvittavia muutoksia, jotta se voidaan ottaa käyttöön. (Hass 2008, 368–369.)

3 TYÖSSÄ KÄYTETYT TYÖKALUT

3.1 Python

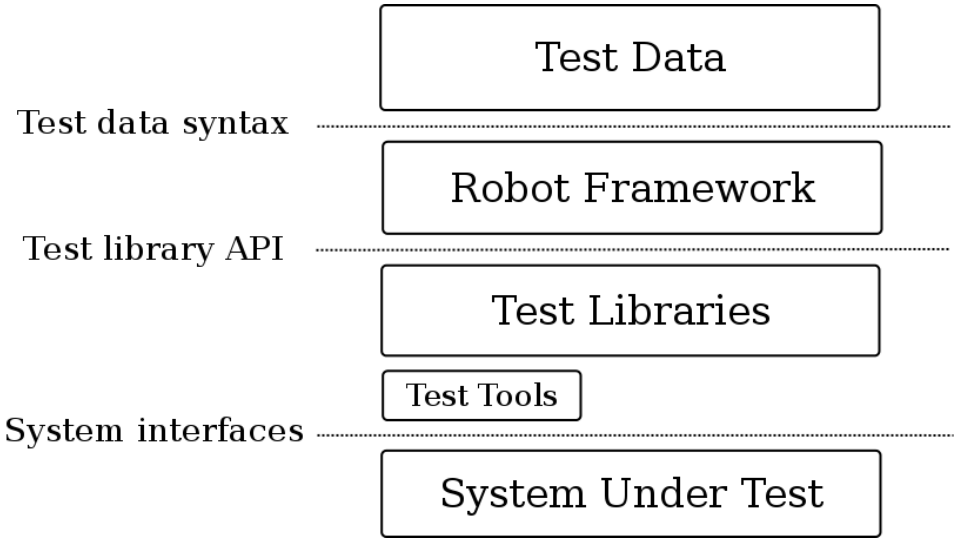
Python on tulkattu olioperustainen ohjelmointikieli. Tulkattavuus mahdollistaa ohjelmakoodin helpon virheenkorjauksen ja päivittämisen, koska ohjelmakoodia ei tarvitse kääntää. Se on järjestelmäriippumattomuus mahdollistaa saman ohjelmakoodin ajamisen muokkaamattomana eri käyttöjärjestelmillä, jotka sitä osaavat tulkata. Python on laajasti käytetty erilaisissa sovelluksissa ja sille on saatavilla runsaasti työkaluja ja kirjastoja. Python on julkaistu avoimen lähdekoodin lisenssillä. (Telles 2007, 1–13.)

3.2 Robot Framework

Robot Framework pohjautuu Pekka Klärckin avainsanapohjaiseen prototyyppiin suuren mittakaavan testausautomaatiokehysympäristöjä käsittelevässä diplomityössään. Tähän perustuvaa kehysympäristöä alettiin myöhemmin kehittämään Nokia Networksillä asiakasprojektissa. Kesäkuussa 2008 Robot Framework julkaistiin avoimen lähdekoodin ohjelmistona, jonka kehitystyössä Pekka Klärck on vieläkin vahvasti mukana. (Klärck 2017.)

Robot Framework on ytimeltään Pythonilla toteutettu hyväksyntätestaukseen tarkoitettu sovelluskehys, joka hyödyntää avainsanapohjaisen testauksen lähestymistapaa. Se käytön kykenevyydet ovat laajennettavissa testikirjastojen avulla ja sen syntaksi on taulukkomuotoinen. Käyttäjät voivat kirjoittaa testikirjastoja omiin tarpeisiinsa Pythonin avulla ja luoda valmiista avainsanoista uusia korkeamman tason avainsanoja. Testisuorituksen käynnistyessä Robot Framework jäsentää ensiksi testidatan, jonka jälkeen se hyödyntää testikirjastojen avainsanoja vuorovaikuttaakseen testattavan järjestelmän kanssa. (Robot Framework 2017.)

Robot Frameworkissa on modulaarinen arkkitehtuuri ja se toimii front endinä sen valmiiksi asennettujen testikirjastojen, sekä sen käyttöominaisuuksia laajentavien ulkoisten testikirjastojen kanssa. (Kuva 1).



Kuva 1. Robot Frameworkin arkkitehtuuri (Robot Framework 2017).

Robot Frameworkin laajennettavuuden ansiosta sitä voidaan käyttää monilla eri tavoilla. Yleisin käyttötapa on sen käyttö verkkosivustojen automatisointiin yhdessä Seleniumin kanssa. Robot Frameworkin tuottaa tulosteena HTML-muotoisen raportin ja lokitiedoston, jotka sisältävät tarkat tiedot tehdyistä testioperaatioista. Nämä tiedostot ovat myös edelleen yhdisteltävissä ja muokattavissa esimerkiksi sen tuottaman koneluettavan tiedoston avulla. (Bisht 2013, 26.)

3.3 Selenium2Library

Verkkoselainautomaatioon tarkoitetun Seleniumin käyttäjiltä kehitysympäristön vaativuuksien yksinkertaistamiseen tulleiden toiveiden johdosta sen rinnalle kehitettiin Selenium2, joka käyttää WebDriveriä verkkoselainten suoraan ajamiseen. Selenium WebDriverin avulla Selenium-testeistä voidaan tehdä joustavampia ja niitä voidaan käyttää useilla eri automatisointia tukevilla verkkoselaimilla. (Bisht 2013, 77)

Selenium2Library on web-applikaatioiden testikirjasto, joka hyödyntää Selenium WebDriveria verkkoselaimen ohjaamiseen. Se käynnistää verkkoselaimen, jossa se ajaa testit natiivisti ikään kuin käyttäjän käyttämänä. (Wallin 2017.)

3.4 Pabot

Testityökalu Pabotin avulla voidaan ajaa testejä rinnakkain, joka vähentää testien suorittamiseen käytettyä aikaa jakamalla testit useampaan prosessiin. Testien jako tapahtuu tiedostoittain eli jokainen testi, joka halutaan ajaa rinnakkain, tulee olla omassa tiedostossaan. Testit tulee suunnitella siten, että niiden samanaikaisuus johda virheisiin jos ne esimerkiksi käsittelevät samaa dataa. (Korpela 2017.)

3.5 Pandas

Pandas on avoimen lähdekoodin kirjasto Python-ohjelmointikielelle, joka tarjoaa työkaluja datan strukturointiin ja data-analyysiin. Pandasin avulla voidaan lukea ja kirjoittaa erimuotoisia tiedostotyyppejä, kuten CSV ja Excel-tiedostoja. (Pandas 2017.)

4 TESTATTAVAN OHJELMAN KUVAUS

PerkinElmer LifeCycle on raskausajan seulontalaboratorioihin tarkoitettu datanhallintatyökalu. Se tarjoaa työkalut potilastietojen syöttöön, riskilaskentaan ja potilasraportointiin, sekä valvontanäkymiä suorituskyvyn tarkasteluun. Ohjelmaa voidaan käyttää tuetuilla verkkoselaimilla kaikilla laitteilla jotka ovat samassa verkossa kuin työasema johon ohjelma on asennettu. (PerkinElmer 2017.)

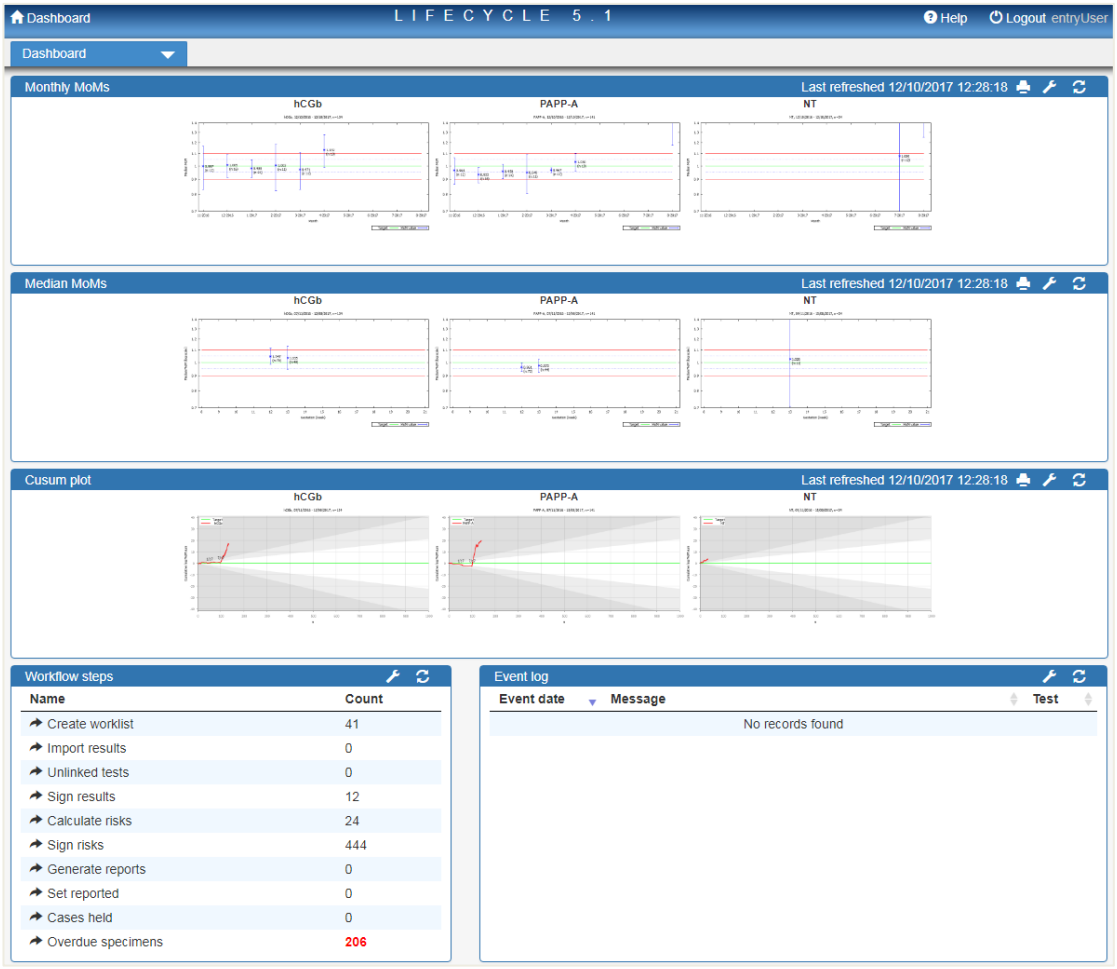
4.1 Sikiöseulonta

Sikiöseulonnalla pyritään havaitsemaan sikiön epämuodostumia eli kromosomi- ja rakennepoikkeavuuksia. Kromosomipoikkeavuuksiin liittyy usein rakennepoikkeavuuksia. Pääosin sikiöseulonnalla pyritään tunnistamaan kromosomipoikkeavuuksta johtuvia oireyhtymiä 21-trisomiaa eli Downin oireyhtymää, 18-trisomiaa eli Edwardsin oireyhtymä ja 13-trisomiaa eli Pataun oireyhtymää. (Autti-Rämö ym. 2005, 19–26.)

Ensisijainen sikiöseulonnan tapa on yhdistelmäseulonta, jossa tulokset saadaan yhdistämällä äidin verinäytteestä mitattujen merkkiaineiden pitoisuuksien määrän tutkiminen ja ultraäänitutkimuksessa niskaturvotuksen paksuuden mittaus. Äidin verinäytteestä tutkittavat pitoisuudet ovat istukkaperäisen hormonin HCG-betan määrä ja raskauteen liittyvän plasman proteiini A:n eli PAPP-A:n määrä. Ultraäänitutkimuksessa mitataan sikiön vartalon pituus eli pääperämitta ja sikiön niskaturvotus. Riskilaskennassa mittaustulokset ja muut riskilaskentaan vaikuttavat tekijät, kuten äidin ikä, yhdistetään laskentaohjelmassa ja riskilaskennan tulos ilmoitetaan kromosomipoikkeavuuden todennäköisyytenä, esimerkiksi 1/250. (Väestöliitto 2017.)

4.2 Käyttöliittymä

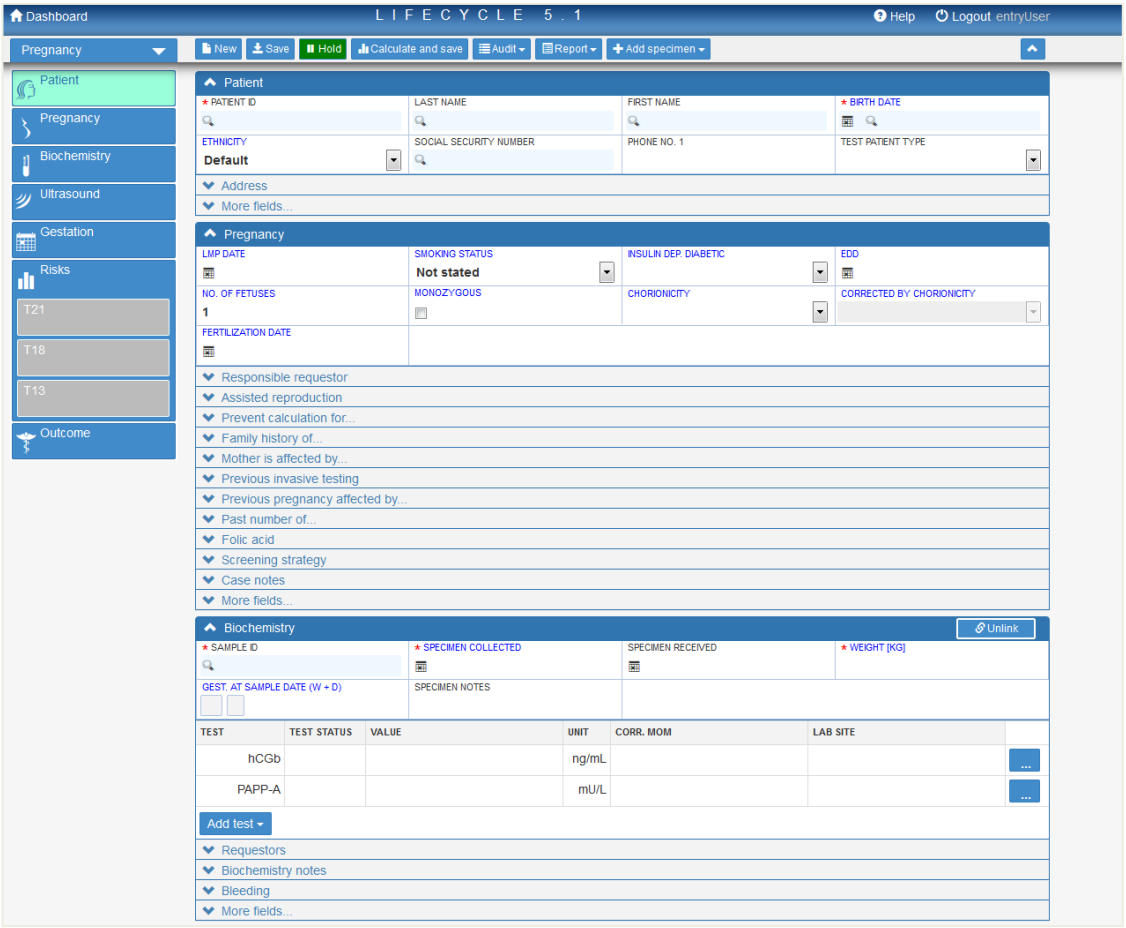
LifeCyclen käyttöliittymä toimii verkkoselaimella ja se on vahvasti muokattavissa. Käyttäjän kirjautuessa ohjelmaan avautuu aloitussivuksi yleiskatsauksen erilaisiin tietoihin tarjoava koontinäyttö (Kuva 2).



Kuva 2. LifeCycle 5.1 -ohjelman koontinäyttö.

Koontinäkymässä on erilaista statistiikkaa näyttäviä widgettejä, joita voidaan muokata käyttäjän tarpeisiin.

Potilastietonäkymässä on erityyppisiä syöttökenttiä, kuten pudotusvalikoita ja valintaruutuja (Kuva 3).



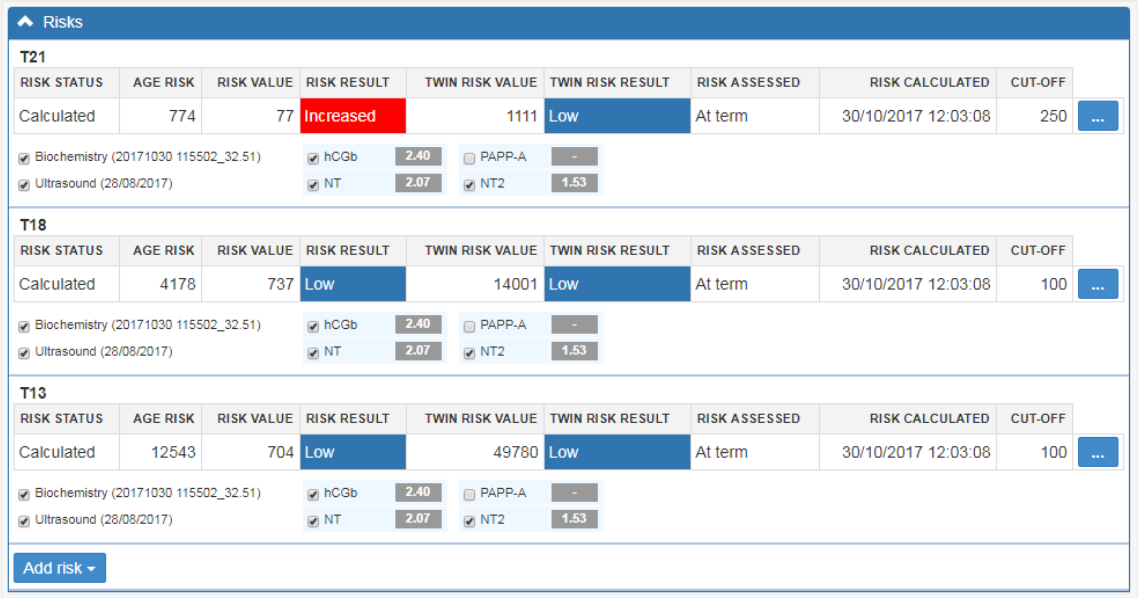
Kuva 3. LifeCycle 5.1 -ohjelman potilastietonäkymä.

Osa potilastietonäkymän syöttökentistä on dynaamisia eli niihin syötettäessä voi käyttöliittymän muissa elementeissa tapahtua muutoksia.

Riskilukuja voidaan laskea erilaisilla laskutavoilla riippuen niistä riskilukuun vaikuttavista arvoista, joita potilaasta on kerätty. Riskilaskennassa käytetty potilaan raskausaika voidaan valita manuaalisesti syötetty ajan, ultraäänitutkimuksessa saadun pääperämitan avulla lasketun ajan tai viimeisten kuukautisten päivämäärän perusteella lasketun ajan välillä.

4.3 Raportointi

LifeCycle 5.1 tarjoaa valmiiden potilasraporttien generoinnin pdf-muotoisena ja riskilaskennan tulokset näkyvät ohjelman käyttöliittymässä (Kuva 4).



Kuva 4. Näkymä riskilaskennan tuloksista.

Kuvasta voidaan nähdä riskilaskennan olevan laskettu ilman PAPP-A-arvoa, koska sitä ei ole syötettynä saatavilla. Testitodistukseksi tarvitaan kuvankaappaus toteutuneesta riskilaskennasta.

5 TYÖN TOTEUTUS

5.1 Testiympäristö

Testattava ohjelma asennettiin samassa verkossa olevalle virtuaalikoneelle etäyhteyden kautta. Ohjelman asennuksen jälkeen sitä pääsee käyttämään samassa verkossa olevalla tietokoneella selaimen kautta. Robot Framework on asennettuna toiselle virtuaalikoneelle, jolloin testien suoritus voidaan siirtää myöhemmin jatkuvan integraation järjestelmään.

Testien kirjoittamista varten Robot Framework -ympäristö asennettiin myös kannettavalle tietokoneelle, jossa on Windows 7 -käyttöjärjestelmä. Tietokoneelle asennettiin Python 2.7, jonka asennus sisältää pip-paketinhallintatyökalun. Paketinhallintatyökalun avulla asennettiin Robot Framework 3.0.2:n lisäksi pandas-kirjaston versio 0.20.3 ja Pabot-työkalun versio 0.38, sekä Selenium2Library 1.8.0 ja ExcelLibrary 0.0.2.

Robot Frameworkin asennus sisältää RIDE-editorin, jolla voidaan kirjoittaa testejä taulukkomuotoisesti, sekä ajaa niitä suoraan editorista. Testit voidaan ajaa myös komentoriviltä, jolloin niiden suorittamiseen saadaan lisää vaihtoehtoja komentorivivalitsimilla ja testien suorituksia voidaan niputtaa yhteen batch-skriptauksella.

Testien kirjoittamista varten asennettiin myös PyCharm Community Edition IntelliBot-liitännäisellä, joka helpottaa testien kirjoitusta tekstimuodossa värikoodaamalla .robot-päätteiset tiedostot ja tunnistamalla asennettujen kirjastojen, sekä käyttäjän kirjoittamat avainsanat.

Eri verkkoselaimiin tarvittiin myös niiden tarjoamat ohjaintyökalut, joiden avulla automatisoita testejä voidaan ajaa. Tarvittavat ajurit ladattiin Chromelle, Internet Explorerille ja FireFoxille. Niiden käyttämiseen luotiin myös tarvittavat ympäristömuuttujat.

5.2 Suunnittelu

Testien kirjoitus suunniteltiin aloitettavaksi ensin kirjoittamalla testi, joka testaa onnistuvaa käyttäjän sisäänkirjausta. Onnistunut sisäänkirjaus vaaditaan myös muissa testitapauksissa, joten sisäänkirjautumisen testi toimii myös ns. smoke-testinä, joka voidaan ajaa ensimmäisenä, jotta saadaan varmistettua sisäänkirjautumisen onnistuminen ilman, että aikaa tuhlautuu muiden testien ajamiseen.

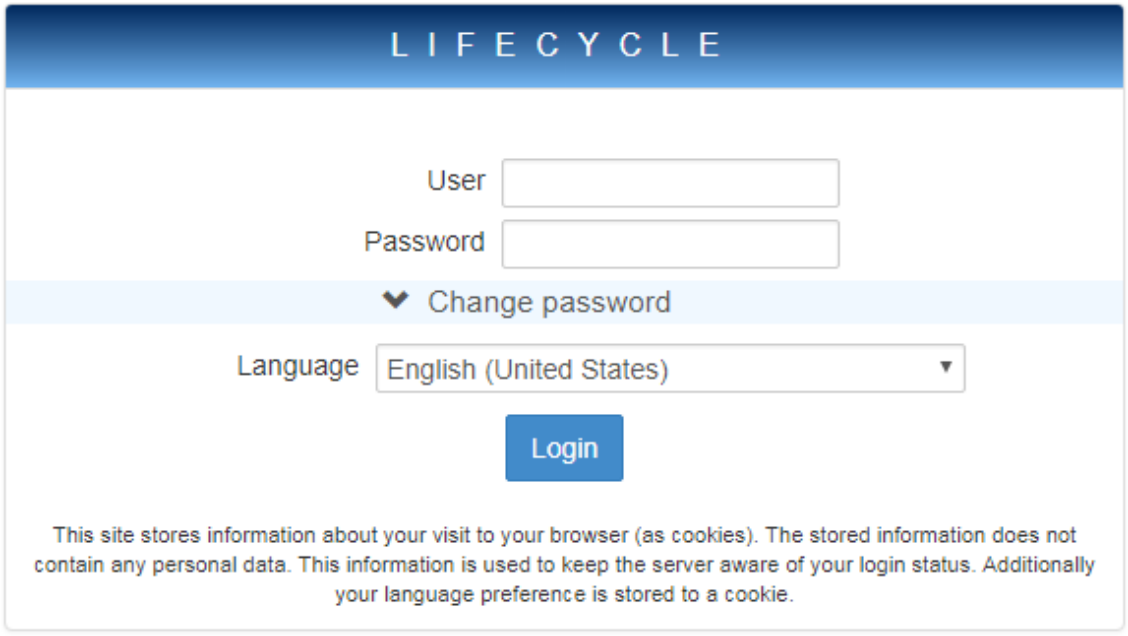
Testitapauksen määritettiin testaavan testipotilaan tietojen syöttöä ja riskilukujen laskentaa. Ensiksi selvitettiin miten nämä toiminnot tehdään manuaalisesti ja minkälaisia arvoja käyttöliittymän eri syöttökenttiin tulee syöttää.

5.3 Testien kirjoitus

Tämä luku sisältää testien kirjoituksen toteutuksen niiden toteutuksen mukaisessa järjestyksessä. Alaotsikoissa on testitapauksia, joissa on erilaisia Robot Frameworkin käyttötapoja hyödyntäviä testejä. Sisäänkirjautumistestin toteutuksen yhteydessä on kuvailtu Robot Frameworkin tuottamat tulosteet ja Excel-tiedostosta dataa lukevan testin toteutuksessa on kuvailtu kuvankaappauksen ottamista testitodistukseksi.

5.3.1 Sisäänkirjautumistesti

Työn toteutus aloitettiin kirjoittamalla testi, joka tekee onnistuneen sisäänkirjautumisen ohjelman selainpohjaiseen käyttöliittymään (Kuva 5).



Kuva 5. LifeCycle 5.1 -ohjelman kirjautumisikkuna.

Robot Frameworkin syntaksissa avainsanat ja argumentit erotetaan toisistaan käyttämällä vähintään kahta välilyöntimerkkiä. Tässä työssä on ohjelmakoodien lukemisen helpottamiseksi käytetty esitystapaa, jossa avainsanat erotetaan argumenteista putkimerkillä. Kirjoitettu testi ohjaa selainta syöttämään ohjelmaan käyttäjänimen ja salasanan, jonka jälkeen painaa kirjautumispainiketta ja tarkistaa, että kirjautuminen onnistuu (Ohjelma 1).

```
*** Settings ***
| Documentation | Test for succesful login.
| Resource | resource.robot

*** Test Cases ***
| Valid Login
| | Open Browser To Login Page
| | Input Username | entryuser
| | Input Password | password
| | Submit Credentials
| | Dashboard Page Should Be Open
| | [Teardown] Close Browser
```

Ohjelma 1. Sisäänkirjautumistesti.

Testi sisältää Selenium2Libraryn selainta ohjaavia avainsanoja, sekä erillisestä resurssitiedostosta löytyviä avainsanoja. Resurssitiedostoon voidaan kirjoittaa avainsanat, jotka sisältävät testin teknisimpiä osia. Tämä mahdollistaa testitiedoston

pysymisen helppolukuisessa muodossa. Resurssitiedoston käyttö helpottaa ja nopeuttaa uusien testitapausten kirjoittamista, koska siinä olevia avainsanoja voidaan käyttää myös muissa testitiedostossa.

Resurssitiedostossa nähdään sisäänkirjautumistestin käyttämät avainsanat, jotka on rakennettu yhdistämällä muita avainsanoja (Ohjelma 2).

```
*** Settings ***
Library | Selenium2Library

*** Variables ***
${LOGIN URL} | https://st-vm-7-8/content/login.html
${BROWSER} | Chrome
${DELAY} | .2s
${WAIT} | 16s

*** Keywords ***
| Open Browser To Login Page
| | Open Browser | ${LOGIN URL} | ${BROWSER}
| | Maximize Browser Window
| | Set Selenium Speed | ${DELAY}
| | Set Selenium Timeout | ${WAIT}
| | Login Page Should Be Open

| Login Page Should Be Open
| | Title Should Be | Login

| Input Username
| | [Arguments] | ${Username}
| | Wait Until Element Is Visible | id=userName
| | Input Text | id=username | ${Username}

| Input Password
| | [Arguments] | ${Password}
| | Wait Until Element Is Visible | id=password
| | Input Text | id=password | ${Password}

| Submit Credentials
| | Click Button | btnLogin

| Dashboard Page Should Be Open
| | Wait Until Element Is Not Visible | loadingOverlay
| | Wait Until Page Contains Element |
//li[@data-i18n='lastAction:action_view_changed']
| |Page Should Contain Button | id=main_nav_btn
```

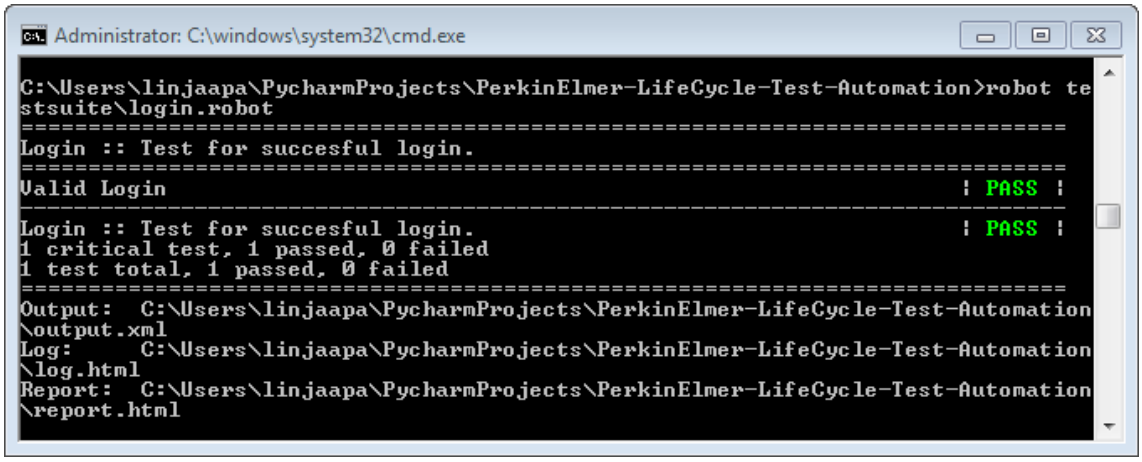
Ohjelma 2. Resurssitiedoston avainsanat.

Muuttujilla voidaan helpottaa testien luettavuutta ja ylläpidettävyyttä. Tässä tapauksessa muuttujina määritellään käytettävän verkkoselaimen ja verkko-osoitteen

lisäksi myös komentojen suorituksen välissä odotettava aika ja eräiden avainsanojen argumenttina käyttämä aikakatkaisu.

Selenium2Libraryn tiettyyn elementtiin kohdistuvat avainsanat ottavat argumenttina etsittävän elementin paikantimen. Tässä tapauksessa paikantimena on käytetty syöttökentän nimeä tai id-attribuuttia ja Xpath-polkulauseketta.

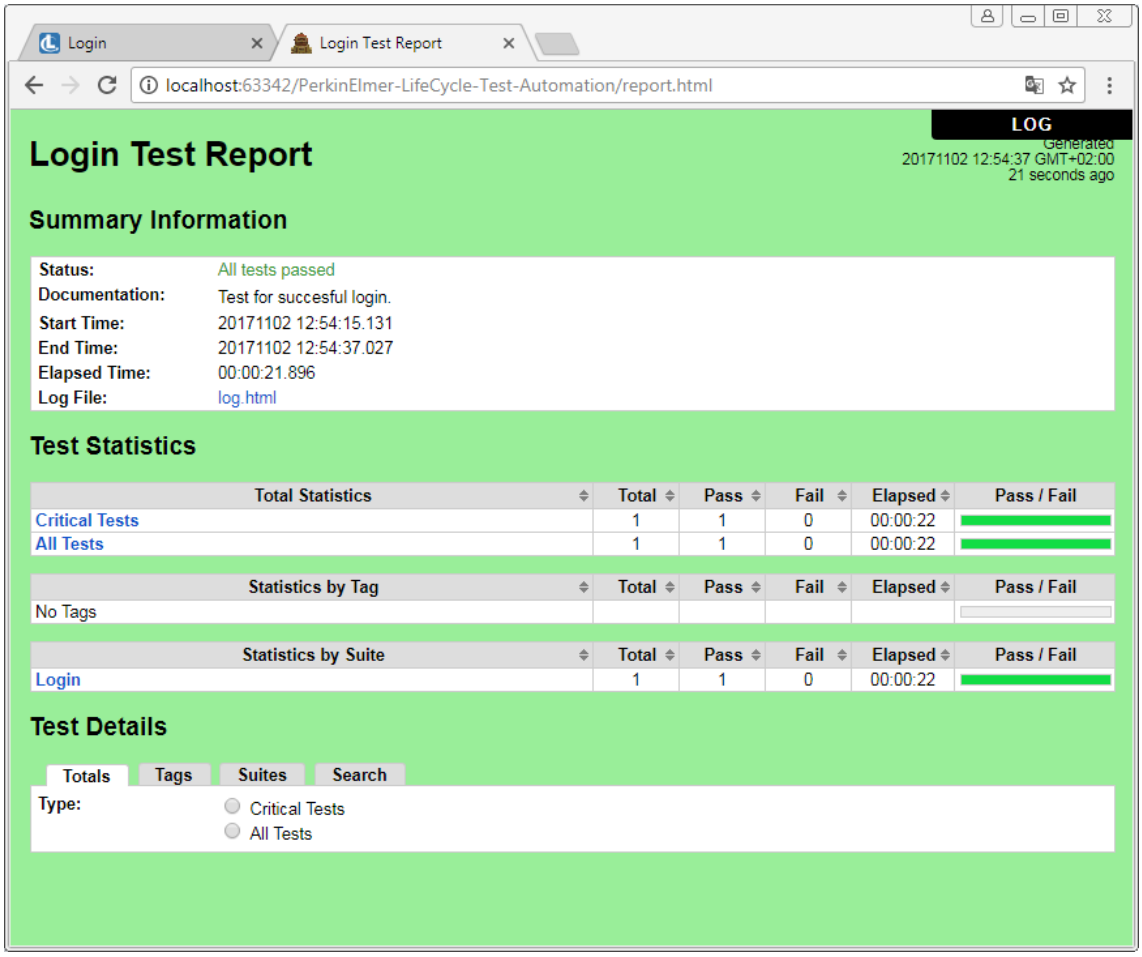
Sisäänkirjautumistesti voidaan suorittaa komentoriviltä ja testisuorituksen tulokset tulostuvat komentoriville (Kuva 6).



Kuva 6. Sisäänkirjautumistestin suoritus komentoriviltä.

Komentorivin tulosteesta voidaan nähdä suorituksen onnistuneen tai epäonnistuneen. Testiraportit generoidaan samaan kansioon, jossa testi ajetaan, jos muuta tulostekansiota ei erikseen määritetä.

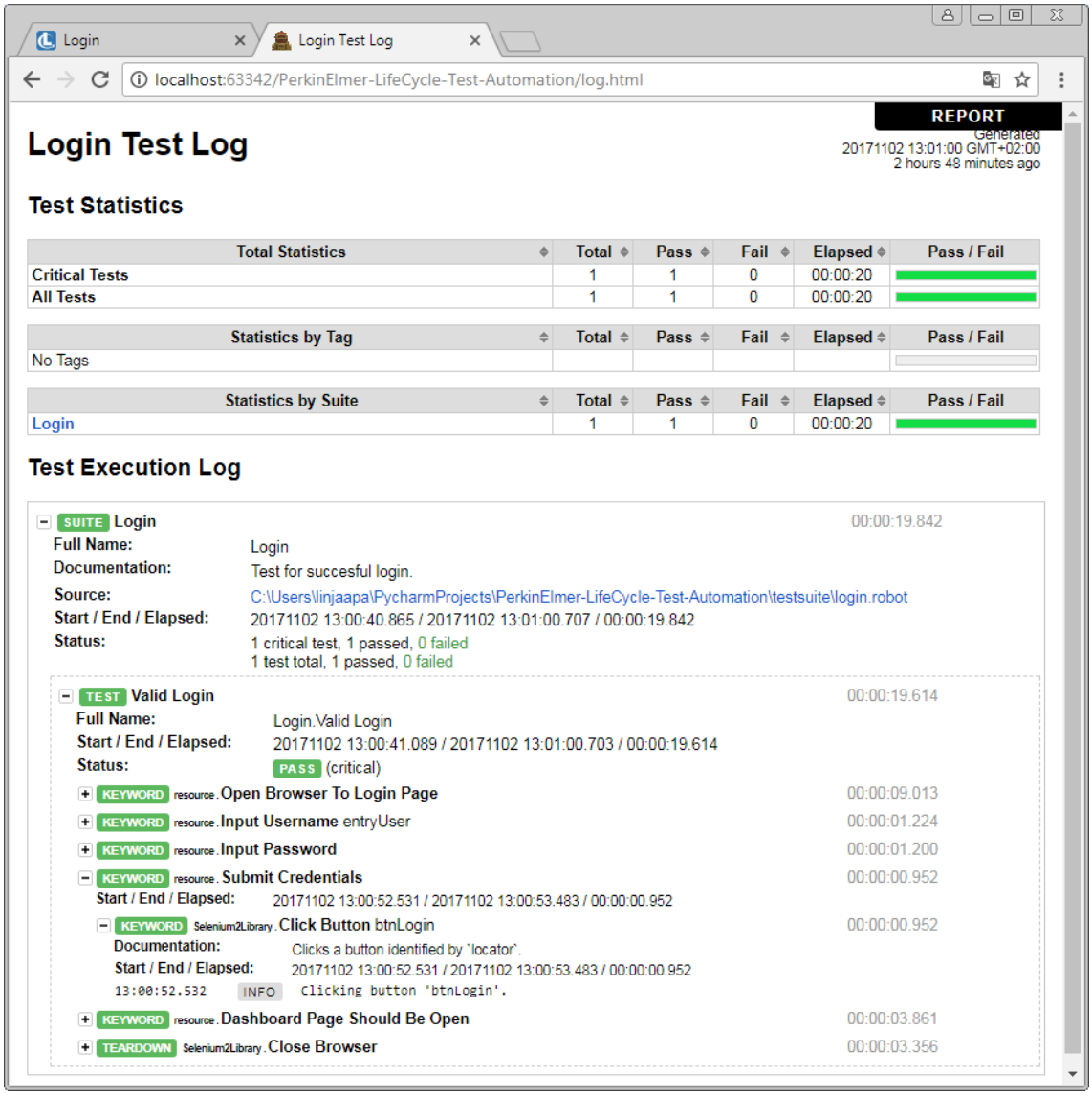
Robot Framework tuottaa testin suorituksesta raportin, josta voidaan tarkastella tietoja testin suorituksesta (Kuva 7).



Kuva 7. Robot Frameworkin tuottama testiraportti.

Raportin vihreästä taustaväristä voidaan heti vilkaisemalla nähdä testien onnistuneen. Taustaväri on punainen, jos jokin testi on epäonnistunut.

Robot Framework tuottaa myös lokitiedoston, josta nähdään testien ja sen sisältämien avainsanojen suoritusjärjestys ja hierarkia (Kuva 8).



Kuva 8. Robot Frameworkin tuottama lokitiedosto.

Lokitiedoston tarjoaa yksityiskohtaisia tietoja avainsanojen suorituksesta ja sen avulla voidaan selvittää minkä avainsanan kohdalla mahdollinen virhe on tapahtunut.

5.3.2 Testien suoritus batch-skriptauksella

Testitapausten kirjoittamista jatkettiin toteuttamalla testi, joka vaihtaa valmiiksi syötetyn potilaan riskilaskentaan vaikuttavia arvoja ja vertaa niiden oikeellisuutta valmiiksi laskettuihin odotettuihin arvoihin (Ohjelma 3).

```
*** Settings ***
| Documentation | Data-driven tests
| ... | Test cases have precalculated risk values
| ... | Changes test values on created patient and checks for correct
risk calculation
| Suite Setup | Open Patient Page
| Suite Teardown | Close Browser
| Test Template | Check Risk Values
| Resource | resource.robot

*** Test Cases *** | hCGb | PAPP-A | T21 | T18 | T13
| Low Risk Values | 38.69 | 2056.79 | 3022 | 100000 | 100000

| Moderate Risk Values | 39.69 | 2056.89 | 2854 | 100000 | 100000

| High risk values | 128.70 | 1950.05 | 196 | 100000 | 100000

*** Keywords ***
| Open Patient Page
| | Login And Open Dashboard Page
| | Open Pregnancy Entry
| | Input Patient | 005

| Check Risk Values
| | [Arguments] | ${hcgb} | ${papp-a} | ${t21} | ${t18} | ${t13}
| | Change hCGb Value | ${hcgb}
| | Change PAPP-A Value | ${papp-a}
| | Calculate Risk Value
| | Risk Values Should Match | ${t21} | ${t18} | ${t13}
| | Take Screenshot
| | [Teardown] | Close Browser
```

Ohjelma 3. Potilaan syötettyjä arvoja vaihtava testi.

Testitapaus on toteutettu datavetoisesti. Testisarjan aluksi on määritelty suorittavaksi valmistava avainsana joka kirjautuu ohjelmaan sisään ja avaa potilastietonäkymän. Varsinainena testitapauksena suoritetaan kolmeen kertaan riskiarvoja eri arvoilla syöttävä avainsana.

Testitiedostojen peräkkäistä suorittamista ja niiden raporttien yhdistämistä batch-skriptauksen kokeiltiin ajamalla sisäänkirjautumistesti ja potilaan riskiarvoja vaihtava testi peräkkäin kahdella eri selaimella (Ohjelma 4).

```
@ECHO OFF

call robot --variable BROWSER:Firefox --name Firefox_Login --log none
--report none --output output/firefox_login.xml testsuite/login.robot

call robot --variable BROWSER:Chrome --name Chrome_Login --log none
--report none --output output/chrome_login.xml testsuite/login.robot

call rebot --name Login --outputdir output --log none --report none
--output login.xml output/firefox_login.xml output/chrome_login.xml

call robot --variable BROWSER:Firefox --name Firefox_Calculate
--log none --report none --output output/firefox_calculate.xml
testsuite/calculate.robot

call robot --variable BROWSER:Chrome --name Chrome_Calculate
--log none --report none --output output/chrome_calculate.xml
testsuite/calculate.robot

call rebot --name Calculate --outputdir output --log none
--report none --output calculate.xml output/firefox_calculate.xml
output/chrome_calculate.xml

call rebot --name Calculate_and_Login --outputdir output
--output login_and_calculate.xml output/login.xml output/calculate.xml
```

Ohjelma 4. Testiraportit yhdistävä batch-skripti.

Kun käytettävä verkkoselain on määritetty muuttujaksi, voidaan komentirivivalitsimella vaihtaa testissä käytettävää verkkoselainta muuttamalla muuttujan arvoa. Testien tulosteista jätetään myös raportti ja lokitiedosto luomatta, koska raporttien yhdistämiseen tarvitaan ainoastaan XML-tiedosto.

5.3.3 Toisistaan riippumattomien testien ajo tagien avulla

Tagien käyttöä ja toisistaan riippumattomien testien ajoa varten tehtiin testitiedosto, jonka sisältämät testitapaukset muuttavat riskiarvojen laskutapaa. Testi luo myös potilaan jos sitä ei ole vielä luotu. Testitiedoston testit suorittavat riskiarvojen laskun verinäytteestä saaduilla arvoilla, sikiön ultraäänitutkimuksesta lisätyillä arvoilla, manuaalisesti lisätyn raskausajan kanssa, sekä testaa tupakoinnin vaikutuksen riskiarvoihin (Ohjelma 5).

```
*** Test Cases ***
| Change Blood Test Values
| | [Documentation] | Calculates risk with hCGb and PAPP-A values
| | [Tags] | blood_test
| | Change hCGb Value | ${HCGB}
| | Change PAPP-A Value | ${PAPP-A}
| | Select Gestation Method LMP
| | Save And Calculate Risk
| | Risk Values Should Match | ${BLD T21 RISK} | ${BLD T18 RISK} |
${BLD T13 RISK}
| | Capture Page Screenshot

| Change Ultrasound Scan Values
| | [Documentation] | Includes CRL and NT values in risk calculation
and calculates risk
| | [Tags] | ultrasound_scan
| | Change CRL Value | ${CRL}
| | Change NT Value | ${NT}
| | Select Gestation Method CRL
| | Save And Calculate Risk
| | Risk Values Should Match | ${USCAN T21 RISK} | ${USCAN T18 RISK} |
${USCAN T13 RISK}
| | Capture Page Screenshot

| Manual Gestation Entry
| | [Documentation] | Calculates risk with manual gestation value
| | [Tags] | manual_gestation
| | Change Manual Gestation Value | ${GEST WEEKS} | ${GEST DAYS}
| | Select Gestation Method Manual Entry
| | Save and Calculate Risk
| | Risk Values Should Match | ${MANUAL T21 RISK} | ${MANUAL
T18 RISK} | ${MANUAL T13 RISK}
| | Capture Page Screenshot

| Change Smoking Status
| | [Documentation] | Inputs patient smoking status as a smoker and
calculates risk
| | [Tags] | smoking_status
| | Click Element | xpath=//*[@id="cd1Smoking"]/option[4]
| | Save And Calculate Risk
| | Risk Values Should Match | ${SMOKE T21 RISK} | ${SMOKE T18 RISK} |
${SMOKE T13 RISK}
| | Capture Page Screenshot
```

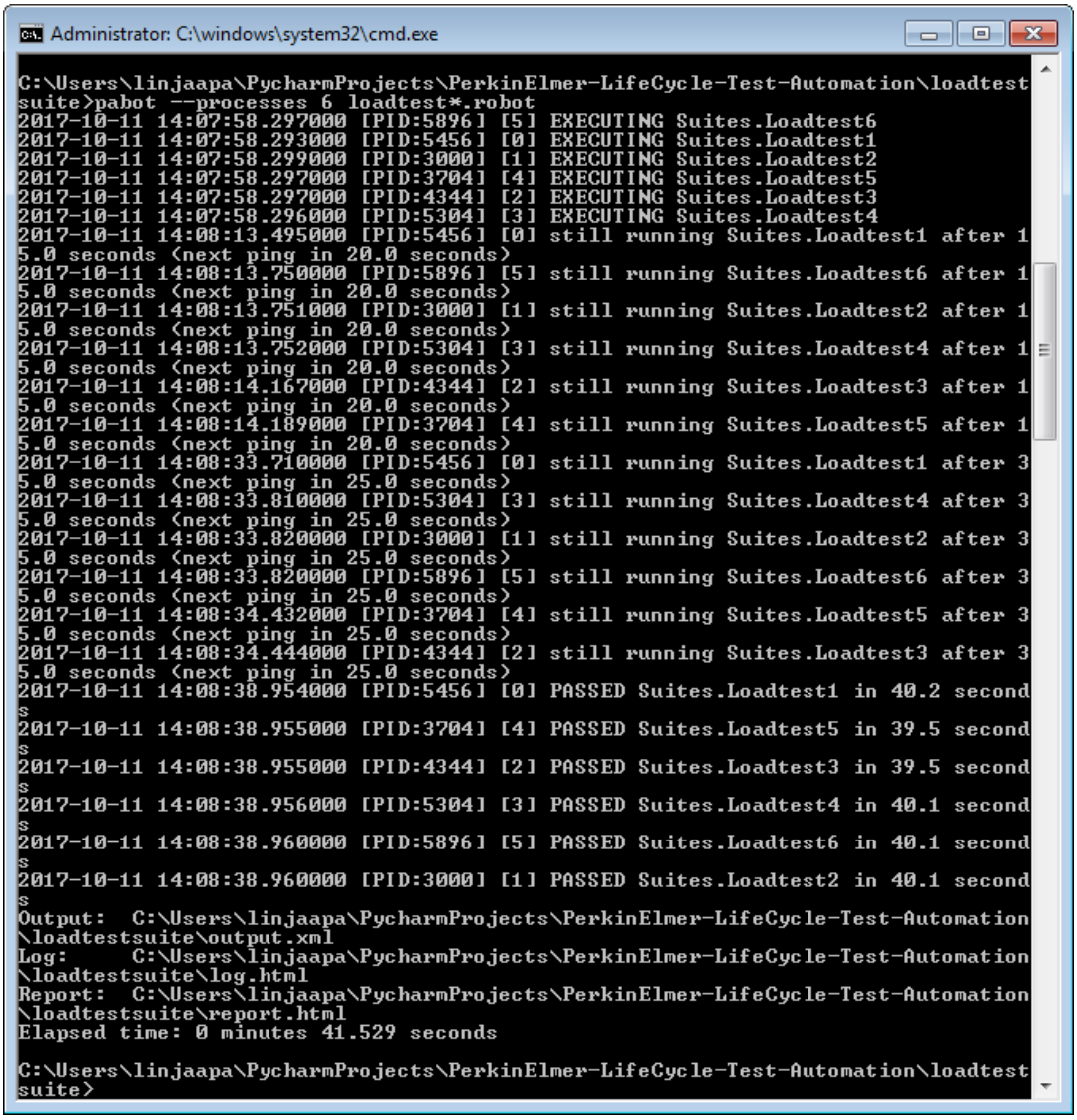
Ohjelma 5. Toisistaan riippumattomat testit.

Testit voidaan ajaa yhdessä tai yksittäin testitapauksille määritettyjen tagien avulla, koska ne on toteutettu toimimaan itsenäisesti ja testisarjan käynnistyessä asetetaan kaikki vaihdettavat potilaan tiedot tiettyihin oletusarvoihin. Tagit näkyvät myös

testiraportissa ja niiden avulla voidaan mm. määrittää tietyt testit kriittisiksi, jolloin niiden testien suorituksen voi priorisoida.

5.3.4 Testien rinnakkaisajo

Stressitestauksen mahdollisuutta kokeiltiin Pabot-työkalulla, joka mahdollistaa testien rinnakkaisen suorituksen (Kuva 9). Pabotilla voitiin ajaa ainakin kuusi rinnakkaista testiprosessia, kunnes kannettavan tietokoneen prosessointiteho ja muistimäärä ei enää riittänyt.



Kuva 9. Testien rinnakkaisajon suoritus Pabotilla.

Tämän testin tarkoituksena oli kokeilla Robot Frameworkin soveltuvuutta stressitestaukseen ja sen toteutus jätettiin varhaiselle tasolle. Havaittiin, että se soveltuu stressitestaukseen ainakin pienimuotoisesti. Todellisuudessa stressitestaus toteutettaisiin jatkuvan integroinnin ohjelmistolla, jotka mahdollistavat myös testien rinnakkaisen suorituksen tai varsinaiseen stressitestaukseen tarkoitetulla työkalulla. Testien samanaikaisen suorituksen avulla voidaan saavuttaa merkittävää hyötyjä, koska testejä voidaan suorittaa useita samanaikaisesti näin säästäen testien suoritukseen tarvittavaa aikaa.

5.3.5 Testidatan syöttö Excel-tiedostosta

Koska aikaisemmat testit sisältävät kovakoodattuja testipotilaita, tehtiin testi joka ottaa potilaan tiedot niitä luovasta Excel-tiedostosta. Näin saadaan testi, joka voitaisiin ottaa käyttöön ja lisätä jatkuvan integraation järjestelmään. Excel-tiedosto generoi 50 erilaista potilasta, jotka sisältävät riskilaskentaan tarvittavat tiedot. Taulukossa 1 on esimerkkinä potilastiedot viidestä luodusta potilaasta, jotka voidaan syöttää ohjelman käyttöliittymään.

Taulukko 1. Esimerkki Excel-tiedoston luomista potilastiedoista.

Tunnus	Syntymäaika	Paino	hCGb	PAPP-A	CRL1	CRL2	NT1	NT2
20171030 124001	19970525	99,2	226,28	10736,82	70,0		0,76	
20171030 124002	19800620	63,6	272,17					
20171030 124003	19770525	91,6		2656,75	44,7	44,6	1,65	0,9
20171030 124004	19721127	71,2		8514,31	55,0		4,17	
20171030 124005	19900716	105,1						

Excel-tiedostosta saatujen arvojen avulla saadaan onnistunut riskilaskennan tulos näkyviin, vaikka sen tarjoamat mitattujen arvojen määrät vaihtelevat. Tämän esimerkin viimeisestä potilaasta ei ole saatavilla riittävästi mitattuja arvoja, jotta riskilukuja voitaisiin laskea, mutta potilaan tiedot saadaan silti syötettyä ja tallennettua onnistuneesti. Excel-tiedosto luo näiden tietojen lisäksi myös muita riskilaskentaan vaikuttavia potilastietoja, kuten potilaan etnisyyden ja tiedon tupakoinnista.

Excel-tiedoston lukua toteutettiin ensiksi Robot Frameworkiin löytyvällä ExcelLibrary-kirjastolla. Tämän kirjaston käytössä vaadittiin kuitenkin Excel-tiedostossa olevien päivämäärien muokkaamista ja tiedostopäätteen muuttamista kirjaston lukurajoitteiden takia. Ratkaisuksi päädyttiin toteuttamaan oma kirjasto Pythonilla, joka hyödyntää pandas-työkalua.

Toteus aloitettiin kirjoittamalla Python-kirjasto, joka lukee Excel-tiedostosta löytyvät potilastiedot ja muuttaa ne kolumneittain pandas-työkalun tarjoamiksi DataFrameiksi. DataFramet muutetaan ja nimetään arvoja kuvaavalla nimellä Python dictionaryiksi, jotka sisältävät kaikkien potilaiden vastaavat arvot (Ohjelma 6).

```
import pandas

def get_excel_data():
    values = {}

    df =
    pandas.read_excel('C:\Users\linjaapa\PycharmProjects\PerkinElmer-
    LifeCycle-Test-Automation\Add_50samples_random.xlsm',
    sheetname='InputData', keep_default_na=False)

    values[u"PatientID"] = df['txtptnmedicalrecord1'].values
    values[u"LastName"] = df['txtptnSurname'].values
    values[u"FirstName"] = df['txtptnFirstName'].values
    values[u"PtnDateOfBirth"] = df['txtptnDateOfBirth'].values
    values[u"Ethnicity"] = df['cboptnEthnic'].values

    values[u"DaysByUS"] = df['txtsd1DaysByUS'].values
    values[u"Smoking"] = df['Smoking'].values
    values[u"NumberOfFetuses"] = df['txtcd1NumberOfFetuses'].values

    values[u"SampleID"] = df['txtspcextcode1'].values
    values[u"SpcDateCollected"] = df['txtspcTimeCollected '].values
    values[u"Weight"] = df['txtsd1Weight'].values
    values[u"hCGbValue"] = df['ansValuePlain_HCGb'].values
    values[u"PAPP-AValue"] = df['ansValuePlain_PAPP_A'].values

    values[u"USDate"] = df['txtsd1USDate'].values
    values[u"CRL"] = df['txtsd1CRL'].values
    values[u"CRL2"] = df['txtsd1CRL2'].values
    values[u"NT"] = df['ansValuePlain_NT'].values
    values[u"NT2"] = df['ansValuePlain_NT2'].values

    return values
```

Ohjelma 6. Kirjasto Excel-tiedoston lukemista varten.

Testitiedostossa määritetään potilaan rivinumero Excel-taulukossa, jonka sisältämät tiedot syötetään vastaaviin syöttökenttiin ja määritetään uudeksi potilaaksi (Ohjelma 7). Tämän jälkeen lasketaan riskiarvot syötettyjen arvojen avulla.

```
*** Settings ***

| Library | excellib.py
| Library | Selenium2Library
| Library | Collections
| Library | DateTime
| Library | String

*** Variables ***
| ${ExcelData}
| ${Browser} | Chrome
| ${Username} | entryUser
| ${Password} | password
| ${Patient} | 5

*** Test Cases ***
|Fill Patient Information From Excel File
| Login
| Get Excel Test Data
| Navigate To Pregnancy Page
| Fill Patient Data From Excel | ${Patient} | # 1...50 Patient number
| Save Patient And Calculate Risk Values
| Gather Test Evidence
| [Teardown] | Close All Browsers
```

Ohjelma 7. Excel-tiedostoa lukeva testitapaus.

Testikoodin avainsanat sisältävät tyyppimuunnoksia ja jäsennyksiä, jotta taulukosta luetut arvot saadaan syötettyä oikeassa muodossa. Riskilaskentaa muokataan myös riippuen siitä, mitkä riskilaskentaan vaikuttavat arvot taulukko tarjoaa. Näin riskilaskennan tulos saadaan aina näkyviin.

Dynaamiset syöttökentät osoittautuivat haasteellisiksi ja niiden ratkaisuksi käytettiin avainsanaa syöttökentän tarkennuksesta poistumiseksi (Ohjelma 8).

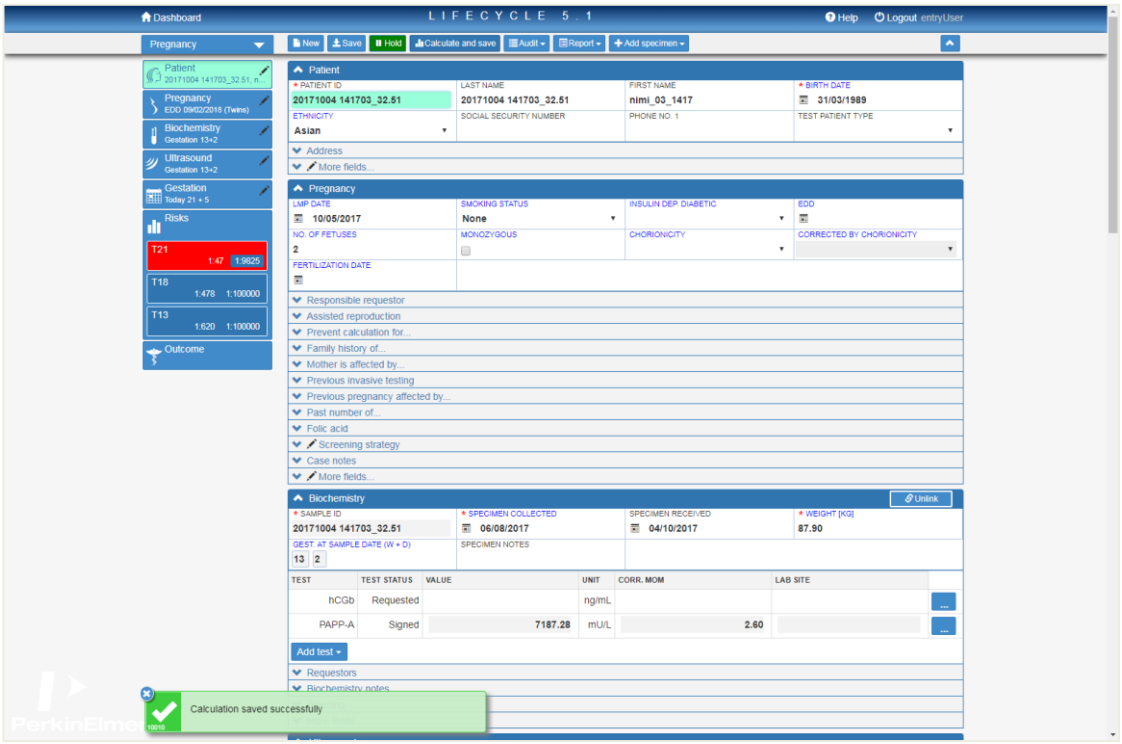
```
| Click Element Out Of Focus
| | Click Element | //*[@id="applicationTitle"]
| | Wait Until Element Is Not Visible | loadingOverlay
```

Ohjelma 8. Avainsana syöttökentän tarkennuksesta poistumiseen.

Avainsana klikkaa käyttöliittymän otsikkoelementtiä ja odottaa latausikonin poistumista näkyvistä, minkä jälkeen voidaan jatkaa kenttien syöttöä.

Testikoodi syöttää vain yhden potilaan ja se vaatii muokkaamista, jos testin halutaan syöttävän useamman potilaan tiedot. Tämän ratkaisu jätettiin kuitenkin tästä työstä, sillä jatkuvan integroinnin ympäristön avulla voidaan testipotilaan syöttö tehdä rinnakkain säästäen testisuoritukseen käytettyä aikaa.

Testitodistukseksi määritettiin kuvankaappaus toteutuneesta riskilaskennasta (Kuva 10). Selenium2Library tarjoaa tämän toiminnallisuuden omana avainsananaan. Kuvankaappauksen nimeä ja tallennuspaikkaa voidaan myös muuttaa, ja niiden nimen perään lisätään indeksi, jotta sama testisuorituksen mahdolliset useat kuvankaappaukset saadaan tallennettua.



Kuva 10. Kuvankaappaus testitodistukseksi läpäistystä testistä.

Kuvasta voidaan huomata tässä testitapauksessa toisella sikiöllä Downin syndrooman riskin kohonneen, jolloin se on merkattu punaisella värillä. Kuvankaappaukset näkyvät myös testiraportissa, jos ne tallennetaan yhdessä samaan kansioon.

Ohjelmakoodissa määritettiin avainsana, joka tallentaa kuvankaappauksen, jos testin suoritus epäonnistuu (Ohjelma 9).

```
| Run On Test Failure
| | #Log Source
| | Capture Page Screenshot | filename=Test_Failure_{index}.png
```

Ohjelma 9. Testisuorituksen epäonnistuessa suoritettava avainsana.

Tämän avainsanan avulla voidaan myös tehdä muita toimintoja, kuten testattavan ohjelman käyttöliittymäsivun lähdekoodin tallentaminen. Kuvankaappauksen ottaminen hidastaa myös testin suoritusaikaa jos testi sisältää sellaisia avainsanoja, jotka voivat epäonnistua, mutta testisuoritusta ei ole määritetty keskeytettäväksi niiden takia.

6 HAVAINNOT

Testausautomaatio ei täysin korvaa manuaalista testausta, koska kaikkea testausta ei yleisesti ottaen voi automatisoida. On siis tärkeää tunnistaa testit, jotka kannattaa automatisoida, ja testit, jotka kannattaa ajaa käsin. Testien ylläpito vaatii resursseja, mutta sen implementoinnilla voidaan säästää aikaa ja rahaa pidemmällä tähtäimellä. Automaattinen testaus ei havaitse kaikkia ongelmia, mutta testitodistuksena otettuja kuvankaappauksia katsomalla voidaan varmistaa testin suorituksen oikeellisuus. Testausautomaatio sopii hyvin regressiotestaukseen, koska niiden suoritus on nopeaa ja tarkkaa.

Robot Frameworkin etuna voidaan pitää testien helppolukuisuutta ja avainsanakirjastojen tarjoamaa monipuolisuutta ja yleistä laajennettavuutta. Testitapaukset ovat ylläpidettäviä jos ohjelmistoon ei tehdä laajoja muutoksia. Robot Frameworkin suorituksen havaittiin olevan nopeaa ja sen luomien testilogien ja kuvankaappauksen avulla voidaan määrittää testin suorituksen epäonnistuessa sen aihettama kohta, vaikka testitapaukseen ei olisi sisällytetty tarkistuksia. Testisuorituksien nopeuttamisella havaittiin olevan lisääntyneitä mahdollisuuksia virheelliseen datansyöttöön syöttökenttien dynaamisuuden ja palvelinvastauksen vuoksi. Testitapausten kirjoituksessa tämän huomionti on tarpeellista, jotta testit kirjoittavat varmasti syöttökenttiin oikean datan, eivätkä jätä syöttökenttiä tyhjäksi, jolloin testisuoritus saatetaan tulkita onnistuneeksi, vaikka testi on todellisuudessa epäonnistunut. Tässä työssä näitä ongelmia ratkaistiin hyödyntämällä osittain jo aiemmin todettuja ratkaisuja, eikä pyritty mahdollisimman vähäisen ajankäytön testisuoritukseen.

Testien kirjoitus vaatii kuitenkin yhteneväisyyttä kehittäjäyhteisön sisällä, jotta avainsanapohjaisen testauksen hyötyjä ei kumota epäjohdonmukaisilla testitapauksilla. Tämän työn tuloksena saatua ohjelmistokoodia on kommentoitu laajalti, jotta testitapauksen suorittamien vaiheiden seuraaminen on selkeämpää. Tämä on kuitenkin jossain määrin huono menettelytapa, koska käytetyistä avainsanoista tulisi suoraan selvitä, mitä ohjelmakoodi tekee. Ohjelmakoodin tarpeeton kommentointi vaikeuttaa testitapausten ylläpitoa ja luo turhaa toistoa.

7 YHTEENVETO

Tässä työssä keskityttiin Robot Frameworkin pilotointiin, ja sen osalta työn tavoitteet tulivat toteutetuiksi. Tekijä syvensi tietämystään testausautomaation käytön ohjelmistojen testaamisen lisäksi myös raskausajan sikiöseulonnasta. Työn aloitusvaiheessa testiympäristö saatiin toimintaan nopeasti ja vaivattomasti, joten testien kirjoitusta voitiin ryhtyä heti tutkimaan käytännön kautta samalla lähdemateriaaliin tutustuen.

Robot Frameworkin erilaisia ominaisuuksia kokeilemalla havaittiin olevan sen kunnollinen ja sopivasti laajennettavissa oleva työkalu. Robot Frameworkin helposti luettavissa oleva syntaksi on hyödyllinen ja testien kirjoittaminen on melko vaivatonta. Sen käyttöön löytyy yksityiskohtainen käyttöohje ja runsaasti opetuksellista materiaalia verkosta sekä konsultointia tarjoavia yrityksiä.

Käyttöliittymätestauksessa testausautomaation lisäksi tarvitaan myös manuaalista tarkastelua, koska automatisoidut testit eivät havaitse mahdollisia epätoivottuja visuaalisia muutoksia. Tämä voidaan toteuttaa esimerkiksi jälkitarkastelemalla testisuorituksen aikana otettuja kuvankaappauksia. Testausautomaatiolla ei siksi voida saavuttaa testauksen täydellistä kattavuutta, vaan se tulee implementoida niihin testauksen osiin, joissa sen avulla saavutetaan hyötyjä.

Työn tuloksena kirjoitettujen testitapauksien ja niistä kerättyjen havaintojen avulla saavutettiin verrattain hyvät lähtökohdat testitapauksien jatkokehittämistä varten. Vaatimuksena määritetty toteutus potilastietojen lukuun Excel-tiedostosta tuotti testitapauksen, jota voitaisiin lähes sellaisenaan hyödyntää oikeassa regressiotestauksen integroinnin vaiheessa.

Seuraava vaihe Robot Frameworkin implementoinnissa olisi sen lisääminen jatkuvan integraation järjestelmään. Tämä jätettiin kuitenkin toteuttamatta työhön määritetyn laajuuden vuoksi.

LÄHTEET

Autti-Rämö, I.; Koskinen, H.; Mäkelä, M.; Ritvanen, A. & Taipale, P 2005. Raskauden ajan ultraäänitutkimukset ja seerumiseulonnat rakenne- ja kromosomipoikkeavuuksien tunnistamisessa. Helsinki: Stakes/FinOHTA. http://www.julkari.fi/handle/10024/76012.

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

Hass, A. M. J. 2008. Guide to advanced software testing. Norwood, Massachusetts: Artech House.

Homes, B. 2013. Fundamentals of Software Testing. London: Wiley.

ISTQB Exam Certification 2017. What is a proof-of-concept or piloting phase for tool evaluation in software testing? Viitattu 16.10.2017 http://istqbexamcertification.com/ > Proof-of-concept or piloting phase for tool evaluation.

Klärck, P. 2017. Experience. Viitattu 5.9.2017 http://www.eliga.fi/ > Experience.

Korpela, M. 2015. Speeding up test execution with Pabot. Viitattu 16.10.2017 http://hereberobots.blogspot.fi/2015/02/speeding-up-test-execution-with-pabot.html.

Myers, G.J.; Sandler, C. & Badgett, T. 2012. The art of software testing. 3. painos. Hoboken, New Jersey: John Wiley & Sons.

Pandas 2017. Python Data Analysis Library. Viitattu 1.11.2017 http://pandas.pydata.org.

PerkinElmer 2017. Overview. Viitattu 6.9.2017 http://www.perkinelmer.com/fi/lifecycle > Overview.

Robot Framework 2017. Introduction. Viitattu 5.9.2017 http://www.robotframework.org/ > Introduction.

Telles, M. A 2007. Python Power!: The Comprehensive Guide. Boston, Massachusetts: Thomson Course Technology PTR.

Väestöliitto 2017. Sikiön kromosomipoikkeavuuksien seulontatutkimukset. Viitattu 30.10.2017 https://www.vaestoliitto.fi/perinnollisyys/perinnollisyysneuvonta/sikiotutkimukset/sikion_kromosomipoikkeavuuksien/.

Wallin, M. 2017. Web application test automation with Robot Framework Viitattu 5.9.2017 https://www.ruleoftech.com/2013/web-application-test-automation-with-robot-framework#python.