



# **AUTOMATISOIDUN TOIMINNALLISEN TESTAUKSEN KEHITTÄMINEN**

Juho Hautaniemi

Opinnäytetyö  
Maaliskuu 2015  
Tietojenkäsittely  
Ohjelmistotuotanto

## TIIVISTELMÄ

Tampereen ammattikorkeakoulu  
Tietojenkäsittelyn koulutusohjelma  
Ohjelmistotuotanto

HAUTANIEMI, JUHO:

Automatisoidun toiminnallisen testauksen kehittäminen

Opinnäytetyö 54 sivua, joista liitteitä 8 sivua  
Maaliskuu 2015

---

Opinnäytetyön toimeksiantaja, Jakamo Osakeyhtiö, kehittää pilvipalveluna toimivaa toimitusketjun hallintajärjestelmää, Jakamoa. Vielä startup-vaiheessa olevalla yrityksellä tuotteen testaukseen riittävät resurssit ovat erittäin rajalliset, minkä takia tuotekehityksen laadunvarmistusta halutaan parantaa toiminnallista testausta automatisoimalla. Opinnäytetyön tavoitteena oli selvittää, miten testausprosessia voitaisiin lähteä automatisoimaan.

Jakamo on pilvipalvelu, jota käytetään pääasiassa selaimella. Opinnäytetyöprojektin aikana siitä oli saatavilla myös Windows Phone- ja Android-sovellukset ja iOS-laitteilla toimiva versio oli jo kehitteillä. Opinnäytetyön tarkoituksena oli löytää mahdollisimman hyvin Jakamon tarpeisiin sopiva testaustyökalu sekä tutustua testien toteuttamiseen sillä.

Opinnäytetyön tutkimusmenetelmien lähestymistapoina käytettiin tapaustutkimusta sekä konstruktivistista tutkimusta. Sen lopputuloksena löydettiin toimeksiantajan tarpeisiin sopiva sovelluskehys, Robot Framework, jolla toiminnallisia testejä voidaan automatisoida. Lisäksi työssä toteutettiin muutamia testejä Jakamon kehitysympäristössä ja saatiin käsitys siitä, millä käytännön toimenpiteillä Jakamolle voitaisiin luoda kattava ja kustannustehokas automatisoitu toiminnallisen testauksen testausympäristö.

## **ABSTRACT**

Tampereen ammattikorkeakoulu  
Tampere University of Applied Sciences  
Business Information Systems  
Software Development

HAUTANIEMI, JUHO:  
Developing Automated Functional Testing

Bachelor's thesis 54 pages, appendices 8 pages  
March 2015

---

The company that commissioned this thesis, Jakamo Osakeyhtiö, is developing a supply chain management system which runs as a cloud-based service. Still carrying the status of a startup company, Jakamo has limited resources to spare for software testing. That is the reason the company wants to better their quality assurance process by automatizing functional testing. The objective of this thesis was to find out the initial steps towards automated functional testing.

Jakamo is a cloud service that is used mainly with web browsers. Additionally, mobile applications are available for Android and Windows Phone mobile operating systems and an application for iOS is currently under development. The purpose of the thesis was to find a testing tool that suits Jakamo's test automation needs as well as possible. Moreover, implementing a few tests with the selected tool was another major goal of the thesis.

The research was conducted as a case study as well as a constructive study. As an outcome, a suitable test automating framework, Robot Framework, was found for the client. The study also yielded some example tests and a report about how Jakamo could start setting up a comprehensive and cost-effective testing environment.

---

Key words: software development, software testing, functional testing, cloud service, SaaS

## SISÄLLYS

1	JOHDANTO.....	7
2	OHJELMISTOTESTAUS.....	9
2.1	Ohjelmistotestaus yleisesti.....	9
2.1.1	Ohjelmiston laatu .....	9
2.1.2	Ohjelmistotestauksen eri tasot .....	10
2.1.3	Yksikkötestaus .....	10
2.1.4	Integraatiotestaus.....	11
2.1.5	Järjestelmätestaus.....	11
2.1.6	Hyväksyttämistestaus.....	11
2.1.7	Alfa- ja betatestaus .....	11
2.1.8	Käyttöliittymätestaus.....	12
2.1.9	Regressiotestaus .....	12
2.1.10	Testauksen automatisointi.....	12
2.2	Automatisoitu toiminnallinen testaus .....	13
3	AUTOMATISOIDUN TOIMINNALLISEN TESTAUKSEN KEHITTÄMINEN.....	14
3.1	Testattava järjestelmä .....	14
3.2	Työkalujen etsiminen ja vertailu.....	18
3.3	Työkaluvaihtoehtojen rajaaminen.....	20
3.4	Työkalun valinta .....	22
3.5	Valittu työkalu: Robot Framework .....	24
3.5.1	Yleistä .....	24
3.5.2	Esivaatimukset .....	25
3.5.3	Asentaminen.....	27
3.6	Testien suunnittelu .....	28
3.6.1	Testien rajausta opinnäytetyössä.....	28
3.6.2	Testien suunnitteluprosessi .....	29
3.7	Testien toteutus Robot Frameworkilla.....	32
3.7.1	Syntaksi .....	33
3.7.2	Arkkitehtuuri .....	35
3.7.3	Testien ajaminen .....	37
3.7.4	Testiraportit.....	38
4	TULOKSET .....	41
4.1	Valittu testaustyökalu .....	41
4.2	Esimerkkitestit .....	41
5	POHDINTA.....	43

LÄHTEET .....	45
LIITTEET .....	47
LIITE 1. ”Luo uusi tiedote”-käyttötapauksen testitapaustaulukko .....	47
LIITE 2. Robot Framework-esimerkkitestien koodit .....	49

**ERITYISSANASTO**

.NET	Microsoftin luoma sovelluskehys ohjelmistojen kehittämiseen.
black box-testaus	Mustalaatikkotestaus. Testaustapa, jossa testaaja ei käytä tietoa ohjelmiston sisäisestä toiminnasta testien toteuttamisessa.
IDE	Integrated Development Environment, integroitu kehitysympäristö.
IronPython	Pythonin .NET-toteutus.
Jython	Pythonin Java-toteutus.
pip	Sovellus, jolla voidaan ladata ja asentaa Python-paketteja PyPistä.
PyPi	Python Package Index, Python-pakettihakemisto.
Python	Yleisesti käytetty tulkattava ohjelmointikieli.
reStructuredText	
SaaS	Software as a Service.
SUT	System Under Test, testattava järjestelmä.
TSV	Tab-separated Values. Tapa, jolla dataa voidaan muotoilla taulukoksi.
white box-testaus	Valkolaatikkotestaus. Testaustapa, joka keskittyy erityisesti ohjelmiston sisäisen toiminnan testaamiseen.

## 1 JOHDANTO

Ohjelmistotestauksen tarkoituksena on varmistaa, että testattava ohjelmisto vastaa sille asetettuja vaatimuksia, jotka jaotellaan yleisesti toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Opinnäytetyön toimeksiantajalla, Jakamo Osakeyhtiöllä, oli tarve lähteä kartoittamaan heidän tuotteensa toiminnallisten vaatimusten testauksen automatisointimahdollisuuksia, koska nykyisellään pelkästään manuaalisesti tehtävä testaus vie valtavasti resursseja. Opinnäytetyön tavoitteena oli selvittää, miten testausprosessia voitaisiin lähteä automatisoimaan.

Jakamo Osakeyhtiö kehittää Jakamo-nimistä pilvipalveluna toimivaa toimitusketjun hallintajärjestelmää. Palvelua käytetään ensisijaisesti selaimella, mutta siitä on saatavilla myös mobiilisovellukset Androidille ja Windows Phonelle. IOS-sovellus tullaan julkaisemaan keväällä 2015. Opinnäytetyön tarkoituksena oli etsiä Jakamon käyttämiä alustoja tukeva työkalu toiminnallisen testauksen automatisointiin ja toteuttaa sillä muutamia esimerkkitestejä. Työn tutkimusmenetelminä käytettiin tapaustutkimusta ja konstruktivistista tutkimusta.

Opinnäytetyön ensimmäisessä vaiheessa perehdyttiin tämän hetken monialustaisiin testaustyökaluihin. Keskeisimpänä tavoitteena oli saada kokonaiskuva nykyaikaisista sovelluksista, joilla automatisoituja toiminnallisia testejä voitaisiin toteuttaa mahdollisimman helposti ja kustannustehokkaasti monelle alustalle. Ensisijaisena tiedonlähteenä käytettiin internetiä, koska kirjallisuudesta on mahdotonta saada ajan tasalla olevaa tietoa nopeasti kehittyvistä ohjelmistoista. Hakujen tuloksena löydettyjä testaustyökaluja vertailtiin toimeksiantajan kannalta tärkeillä kriteereillä. Vertailun lopputuloksena päädyttiin valitsemaan Robot Framework, jolla on mahdollista testata websovelluksia Selenium 2 -kirjastoa käyttämällä. Android- ja iOS-sovellusten testaaminen onnistuu Robot Frameworkin Appium-kirjastoa käyttämällä.

Opinnäytetyön toisen vaiheen tavoitteena oli varmistaa valitun työkalun soveltuvuus toimeksiantajan tarpeisiin toteuttamalla sillä muutamia esimerkkitestejä. Tätä varten tutustuttiin myös testien suunnitteluprosesseihin, missä tärkeimpänä lähteenä käytettiin Yogesh Singh'n Software testing -kirjaa. Työssä esitellään käyttötapauksiin perustuva testien suunnitteluprosessi, jota myös sovelletaan toteutusvaiheessa.

Toteutuksen tärkeimpänä tavoitteena oli perehtyä valittuun testaussovelluskehikseen, Robot Frameworkiin, niin hyvin että testit voidaan toteuttaa sillä. Opinnäytetyön raporttiin koottiin Robot-testien toteuttamisen kannalta olennaista tietoa kuten esimerkiksi esivaatimukset, asennusohjeet, syntaksi sekä testien rakenne, raportit ja ajaminen. Toteutuksessa tärkeimpänä lähteenä käytettiin Robot Frameworkin sekä Selenium 2 -kirjaston omia dokumentaationsivustoja.

Opinnäytetyön tuloksena toimeksiantajalle löydettiin sopiva työkalu toiminnallisten testien automatisointiin ja toteutettiin joitakin testejä työkalulla. Työn lopputuloksia ja jatkokehitysehdotuksia arvioidaan raportin viimeisessä luvussa.



## 2 OHJELMISTOTESTAUS

### 2.1 Ohjelmistotestaus yleisesti

#### 2.1.1 Ohjelmiston laatu

ISO-9126-standardissa määritellään muutamia ohjelmiston laatutekijöitä: toimivuus, luotettavuus, käytettävyys, tehokkuus, ylläpidettävyys ja uudelleenkäytettävyys. Ohjelmiston laatu on kuitenkin asia, joka koetaan näkökulmasta riippuen eri tavalla. (Tian 2005, 18–19) Yleisesti ohjelmistoihin liittyy kaksi eri sidosryhmää, jotka näkevät hyvin erilaiset ominaisuudet laadun kannalta tärkeinä. Nämä ryhmät ovat ohjelmiston kuluttajat ja tuottajat. (Tian 2005, 16)

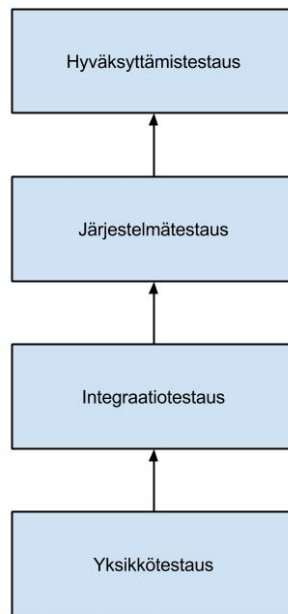
Jos ohjelmistoja ajatellaan yksinkertaisina laatikoina, kuluttajat näkevät ne niin sanottuina mustina laatikoina (*black box*), mikä tarkoittaa, että heillä on käsitys, miltä laatikko näyttää ja miltä sen tulisi näyttää ulospäin. Laatikon sisällöstä kuluttajat eivät tiedä mitään, eikä heidän useimmissa tapauksissa tarvitsekaan. Ohjelmiston laatukäsitysten kannalta voi olla aiheellista jakaa kuluttajat vielä kahteen eri ryhmään: asiakkaat ja käyttäjät. Asiakas voi olla samalla myös käyttäjä, ja päinvastoin. Asiakas hankkii ohjelmiston, kun taas käyttäjä itse asiassa käyttää sitä, joten on selvää, että nämä kaksi ryhmää saattavat ymmärtää sen laadun keskenään hyvinkin eri tavalla. (Tian 2005, 16)

Yleisesti kuluttajat määrittävät ohjelmiston laadukkuuden muutaman ominaisuuden perusteella: toimivuus, luotettavuus, käytettävyys ja käyttöönoton helppous. Mainittujen ulkoisten ominaisuuksien painoarvo riippuu jälleen paljolti käyttäjästä – kokenut käyttäjä saattaa pitää luotettavuutta tärkeänä, kun taas kokematon näkee helppokäyttöisyyden merkittävämpänä. (Tian 2005, 17)

Ohjelmiston tuottajiin kuuluvat kaikki, jotka jollain tavalla ovat osana sen kehitystä. Tämä ryhmä näkee tuotteensa läpinäkyvänä laatikkona (*white box*), mikä tarkoittaa, että heillä on tietoa myös sen teknisestä sisällöstä. Tämän ryhmän eri jäsenet voivat nähdä ohjelmiston tärkeimpinä laatutekijöinä roolistaan riippuen esimerkiksi käytettävyyden, markkina-arvona, koodin ylläpidettävyyden tai sen uudelleenkäytettävyyden. (Tian 2005, 17)

### 2.1.2 Ohjelmistotestauksen eri tasot

Ohjelmistotestauksen eri vaiheet voidaan yleisesti jakaa eri tasoihin, joilla kullakin on omat tavoitteensa. Tällaiset tasot ovat yksikkö-, integraatio-, järjestelmä- ja hyväksyttämistestaus. (Singh 2012, 368) Tasot on kuvattu kuviossa (1).



KUVIO 1. Ohjelmistotestauksen neljä tasoa

### 2.1.3 Yksikkötestaus

Yksikkötestauksella tarkoitetaan yksittäisten pienten koodinpätkien testaamista (Black 2002, 5). Yleisesti tällaisista koodinpätkistä käytetään nimityksiä funktio, metodi tai moduuli, mutta ohjelmistotestauksen yhteydessä puhutaan yksiköistä. Yksikkötestit voidaan suunnitella joko valko- tai mustalaatikkomallin mukaisesti. Käytännössä lähes aina myös ohjelmoija itse tekee yksikkötestausta kirjoittamalleen koodille (Black 2002, 5). Yksikkötestauksessa ongelmaksi voi muodostua se, että yksiköitä ei välttämättä ole tarkoitettu toimimaan toisistaan irrallaan, jolloin ne saattavat sisältää kutsuja muihin yksiköihin. Tällaisissa tapauksissa riippuvuudet korvataan tynkämoduuleilla ja ajureilla. (Singh 2012, 369)

#### 2.1.4 Integraatiotestaus

Integraatiotestauksen tavoitteena on varmistaa, että toisiinsa kytkeytyneet koodikokonaisuudet ja niiden väliset rajapinnat toimivat oikein. Jos kuitenkin testattava ohjelmisto koostuu toisistaan täysin irrallisista kokonaisuuksista, ei integraatiotestausta välttämättä tarvita lainkaan. (Black 2002, 6) Mitä heikommin yksiköt ovat kytkeytyneet toisiinsa, sitä vähemmän integraatiotestausta yksiköiden väliset rajapinnat tarvitsevat (Singh 2012, 371).

#### 2.1.5 Järjestelmätestaus

Järjestelmätestauksessa ohjelmistoa testataan sen oletetussa toimintaympäristössä. Tässä vaiheessa varmistetaan, että järjestelmä toimii halutulla tavalla. Lisäksi sen ei-toiminnallisia vaatimuksia kuten esimerkiksi suorituskykyä, luotettavuutta tai turvallisuutta testataan. (Singh 2012, 373) Mikäli testattava järjestelmä (*system under test eli SUT*) koostuu ohjelmiston lisäksi myös laitteista, tulee myös niitä käyttää järjestelmätestauksessa.

#### 2.1.6 Hyväksyttämistestaus

Hyväksyttämistestauksen tarkoituksena on, että tilaaja varmistaa ohjelmiston vastaavan vaatimusmäärittelyä (Singh 2012, 373). Tässä vaiheessa testausta voivat olla tekemässä myös ohjelmiston loppukäyttäjät, mutta yleensä tätä ei pidetä ohjelmiston toimittajan tehtävänä (Myers ym. 2012, 131). Mikäli testattavalla ohjelmistolla ei varsinaisesti ole asiakasta, ei voida puhua hyväksyttämistestauksesta. Tällöin kyseessä on alfa- ja beta-testaus. (Singh 2012, 373–374)

#### 2.1.7 Alfa- ja betatestaus

Alfa- ja betatestauksessa on kyse pohjimmiltaan samasta asiasta, kuin hyväksyttämistestauksessa. Hyväksyttämistestaus tehdään, mikäli ohjelmisto on toteutettu jollekin tietylle asiakkaalle, kun taas alfa- ja betatestaus tehdään, jos ohjelmistolla ei ole yksilöitävää asiakasta. (Singh 2012, 374)

Alfatestausessa ohjelmiston toimittajan testausryhmä hankkii pienen joukon kohderyhmän loppukäyttäjiä tekemään testausta. Betatestauksessa alfavaiheen testaajamäärää suurempi ryhmä loppukäyttäjiä testaa ohjelmistoa ilman sen toimittajan valvontaa. (Singh 2012, 374)

### **2.1.8 Käyttöliittymätestaus**

Singh'n (2012, 458) mukaan käyttöliittymätestauksessa testataan, että käyttöliittymäelementit toimivat odotetulla tavalla ja että sovellus käsittelee hiiren ja näppäimistön syötteet oikein. Hän jakaa web-sovelluksen käyttöliittymätestauksen kahteen kategori-  
aan: navigointitestaukseen ja lomakepohjaiseen testaukseen (Singh 2012, 458–459).

### **2.1.9 Regressiotestaus**

Kun järjestelmää päivitetään tai siihen luodaan uusia ominaisuuksia, jotkut sen vanhat ominaisuudet saattavat lakata toimimasta. Tämä pyritään huomaamaan regressiotestauksessa, jossa ajetaan järjestelmän vanhat testit ja pyritään varmistumaan siitä, että koko järjestelmä toimii yhä päivityksen jälkeenkin. (Tamres 2002, 223)

### **2.1.10 Testauksen automatisointi**

Yleisesti ottaen kaikki ohjelmistotestaus, käytettävyydestausta lukuun ottamatta, voidaan automatisoida. Tämä on erityisen järkevää silloin, kun testien ajaminen ja ylläpitäminen on lähes ilmaista ja testejä toistetaan tiheään tahtiin. Tällöin automatisoinnista kertyneet korkeat kustannukset tulevat maksamaan itsensä takaisin. (Black 2007, 42)

## 2.2 Automatisoitu toiminnallinen testaus

Kuten lähes kaikki ohjelmistotestauksen tyypit, myös toiminnallinen testaus voidaan automatisoida. Tyypillisesti tähän on olemassa kaksi menetelmää: testaajan käyttöliittymään tekemien toimintojen tallentaminen tai niiden koodaaminen uudelleen ajettaviksi testitapauksiksi.

Toiminnallisen testauksen automatisointiin on olemassa laaja valikoima erilaisia työkaluja ja sovelluskehyskiä, jotka vaihtelevat ominaisuuksiltaan lähinnä tuettujen alustojen ja testien toteutustavan osalta. Monet nykyaikaiset testaustyökalut muistuttavat ohjelmistokehittäjille tuttuja integroituja kehitysympäristöjä: ne tukevat automaattista avainsanojen syöttöä sekä omien testiensä ajamista suoraan käyttöliittymästä käsin. On melko yleistä, että testejä pystytään toteuttamaan tallentamalla käyttäjän toiminnot, kirjoittamatta yhtään koodia. Tällöinkin testaustyökalut luovat toimintoja vastaavan koodin, joka voidaan myöhemmin ajaa. Useimmat työkalut myös tarjoavat mahdollisuuden muokata ohjelman luomaa koodia tai vaikka kirjoittaa sitä käyttämättä lainkaan tallennusominaisuutta. Joissain työkaluissa tallennusominaisuutta ei puolestaan ole lainkaan, vaan testejä toteutetaan ainoastaan koodia kirjoittamalla.

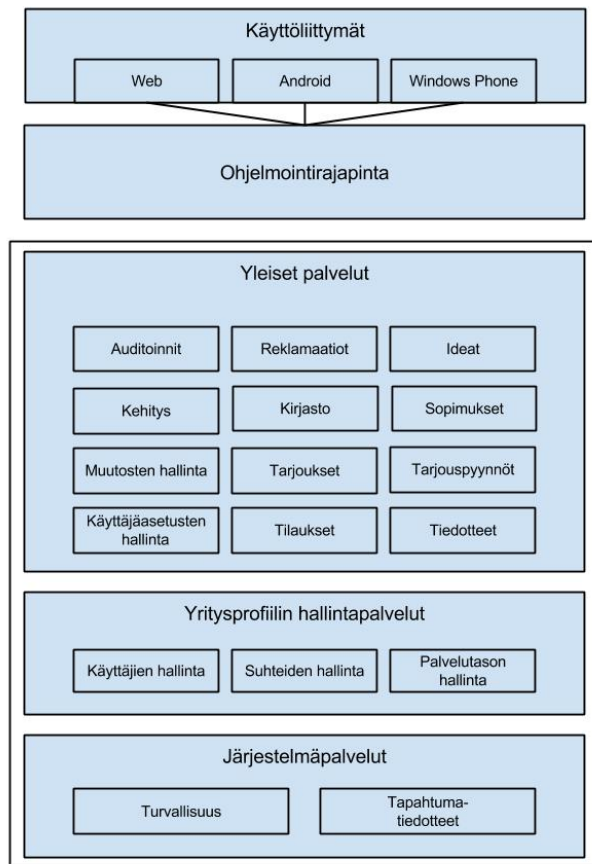
### 3 AUTOMATISOIDUN TOIMINNALLISEN TESTAUKSEN KEHITTÄMINEN

Opinnäytetyön tarkoituksena on kartoittaa toiminnallisen testauksen automatisointimahdollisuuksia toimeksiantajan kehittämälle verkkopalvelulle, Jakamolle, ja tutustua testien toteuttamiseen valitulla menetelmällä. Tässä luvussa käydään läpi käytännön toimenpiteet, jotka tehtiin opinnäytetyön tavoitteeseen pääsemiseksi. Jakamolla luvun tekstissä tarkoitetaan Jakamo-palvelua eikä samannimistä toimeksiantajayritystä. Kaikki kuvankaappauksissa näkyvä tieto on kuvitteellista testidataa.

#### 3.1 Testattava järjestelmä

Opinnäytetyössä käsitellään Jakamon toiminnallisen testauksen automatisoinnin kehittämistä. Jakamo on SaaS- eli pilvipalvelu.

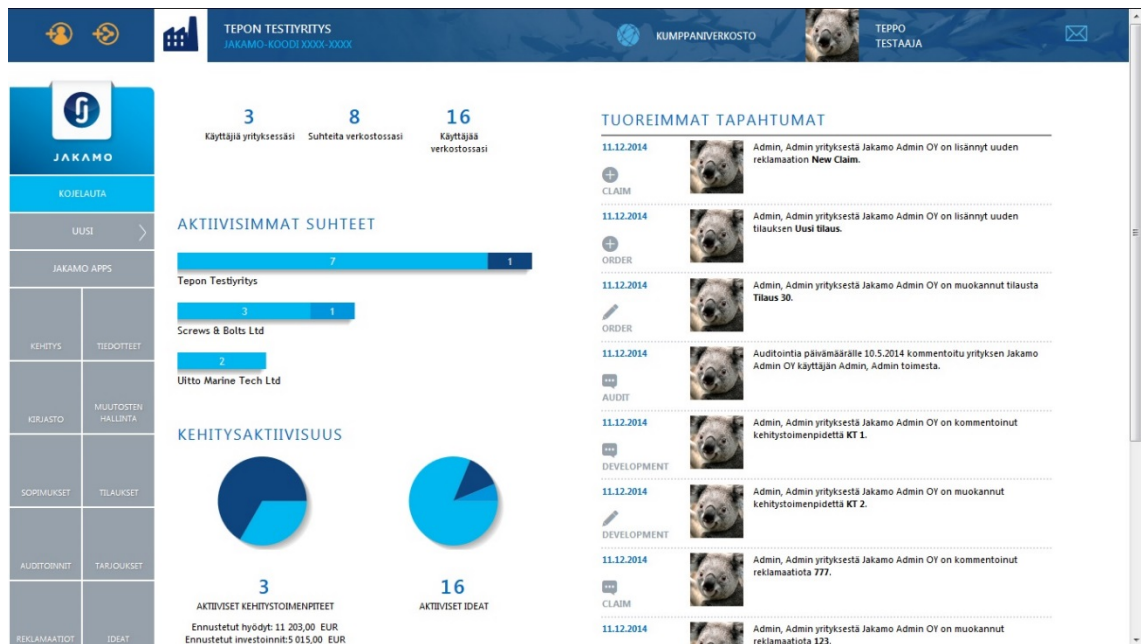
Kuviossa (2) on esitetty Jakamon yksinkertaistettu palveluarkkitehtuuri nykytilassaan. Keskeisimmät palvelut ovat auditoinnit, ideat, kehitys, kirjasto, muutosten hallinta, rek-lamaatiot, tarjoukset, tarjouspyynnöt, tilaukset ja tiedotteet. Jakamon sisällä edellä mainittuja palveluita nimitetään sovelluksiksi. Kukin Jakamo-sovellus koostuu sisällön luonnista, listauksesta ja muokkauksesta. Jokainen Jakamoon luotu merkintä, kuten esimerkiksi yksi tilausmerkintä, sisältää jakamisasetukset, jotka määrittävät mille muille yrityksille kyseinen merkintä näkyy. Jakaminen edellyttää, että yritykset ovat luoneet suhteen toisiinsa.



KUVIO 2. Jakamon palveluarkkitehtuuri

Koska Jakamon merkittävin käyttöalusta on selain, keskitytään tässä opinnäytetyössä käsittelemään Jakamon selainversiota testattavana järjestelmänä.

Kuvassa (1) näkyy Jakamon selainkäyttöliittymän päänäkymä. Kiinteitä elementtejä käyttöliittymässä ovat yläreunassa näkyvä sininen palkki sekä vasemmassa reunassa näkyvä harmaista painikkeista koostuva valikko. Yläpalkista käyttäjä pääsee hallinnoimaan omia asetuksiaan, ja lisäksi yrityksen ylläpitäjäasemassa olevat käyttäjät pääsevät käsiksi yrityksen asetuksiin sen kautta. Vasemman reunan valikon painikkeista päästään muun muassa edellisessä kappaleessa mainittujen sovellusten listanäkymiin. Yleisnäkymää ei ole syytä käydä tässä raportissa sen tarkemmin läpi.



KUVA 1. Jakamon selainkäyttöliittymän yleisnäkymä

Kuvassa (2) nähdään esimerkki Jakamo-sovellusten listanäkymästä. Näkymä koostuu aina ”lisää uusi”-painikkeesta, listan merkintöjen suodatusvalikosta sekä itse merkintälistasta. Lisäksi Auditoinnit-, Reklamaatiot-, Tarjoukset-, Tilaukset- ja Tilauspyynnöt-sovellusten listanäkymät sisältävät painikkeet, joista listan merkinnät voidaan vaihtaa saapuneiden ja lähetettyjen merkintöjen välillä. Kuvan listanäkymä on Tilaukset-sovelluksesta. Listan riviä klikkaamalla yksittäistä merkintää päästään tarkastelemaan tarkemmin.

The screenshot shows the 'Tilaukset' (Orders) application in Jakamo. The top navigation bar includes icons for user, company, and network, along with the text 'JAKAMO ADMIN OY' and 'KUMPPANIVERKOSTO'. The left sidebar contains a menu with options like KOJELAUTA, UUSI, JAKAMO APPS, KEHITYS, TIEDOTTEET, KIRJASTO, MUUTOSTEN HALLINTA, SOPIMUKSET, TILAUKSET, AUDITOINNIT, TARJOUKSET, REKLAMAATIOT, and IDEAT.

The main content area displays a list of orders with filters and a table of order details:

- LISÄÄ UUSI:** A button to add a new order.
- SUODATUSVAIHTOEHDOT:** A section with filters for 'Suhte' (All), 'Tila' (Mikä tahansa), and 'Etsi' (Search). There is also a 'Näytä' button and a 'Oma tili' dropdown.
- Lähetetyt / Saapuneet:** Two buttons to toggle between 'Sent' and 'Received' views.
- SAAPUNEET TILAUKSET:** A table showing a list of orders with columns for 'Tilausnumero', 'Luontiaika', 'Edit time', 'Asiakas', 'Tila', 'Työnumero', 'Projektivaihe', 'Pyydetty pvm', 'Vahvistettu pvm', and 'Toiminnot'.

KUVA 2. Jakamon Tilaukset-sovelluksen saapuneiden merkintöjen listanäkymä



Kuvassa (3) näkyy yksittäisen tilausmerkinnän tiedot. Kunkin sovelluksen merkinnät sisältävät joitakin samoja yleisen tason tietoja kuten esimerkiksi otsikon ja kuvauksen.

KUVA 3. Yksittäisen tilausmerkinnän tarkastelunäkymä

Kuvassa (4) nähdään esimerkki Jakamon uuden merkinnän luontinäkömystä selainkäyttöliittymässä. Näkömässä olevat kentät riippuvat sovelluksesta; kuvassa (4) oleva näkömä on Tilaukset-sovelluksesta. Näkömän ”näytä rivit”-painikkeen takaa käyttäjä pääsee lisäämään ostotilausrivejä tilausmerkintään. ”Liitteet”-kohdasta merkintään voidaan liittää tiedostoja ja ”tagit”-kohdassa tunnisteita, joiden perusteella merkintöjä voidaan suodattaa listanäkymissä.

KUVA 4. Jakamon Tilaukset-sovelluksen uuden merkinnän luontinäkömä

### 3.2 Työkalujen etsiminen ja vertailu

Toiminnallisen testauksen automatisointiin on saatavilla lukuisia eri valmistajien työkaluja. Koska ohjelmistokehityksessä käytettävät ja suosittavat tekniikat vaihtelevat suhteellisen nopeaan tahtiin, oli luontevin tietolähde työkaluvaihtoehtojen etsinnälle internet. Haussa käytettiin muun muassa seuraavia hakusanoja: automated testing tool, automated testing, automated functional testing tool ja cross-platform testing tool. Hakutuloksista rajautui tarkempaan tutustumiseen yhteensä 11 testaustyökalua: Froglogic Squish, Ranorex, eggPlant, Robot Framework, TestComplete, Twist, Telerik TestStudio, Selenium, Appium, Sahi sekä Sauce Labs. Rajaamisessa tärkein kriteeri oli Jakamon käyttämien alustojen tukeminen. Vertailuun valituista työkaluista kaikki paitsi Selenium ja Appium tukevat web-, iOS- ja Android-sovellusten testaamista (Selenium, 2015; Appium, 2015a), mutta ne otettiin kuitenkin mukaan vertailuun, sillä Appium pohjautuu Seleniumiin (Appium, 2015b), minkä johdosta testit toteutetaan näillä kahdella työkalulla hyvin samankaltaisesti. Toimeksiantajan budjettiin sopivia jokaista alustaa tukevia työkaluja ei etsinnän tuloksena löytynyt; useimpien repertuaarista puuttui Windows Phone -mobiilialustan tukeminen.

Aluksi löydettyjä 11 työkalua vertailtaessa käytettiin kriteereinä hintaa, kirjastojen dokumentaation laatua ja referenssejä. Lisäksi huomioon otettiin Jakamolla käytössä olevan CI-palvelimen, Jenkinsin, tukeminen, mikä löytyikin lähes kaikista.

Hintahaarukka asettui valittujen työkalujen kohdalla kohtuullisen laajaksi: Robot Framework, Selenium ja Appium ovat ilmaisia (Appium 2015a; Selenium 2015), kun taas useimmat maksulliset lisenssit olivat kertaveloitteisia ja maksoivat 700 eurosta muutamaa tuhansiin euroihin. Vuotuisia lisenssejä tarjosivat muun muassa Sahi (Sahi, 2015). Telerikilta olisi ollut saatavilla myös lisenssejä kuukaudeksi kerrallaan (Telerik, 2015).

Niin ikään testauskirjastojen dokumentaation laadussa ja saatavilla olevan tukimateriaalin tarjonnassa oli eri työkalujen kesken todella suuria eroja. Maksullisiin lisensseihin kuului poikkeuksetta valmistajan tarjoama käyttäjätuki, ja lisäksi tarjolla oli vaihtelevassa määrin myös muuta ohjemateriaalia kuten esimerkiksi videoita ja blogikirjoituksia. Lähes kaikilla valituista työkaluista oli myös oma fooruminsa, mutta näidenkin aktiivisuuksien välillä oli suuria eroja. Ohjelmistoalalla eräs suosittu yhteisön tuen lähde on stackoverflow.com (Alexa, 2015), josta tarkastin myös kunkin vertailukohteen aktiivisuuden. Yleisiä tukiominaisuuksia näillä kriteereillä arvioimalla edukseen esiintyivät Selenium ja Ranorex, joihin löytyi muihin verrattuna erityisen hyvin tukimateriaalia. Seleniumista tarjolla oli selvästi eniten hakutuloksia stackoverflow.com:ssa ja sen dokumentaatio oli myös selkeä. Ranorexilla puolestaan oli vertailukohteista selvästi aktiivisin foorumi, mikä yhdistettynä maksulliseen lisenssiin kuuluvaan käyttäjätukeen ja selkeästi dokumentoituihin kirjastoihin luo hyvät edellytykset päästä eteenpäin mahdollisesta ongelmatilanteesta.

Yleisesti ottaen automatisoitujen käyttöliittymätestien toteutustapa on joko toimintojen tallentaminen graafisen käyttöliittymän kautta tai sitten skriptien kirjoittaminen. Vertailtavista työkaluista Robot Framework ja Appium tukivat ainoastaan skriptaamista – muilla vaihtoehdoilla testit pystyi toteuttamaan sekä käyttöliittymän kautta tallentamalla että kirjoittamalla.

### **3.3 Työkaluvaihtoehtojen rajaaminen**

Koska kaikkiin 11 työkaluun perusteellisesti tutustuminen ei olisi ollut tarkoituksen mukaista eikä ajallisesti mahdollistakaan, halusin rajata vaihtoehtoja jo alkuvaiheessa. Rajaamisessa käytin taulukon (1) osoittamia valintakriteereitä, joiden mukaan pisteytin kunkin työkalun. Kukin ominaisuus on pisteytetty välillä 1-5. Taulukoiden selkeyden takia niissä kuvataan ainoastaan pienintä ja suurinta pistemäärää vastaavat ominaisuudet.

TAULUKKO 1. Valintakriteerit testaustyökaluvaihtoehtojen rajausvaiheessa

Pisteet	1	5
Hinta (€/vuosi)	>3000	Ilmainen
Dokumentaatio	Todella epäselvä ja puutteellinen testauskirjastojen dokumentaatio	Todella selkeästi dokumentoidut testauskirjastot
Valmistajan tuki	Ei valmistajan tarjoamaa tukea	Rajaton ja ilmainen käyttäjätuki
Yhteisön tuki: foorumien aktiivisuus, blogikirjoitusten määrä yms.	Ei lainkaan yhteisön tarjoamaa tukea	Kiitettävä
Alustojen tukeminen (Web, Android, Windows Phone ja iOS)	Yksi mobiilialusta	Web, Android, Windows Phone ja iOS

Tässä vaiheessa kriteerit koostuivat ominaisuuksista, joita pystyttäisiin luotettavasti arvioimaan päällisin puolin työkaluun tutustumalla. Koska tässä vaiheessa ensisijaisena tietolähteenä toimivat työkalujen omat osaltaan markkinointiin suunnitellut nettisivut, tuli lähteisiin suhtautua riittävän kriittisesti. Pistetulokset on kirjattu taulukkoon (2). Eniten kokonaispisteitä saaneet työkalut on korostettu taulukossa punaisella värillä.

TAULUKKO 2. Testaustyökaluvaihtoehtojen pisteytetyt ominaisuudet ja kokonaispisteet

Työkalun nimi	Hinta	Dokumentaatio	Valmistajan tuki	Yhteisön tuki	Alustojen tukeminen	Yhteensä
Appium	5	3	1	3	2	14
eggPlant	1	4	3	2	4	14
Froglogic Squish	1	4	2	2	4	13
Ranorex	3	5	5	4	4	21
Robot Framework	5	5	1	5	4	20
Sahi	4	3	3	2	2	14
Sauce Labs	4	3	5	2	4	18
Selenium	5	4	1	5	2	17
Telerik Test Studio	3	4	4	3	5	19
TestComplete	2	2	4	4	4	16
Twist	5	4	4	2	4	19

Käytettyjen valintakriteereiden perusteella parhailta vaihtoehdoilta vaikuttivat Ranorex ja Robot Framework, jossa tultaisiin käyttämään sille saatavilla olevia Selenium- ja Appium-liitännäisiä. Tällä tavoin Robot Frameworkilla pystyttäisiin kirjoittamaan testejä selaimelle, Androidille ja iOS:lle, kuten Ranorexillakin pystytään tekemään. Tässä vaiheessa käytettyjen valintakriteereiden perusteella merkittävimmät erot näiden kahden työkalun välillä ovat hinta ja testien toteutustapa.

### 3.4 Työkalun valinta

Ensimmäisen työkalujen rajausvaiheen jälkeen esittelin tulokset ja omat näkemykseni eri työkaluvaihtoehdoista toimeksiantajalle, minkä tuloksena yhdessä päätimme valita toisen vaiheen tarkempaan tutustumiseen Ranorexin ja Robot Frameworkin. Toisessa vaiheessa käytin valintakriteereinä taulukon (3) mukaisia ominaisuuksia.

TAULUKKO 3. Valintakriteerit testaustyökaluvalinnan toisessa vaiheessa

Pisteet	1	5
Hinta (€/vuosi)	>3000	Ilmainen
Alustojen tukeminen (Web, Android, Windows Phone ja iOS)	Yksi mobiilialusta	Web, Android, Windows Phone ja iOS
Selainten tukeminen	Selaintuessa suuria puutteita	Tukee laajasti yleisimpiä selaimia ja niiden uusia sekä vanhoja versioita
Dokumentaatio	Todella epäselvä ja puutteellinen testauskirjastojen dokumentaatio	Todella selkeästi dokumentoidut testauskirjastot
Valmistajan tuki	Ei valmistajan tarjoamaa tukea	Rajaton ja ilmainen käyttäjätuki
Yhteisön tuki: foorumien aktiivisuus, blogikirjoitusten määrä yms.	Ei lainkaan yhteisön tarjoamaa tukea	Kiitettävän paljon yhteisön tarjoamaa tukea
Tukimateriaali	Ei lainkaan opetusvideoita ja -kirjoituksia	Opetusmateriaalia on saatavilla kiitettävän paljon ja monipuolisesti
Oppimiskynnys	Alkuun pääseminen vaatii merkittävän paljon oppettelua	Selkeät aloitusohjeet ja helposti omaksuttava syntaksi sekä testauskirjastot
Kehitysympäristö	Ei lainkaan integroitua kehitysympäristöä saatavilla	Työskentelyä selvästi tehostava integroitu kehitysympäristö

Aikaisemman vertailuvaiheen kriteerit koostuivat työkaluihin tutustumatta saataviin tietoihin. Seuraavassa vaiheessa otin mukaan myös ominaisuuksia, jotka vaativat työkalujen asentamista ja hiukan syvempää tutustumista.

Vertailun viimeisessä vaiheessa Ranorex ja Robot Framework saivat yhtä paljon pisteitä, kuten taulukosta (4) selviää. Valintakriteereistä hinta ja oppimiskynnys koettiin kuitenkin painoarvoiltaan niin suuriksi, että vertailun lopputuloksena päädyttiin Robot Frameworkiin. Ranorex sisältää todella laajat testauskirjastot ja monipuolisen kehitysympäristön, minkä takia sen käytön oppitteluun tulisi käyttää merkittävästi aikaa, jotta testien toteuttaminen onnistuisi tehokkaasti. Myös Ranorexin käyttämän RanoreXPath-viittausjärjestelmä koettiin toimeksiantajan kehitystiimissä hankalaksi.

TAULUKKO 4. Testaustyökaluvalinnan viimeisen vaiheen pisteet

	Ranorex	Robot Framework
<b>Hinta</b>	3	5
<b>Alustojen tukeminen</b>	4	4
<b>Selainten tukeminen</b>	5	5
<b>Dokumentaatio</b>	5	5
<b>Valmistajan tuki</b>	4	1
<b>Yhteisön tuki</b>	5	5
<b>Tukimateriaali</b>	5	5
<b>Oppimiskynnys</b>	3	5
<b>Kehitysympäristö</b>	5	4
<b>Yhteensä</b>	<b>39</b>	<b>39</b>

### 3.5 Valittu työkalu: Robot Framework

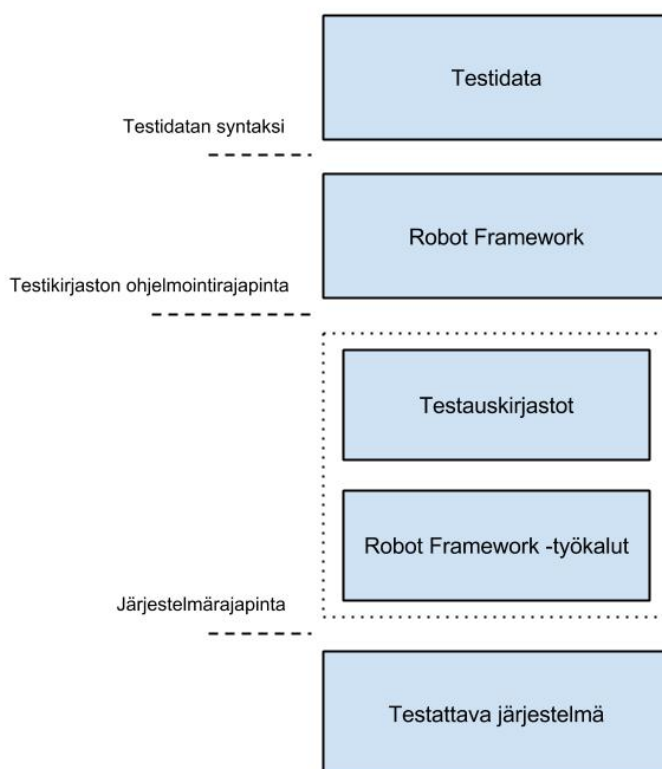
Edellisessä luvussa mainittujen perustelujen nojalla käytettäväksi testaustyökaluksi päätettiin valita Robot Framework, jonka ohessa voitaisiin tarvittaessa käyttää Appium- ja Selenium-liitännäisiä testien kirjoittamiseksi web-, Android- ja iOS-alustoille. Tässä luvussa kerron hiukan tarkemmin Robot Frameworkista.

#### 3.5.1 Yleistä

Aikaisemmin tässä opinnäytetyössä työkalulla on tarkoitettu ohjelmistoa tai sovellusta, jolla testejä voidaan toteuttaa ja ajaa. Robot Frameworkin yhteydessä työkaluilla kuitenkin tarkoitetaan Robotin lisäosia.

Robot Framework on avoimen lähdekoodin ilmainen hyväksyttämistestauksen automatisointisovelluskehys, jonka kehitystä tukee Nokia Networks. Robot Frameworkin rakenne on modulaarinen, minkä johdosta se on joustavasti muokattavissa käyttäjien tarpeisiin erilaisia testauskirjastoja ja -työkaluja käyttämällä. (Robot Framework, 2014a) Sovelluskehysen modulaarista rakennetta on kuvattu kuviossa (3).





KUVIO 3. Robot Frameworkin modulaarinen rakenne

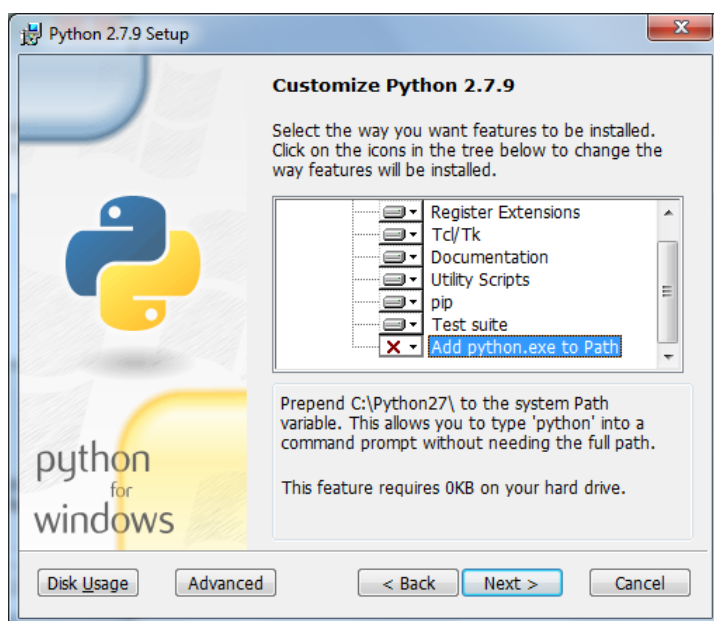
Robot Frameworkissa on esiasennettuna muutamia testauskirjastoja ja -työkaluja, ja myös kolmannet osapuolet kehittävät niitä aktiivisesti. Tällaisia testauskirjastoja ovat esimerkiksi Android-, iOS- ja Selenium-kirjastot. (Robot Framework, 2014b) Omien testauskirjastojen kehittäminen on myös mahdollista. Robotin työkaluilla voidaan esimerkiksi tuottaa raportteja, lokeja tai dokumentaatiota, tai liittää Robot joihinkin yleisesti käytössä oleviin integroituihin kehitysympäristöihin kuten esimerkiksi Eclipseen. (Robot Framework, 2014c)

### 3.5.2 Esivaatimukset

Robot Frameworkin esivaatimuksena on ainoastaan se, että tietokone pystyy ajamaan Python-ohjelmointikielellä toteutettua sovelluskehystä. Tämä edellyttää Pythonin, Jythonin tai IronPythonin asentamista. (Robot Framework, 2014a) Jython on Python-toteutus Java-alustalle (Python, 2014a), kun taas IronPython on vastaava toteutus .NET-sovelluskehykselle. (IronPython, 2014) Opinnäytetyössä asensin tietokoneelleni Pythonin, jonka asennusohjelma on saatavilla Pythonin [www](http://www.python.org)-sivuilta, osoitteesta

www.python.org. Sivustolta on saatavilla sekä Python 2:n että Python 3:n eri versioita, joista uusimmat olivat asennushetkellä 3.4.2 ja 2.7.9. Näistä Python 3.4.2 on luonnollisesti uudempi, mutta sivujen mukaan kannattaa valita Python 2, mikäli jokin käytettävä sovellus ei vielä tue uudempaa versiota. (Python, 2014b) Itse asensin ensin Python 3:n, mutta jouduin myöhemmin vaihtamaan sen vanhempaan versioon, koska halusin kokeilla Robot Frameworkille tehtyä integroitua kehitysympäristöä, RIDE:ä, joka ei toiminut Python 3:lla.

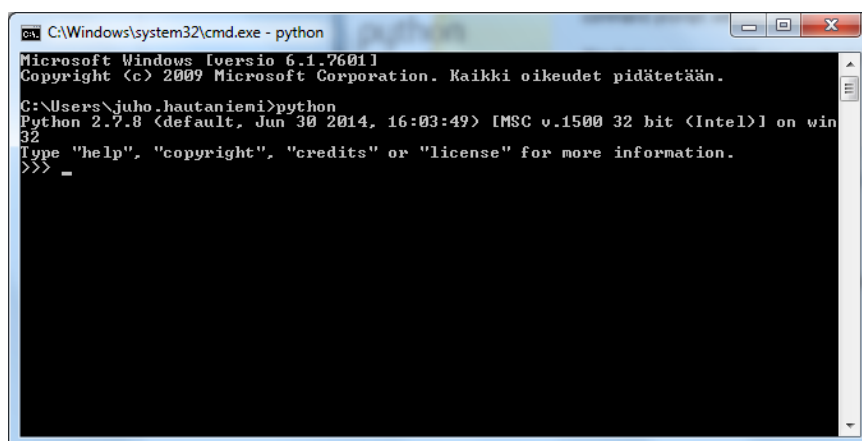
Pythonin asentaminen on hyvin suoraviivainen prosessi: aiemmin mainitulta sivustolta ladataan halutun version asennusohjelma, jonka ajamalla Python asentuu. Asennuksessa on suositeltavaa varmistaa, että pip on valittu asennettavaksi. Pip on ohjelma, jolla voidaan hallita Python-paketteja ja asentaa niitä lisää suoraan Python Package Indexistä, eli PyPI:stä. (PyPI, 2014) Myös Robot Frameworkin asennus onnistuu pip:ä käyttämällä. Lisäksi asennusvaiheessa on hyvä harkita, haluaako asennusohjelman lisäävän Pythonin ympäristömuuttujiin automaattisesti. Tämän salliminen vaatii asennusohjelman ajamista järjestelmänvalvojan käyttäjäoikeuksilla. Kuvassa (5) näkyy asennettavat Python-ominaisuudet, joista kohdan ”Add python.exe to Path” valitsemalla Python lisätään ympäristömuuttujiin.



KUVA 5. Pythonin lisääminen ympäristömuuttujiin asennusvaiheessa

Käytettävyyden kannalta ympäristömuuttujiin lisääminen on järkevää, koska tällöin Python-komennot voidaan tehdä komentoriviltä helpommin.

Kuvassa (6) komentoriville syötetään pelkästään komento *python*, mikä käynnistää Python-komentorivin. Mikäli ympäristömuuttujiin lisäämistä ei olisi tehty, tulisi asennusohjelman asentama *python.exe*-tiedosto ajaa viittaamalla suoraan siihen. Joka tapauksessa asennuksen jälkeen on hyvä todentaa, että se suoritettiin onnistuneesti, mikä voidaan tehdä käynnistämällä Python, kuten kuvassa (6) on tehty. Kun Python on asennettu, voidaan siirtyä itse Robot Frameworkin asentamiseen.

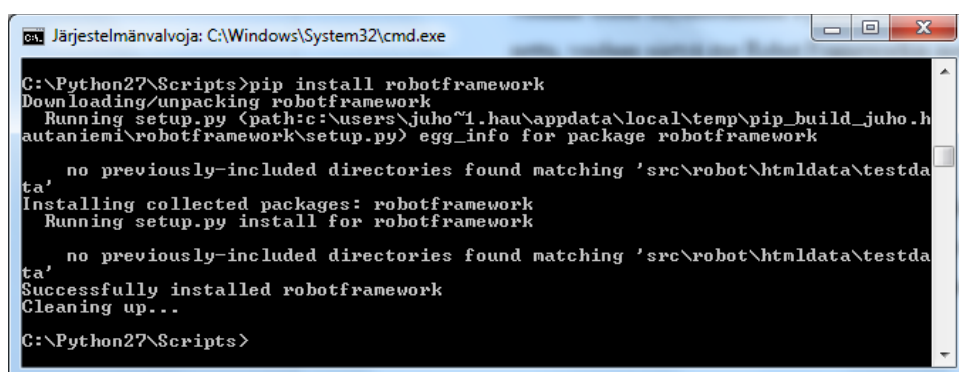


KUVA 6. Ympäristömuuttujiin lisätyn Python-komennon kutsuminen komentoriviltä

### 3.5.3 Asentaminen

Robot Framework voidaan asentaa joko *pip*:ä käyttämällä tai kääntämällä se suoraan lähdekoodista. Opinnäytetyössä *pip*:n asentaminen tehtiin Pythonin asentamisen yhteydessä. Tämä käytiin läpi edellisessä osiossa. *Pip*:llä asentaminen onnistuu syöttämällä sen *install*-komennolle parametri *robotframework*. (Python Software Foundation, 2014)

Kuvassa (7) komentorivillä ajetaan komento *pip install robotframework*, joka asentaa Robot Frameworkin. Kuvasta nähdään, että asennus suoritettiin onnistuneesti.



KUVA 7. Robot Frameworkin asentaminen *pip*:llä

Opinnäytetyön testeissä tullaan käyttämään Robot Frameworkin Selenium 2-kirjastoa, jonka voi myös asentaa pip:ä käyttämällä komennolla *pip install robotframework-selenium2library*.

### 3.6 Testien suunnittelu

Testien toteutuksen pohjana voidaan käyttää esimerkiksi tilakaavioita, päätöstauluja tai käyttötapauksia. (Tamres 2011. 117-118, 124, 150-151) Vaikka toimeksiantajan kehitysprosessiin ei kuulu edellä mainittujen dokumenttien luominen, opinnäytetyön järjestelmätason toiminnallisten testien toteutuksen pohjaksi näistä luontevimmin sopivat käyttötapaukset. Tässä luvussa käydään läpi käyttötapausten perusteella tehty testien suunnitteluprosessi.

#### 3.6.1 Testien rajaus opinnäytetyössä

Aikataulusyistä opinnäytetyöhön ei voida sisällyttää testattavan järjestelmän laajaa testausta, koska tehtäviin kuului olennaisena osana myös eri testaustyökalujen vertailu ja niistä parhaiten käyttötarkoitukseen soveltuvan valinta. Samasta syystä myös mobiilisovellusten testit joudutaan rajaamaan työn ulkopuolelle, vaikka mobiilialustojen tukeminen otettiin huomioon testaustyökalujen vertailussa. Opinnäytetyön aikarajoissa on mahdollista suunnitella yksi kohtuullisen yksinkertainen testitapaus sekä suorittaa se automatisoidusti Robot Frameworkia käyttämällä. Testien kohteeksi valitaan käyttötapaus ”Luo uusi tiedote”.

Taulukossa (4) on esitetty opinnäytetyön testien kohteena olevan käyttötapauksen kuvaus, jossa on käytetty Haikalan (2011, 80) käyttämää esimerkkiä käyttötapauksen kuvaamisesta.

TAULUKKO 4. Käyttötapauksen "Luo uusi tiedote" kuvaus

<b>Nimi:</b>	Luo uusi tiedote
<b>Osallistujat:</b>	Peruskäyttäjä tai yrityksen ylläpitäjäkäyttäjä
<b>Tuloehdot:</b>	Käyttäjä on kirjautunut sisään Jakamoon ja hän on tiedote-luontisivulla.
<b>Kuvaus:</b>	Käyttäjä täyttää lomakkeeseen uuden tiedotteen tiedot. Käyttäjä tallentaa tiedotteen klikkaamalla ”lähetä”-painiketta. Järjestelmä tallentaa tiedotteen ja siirtää käyttäjän juuri luodun tiedotteen näkymään.
<b>Poikkeukset:</b>	Käyttäjä ei täytä lomakkeen kaikkia vaadittuja kenttiä: otsikko, kuvaus ja jakaminen. Käyttäjä yrittää liittää tiedotteeseen liian suuren tiedoston.
<b>Lopputulos:</b>	Uusi tiedote on tallennettu järjestelmään ja käyttäjä on siirretty uuden tiedotteen näkymään.
<b>Muut vaatimukset:</b>	

### 3.6.2 Testien suunnitteluprosessi

Yogesh Singh (2012, 296) jakaa käyttötapauksiin perustuvien testitapausten luomisen kuuteen eri vaiheeseen:

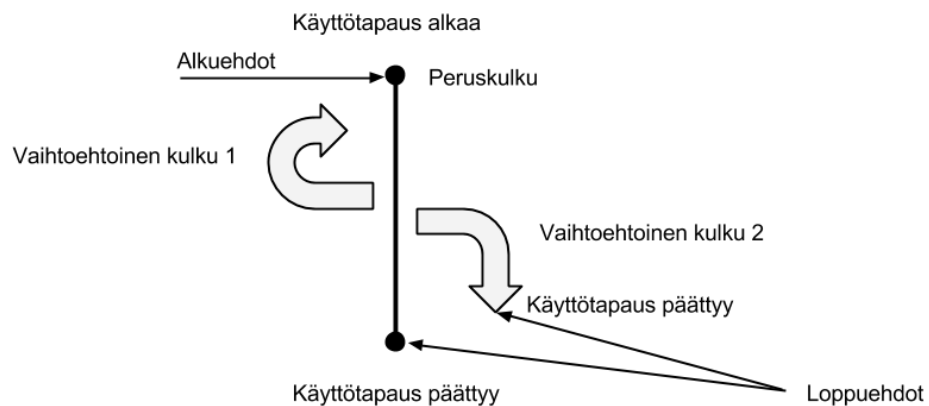
1. käyttötapauksen kulkukaavion luominen
2. käyttötapauksiaulukon luominen
3. käyttötapauksen muuttujien tunnistaminen
4. muuttujien syötteiden tilojen tunnistaminen
5. testitapaustaulukon suunnittelu
6. muuttujien arvojen määrittely

Käyttötapauksen kulkukaaviosta käy ilmi kaikki toiminta, joka sisältyy käyttötapaukseen.

Kuviossa (4) on esimerkki käyttötapauksen kulkukaaviosta. Kuvion keskellä kulkeva suora viiva kuvaa käyttötapauksen peruskulkua. Viivasta poikkeavat nuolet kuvaavat tapauksen vaihtoehtoisia kulkuja, jotka voisivat esimerkiksi poiketen jakautua useam-

paankin haaraan. Oikealla puolella oleva 90-asteisen mutkan tekevä nuoli kuvaa sitä, että vaihtoehtoinen kulku saavuttaa loppuehdon ja päättää käyttötapausten. Viivan päissä olevat pisteet kuvaavat tapaukselle asetettuja alku- ja loppuehtoja, jotka on määritelty käyttötapausten dokumentoinnissa.

Kuvion (4) mukaisen kulkukaavion perusteella käyttötapauksesta luodaan taulukko, jonka tarkoituksena on edelleen havainnollistaa vaihtoehtoisia kulkureittejä ja helpottaa varmistamaan, että kaikki vaihtoehdot tulevat otetuksi huomioon.



KUVIO 4. Yleisesimerkki käyttötapausten kulkukaaviosta

Taulukkoon (5) on koottu kuvion (4) esimerkkikäyttötapausten mahdolliset eri kulkureitit. Tässä kohtaa käytetty esimerkkitapaus on erittäin suppea, minkä takia taulukko näyttää erityisen yksinkertaiselta. Käyttötapaus voisi olla huomattavasti monimutkaisempi, mikä korostaisi taulukon hyödyllisyyttä.

Edellä esitetty kuvio (4) ja taulukko (5) pätevät myös ”Luo uusi tiedote”-käyttötapaukseen, kun kulut nimetään sopivalla tavalla.

TAULUKKO 5. Kuvion (4) kulkukaaviota vastaava käyttötapaustaulukko

Skenaario 1	Peruskulku 1	
Skenaario 2	Peruskulku 1	Poikkeus 1
Skenaario 3	Peruskulku 1	Poikkeus 2

Taulukkoon (6) on kirjoitettu ”Luo uusi tiedote”-käyttötapausten poikkeustilanteiden kuvaukset. Myös peruskulut voitaisiin kuvata samalla tavalla.

TAULUKKO 6. "Luo uusi tiedote"-käyttötapausten poikkeusten nimeäminen

<b>Luo uusi tiedote</b>	
Poikkeus 1	Käyttäjä ei syötä kaikkia pakollisia tietoja
Poikkeus 2	Käyttäjä poistuu tallentamatta tiedotetta

Taulukossa (7) käyttötapausten tietoja on edelleen täydennetty. Pohjana käytetään taulukossa (5) esitettyä mallia.

TAULUKKO 7. "Luo uusi tiedote"-käyttötapausten käyttötapaustaulukko

<b>Skenaario 1 – Luo uusi tiedote</b>	Peruskulku 1	
<b>Skenaario 2 – Luo uusi tiedote, poikkeus: pakollisia tietoja puuttuu</b>	Peruskulku 1	Poikkeus 1
<b>Skenaario 3 – Luo uusi tiedote, poikkeus: käyttäjä poistuu tallentamatta tiedotetta</b>	Peruskulku 1	Poikkeus 2

Jotta käyttötapaus saadaan saatettua testitapaukseksi ja edelleen valmiiksi suoritettavaksi testiksi, tulee seuraavaksi tunnistaa tapauksessa käytettävät muuttujat. Jakamossa tiedotteen pakollisia muuttujia ovat otsikko, kuvaus ja jakamisasetukset. Jakamisasetukset koostuvat kohdista ”sisäinen jakaminen”, ”suhteet” ja ”koko verkosto”, joista yksi täytyy olla valittuna. Muita muuttujia ovat tiedotteen piilotusasetukset, liitetiedostot, liitteiden näkyvyysasetus, liitteiden poistoasetus, tiedotteen tunnisteet sekä julkaisu- ja päättymisaika.

Taulukossa (8) on kuvattu osa ”Luo uusi tiedote”-käyttötapausten testitapaustaulukkoa, joka sisältää useita testitapauksia. Testitapaustaulukon testien voidaan ajatella myös muodostavan testiryhmän. Taulukko löytyy kokonaisuudessaan liitteistä (liite 1).

TAULUKKO 8. Ote ”Lue uusi tiedote”-käyttötapausten testitapaustaulukosta

ID	Skenaarion nimi ja kuvaus	Title *	Description *	**		
				IsInternal	Relationship[]	IsPublic
TT1.1	Skenaario 1 - Luo uusi tiedote	Tammikuun asiakastapaamiset	Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015	TRUE		
TT1.2	Skenaario 1 - Luo uusi tiedote	Tammikuun asiakastapaamiset	Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015		1, 2, 3, 4	
TT1.3	Skenaario 1 - Luo uusi tiedote	Tammikuun asiakastapaamiset	Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015			TRUE
TT2.1	Skenaario 2 – Luo uusi tiedote, poikkeus: pakollisia tietoja puuttuu		Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015	TRUE		
TT2.2	Skenaario 2 – Luo uusi tiedote, poikkeus: pakollisia tietoja puuttuu	Tammikuun asiakastapaamiset		TRUE		
TT2.3	Skenaario 2 – Luo uusi tiedote, poikkeus: pakollisia tietoja puuttuu	Tammikuun asiakastapaamiset	Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015			
* = pakollinen muuttuja, ** = yhdellä sarakkeen muuttujista oltava arvo						

### 3.7 Testien toteutus Robot Frameworkilla

Tässä luvussa käydään läpi edellisessä luvussa esitellyn ”Luo uusi tiedote”-käyttötapausten testien toteutus, ajaminen sekä testausraportit. Opinnäytetyö ei kata kokonaisen testausarkkitehtuurin suunnittelua ja rakentamista. Robot Framework tukee testien toteuttamista avainsana-, tieto- sekä käytöspäätteisesti, joista opinnäytetyössä käytetään avainsanaperusteista toteutustapaa.



### 3.7.1 Syntaksi

Robot Frameworkissa testit kirjoitetaan taulukkomuotoon, ja se tukee HTML-, TSV-, reStructuredText- ja ”pelkkä teksti”-muotoilua. Pelkkää tekstiä käytettäessä testitaulukoiden sarakkeet voidaan erottaa välilyönnein tai pystyviivamerkkiä käyttämällä. (Robot Framework, 2014d) Testeissä käytettävä muotoilutapa riippuu ainoastaan testaajien mieltymyksistä. Opinnäytetyössä käytetään pelkkää tekstiä, välilyönti sarakeerottimena.

Robot-testit koostuvat tiedostoista, jotka sisältävät erilaisia taulukoita, kuten asetus-, muuttuja-, testitapaus- ja avainsanataulukoita. Asetustaulukoissa voidaan määritellä esimerkiksi käytettävät testauskirjastot tai ulkoiset resurssi- ja muuttujatiedostot. Muuttujataulukossa puolestaan määritellään tiedostossa tarvittavat muuttujat. Testitapaustaulukko sisältää nimensä mukaisesti testitapaukset, jotka puolestaan koostuvat avainsanataulukoissa esitellyistä avainsanoista. Robot Frameworkin avainsana muistuttaa hyvin paljon funktiota tai metodia, joka suorittaa jonkin yksittäisen tehtävän. Avainsanoille voidaan antaa parametreja ja ne voivat palauttaa arvoja. (Robot Framework, 2014d)

Koodiesimerkissä (1) nähdään, miltä asetus- ja muuttujataulukot voivat näyttää. Kukin taulukko täytyy olla otsikoitu oikein, jotta Robot osaa lukea sen sisällön. Otsikko erotetaan käyttämällä vähintään yhtä asteriskia otsikon molemmin puolin. Koodiesimerkissä (1) otsikko on erotettu kolmella asteriskilla molemmin puolin.

```

1  *** Settings ***
2  Documentation      A resource file with reusable keywords and variables.
3  ...
4  ...                The system specific keywords created here form our own
5  ...                domain specific language. They utilize keywords provided
6  ...                by the imported Selenium2Library.
7  Library            Selenium2Library
8
9  *** Variables ***
10 ${SERVER}          localhost
11 ${BROWSER}         Firefox
12 ${DELAY}           1
13 ${VALID EMAIL}     test@example.com
14 ${VALID PASSWORD}  test
15 ${FRONT PAGE URL}  http://${SERVER}
16 ${LOGIN URL}       http://${SERVER}/login
17 ${NEW BULLETIN URL} http://${SERVER}/bulletins/new
18

```

KOODIESIMERKKI 1. Esimerkki Robot Frameworkin asetus- ja muuttujataulukoista

Esimerkissä on käytetty välilyöntejä sarake-erottimina. Robot tunnistaa sarakkeen vaihtumisen vähintään kahdesta välilyönnistä, jolloin sarakkeita on mahdollista muotoilla helppolukuisemmiksi (Robot Framework, 2014d). Koodiesimerkin (1) muuttujataulukossa on kahdeksan riviä ja kaksi välilyönnein erotettua saraketta.

Robot Frameworkissa käytetään kahden tyyppisiä muuttujia: skalaari- ja listamuuttujia. Skalaarimuuttujia merkitään `${<muuttujan nimi>}`-syntaksilla. Koodiesimerkissä (1) `${SERVER}`-muuttuja pitää sisällään merkkijonon "localhost". Koodissa määriteltävien muuttujien lisäksi Robotissa on mahdollista käyttää tietokoneen ympäristömuuttujia sekä Javan järjestelmäominaisuuksia (system property), mikäli testejä ajetaan Jyt-honilla. (Robot Framework, 2014d)

Koodiesimerkissä (2) määritellään listamuuttuja `@{RELATIONSHIPS}`, joka pitää sisällään arvot 1, 43 ja 57. Listamuuttujan eri alkioita voidaan kutsua yleisesti ohjelmoinnista tutulla tavalla: esimerkiksi `@{RELATIONSHIPS}[0]` si `@{RELATIONSHIPS}`-listan ensimmäisen alkion, joka koodiesimerkissä (2) olisi luku 1.

```
12 @{RELATIONSHIPS}      1      43      57
13
```

## KOODIESIMERKKI 2. Listamuuttujan määrittäminen Robot Frameworkissa

Testitapaustaulukko koostuu joukosta testitapauksia, jotka puolestaan ovat lista avainsanakutsuja. Taulukolle ei ole määritelty testitapausten enimmäismäärää, mutta ei ole suositeltavaa, että se sisältäisi yli kymmenen testitapausta (Robot Framework, 2014d). Testitapaustaulukko muodostaa testiryhmän (*test suite*).

Koodiesimerkissä (3) on kuvattu eräs testitapaustaulukko, joka pitää sisällään testitapaukset *New bulletin title missing* sekä *New bulletin description missing*. Otsikkorivin jälkeen kullakin rivillä on yksi avainsanakutsu. Robot Framework suosittelee nimeämään testitapaukset mahdollisimman kuvaavasti, jotta pelkästään otsikon perusteella voidaan päätellä, mitä testitapauksen on tarkoitus tehdä (Robot Framework, 2014d). Avainsanakutsujen ensimmäinen sarake sisältää kutsuttavan avainsanan nimen. Muut sarakkeet sisältävät avainsanalle syötettävät parametrit. Koodiesimerkissä (3) ensimmäisen testitapauksen kuudennella rivillä avainsanalle *sleep* syötetään parametrit *5s*

ja *wait for reply*, mikä tarkoittaa sitä, että testiä ajettaessa pidetään viiden sekunnin tauko. Ensimmäinen parametri kuvaa tauon pituutta ja toinen sen syytä (Robot Framework, 2014d).

```

13
14 *** Test Cases ***
15 New Bulletin Title Missing
16     [Setup]    Open Browser To Login Page And Login
17     Go To New Bulletin Page
18     Insert Item Basic Info    ${EMPTY}    ${DESCRIPTION}
19     Set Item Internal
20     Submit Item
21     Sleep    5s    Wait for reply
22     Location Should Contain    /bulletins/new
23     Page Should Contain Element    css=div.errors
24     Page Should Contain    Title is a required field.
25     [Teardown]    Close Browser
26
27 New Bulletin Description Missing
28     [Setup]    Open Browser To Login Page And Login
29     Go To New Bulletin Page
30     Insert Item Basic Info    ${TITLE}    ${EMPTY}
31     Set Item Internal
32     Submit Item
33     Sleep    5s    Wait for Reply
34     Location Should Contain    /bulletins/new
35     Page Should Contain Element    css=div.errors
36     Page Should Contain    Description for the bulletin is required.
37     [Teardown]    Close Browser
38

```

### KOODIESIMERKKI 3. Testitapaustaulukko Robot Frameworkissa

#### 3.7.2 Arkkitehtuuri

Koska testausprojekti saattaa sisältää todella paljon testiryhmiä, on suositeltavaa, että testit jaetaan eri hakemistoihin ja tiedostoihin. Robot Frameworkissa hakemistorakenteet voivat olla kuinka syviä tahansa. Kukin hakemisto muodostaa ylemmän tason testiryhmän sisällään sijaitsevista testiryhmistä ja -tapauksista. (Robot Framework, 2014d)

Testeissä käytettävät tiedot voidaan jakaa eri tiedostoihin, kuten testitapaus-, muuttuja- ja resurssitiedostoihin. Testitapaustiedostot sisältävät testitapaustaulukon ja siihen sisältyviä tietoja kuten esimerkiksi asetuksia, muuttujia ja avainsanoja. Testitapaustiedostoon kirjoitetut muuttujat ja avainsanat ovat kuitenkin käytettävissä ainoastaan kyseisen tiedoston sisäisesti eivätkä siten ole kovinkaan helposti uudelleen käytettävissä. Tämän takia yleisemmin tarvittavia avainsanoja ja muuttujia kannattaa kirjoittaa erillisiin re-

surssi- ja muuttujatiedostoihin, jotka on mahdollista tuoda eri testiryhmien käyttöön. (Robot Framework, 2014d)

Resurssitiedostot voivat sisältää kaikkia testeissä käytettäviä tietoja, mutta eivät itse testitapauksia. Ne ovat testiryhmien ulkoisia resursseja, joita useampi testitapaus voi hyödyntää.

Koodiesimerkissä (4) erääseen testitapaustiedostoon tuodaan kaksi ulkoista resurssitiedostoa. Esimerkin rivit sisältyvät testitapaustiedoston asetustaulukkoon. (Robot Framework, 2014d)

```

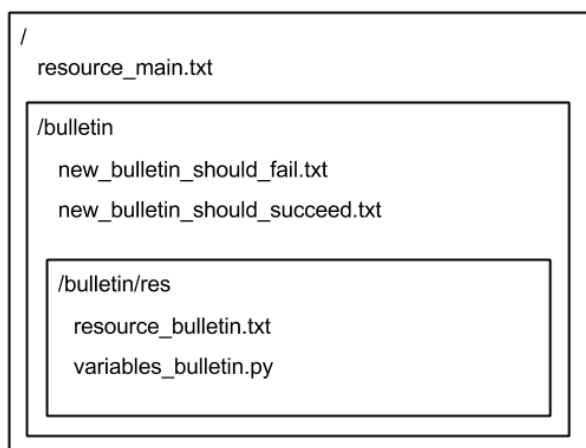
6 Resource      ../resource.txt
7 Resource      res/bulletin_resource.txt
8

```

**KOODIESIMERKKI 4.** Resurssitiedoston tuominen testitapaustiedostoon asetustaulukon kautta

Ulkoisista muuttujatiedostoista on uudelleenkäytettävyyden ja projektin selkeyttämisen lisäksi hyötyä myös siten, että näissä tiedostoissa pystytään määrittelemään minkä tahansa tyyppisiä muuttujia Python- ja Java-ohjelmointikieliä käyttämällä. (Robot Framework, 2014d)

Kuviossa (5) on esitetty opinnäytetyön esimerkkitesteissä käytetty hakemistorakenne. Kun testausprojektia lähdetään laajentamaan, luodaan bulletin-hakemiston rinnalle muita testattavia Jakamo-sovelluksia vastaavat hakemistot. Mikäli projekti alkaa sen paisuessa näyttää sekavalta, voitaisiin sitä mahdollisesti selkeyttää lisäämällä kunkin sovelluksen hakemistoihin suoritettavaa toimintoa vastaavat alihakemistot: *si /bulletin/new* ja */bulletin/edit*.



KUVIO 5. Opinnäytetyön esimerkkitestien hakemistorakenne

Opinnäytetyön esimerkkitestien juurihakemistossa on to *resource\_main.txt*, joka sisältää yleisiä tai kaikille Jakamo-sovelluksille yhteisiä muuttujia ja avainsanoja. Bulletin-hakemisto muodostaa testiryhmän sisältäen testitapaustiedostot *new\_bulletin\_should\_fail.txt* sekä *new\_bulletin\_should\_succeed.txt*. Edellä mainitut tiedostot niin ikään muodostavat testiryhmät sisältämillään testitapauksilla. Hakemistorakenteen selkeyttämiseksi tiedotteisiin liittyvät resurssit on tu */bulletin/res*-hakemiston muuttuja- ja resurssitiedostoihin, jotka sisältävät erityisesti tiedotteille suunnattuja muuttujia ja avainsanoja. Esimerkkitestitiedostojen sisällöt löytyvät kokonaisuudessaan liitteestä (2).

### 3.7.3 Testien ajaminen

Robot Frameworkissa testit ajetaan komentoriviltä *pybot*-, *jybot*- tai *ipybot*-komentosarjaa käyttämällä. Käytettävä komentosarja riippuu siitä, millä Python-tulkilla komentoja ollaan ajamassa; *pybot* toimii Pythonilla, *jybot* Jythonilla ja *ipybot* IronPythonilla. Opinnäytetyössä käytettiin Pythonia, jonka asentaminen käytiin läpi luvussa 3.5.2. Esivaatimukset. Komentosarja löytyy oletuksena Pythonin asennushakemiston alihakemistosta *scripts*.

Kuvassa (8) on suoritettu Robot Framework-testijoukko *bulletin*, joka sisältää alitestijoukot *New bulletin should fail* ja *New bulletin should succeed*. Ajetun testijoukon hakemistorakenne on havainnollistettu myös kuviossa (5). Komentorivin tulosteesta nähdään, että kaikki testit suoritettiin onnistuneesti. (Robot Framework, 2014d)

```

C:\jakamo\jakamo\test>pybot bulletin
=====
Bulletin
=====
Bulletin.New Bulletin Should Fail :: A test suite for testing that a new bu...
=====
New Bulletin Title Missing                                     ! PASS !
New Bulletin Description Missing                             ! PASS !
New Bulletin Sharing Options Missing                         ! PASS !
Bulletin.New Bulletin Should Fail :: A test suite for testing that... ! PASS !
3 critical tests, 3 passed, 0 failed
3 tests total, 3 passed, 0 failed
=====
Bulletin.New Bulletin Should Succeed :: A test suite for testing that a new...
=====
New Bulletin Internal                                       ! PASS !
New Bulletin Shared                                        ! PASS !
New Bulletin Public                                        ! PASS !
Bulletin.New Bulletin Should Succeed :: A test suite for testing t... ! PASS !
3 critical tests, 3 passed, 0 failed
3 tests total, 3 passed, 0 failed
=====
Bulletin
6 critical tests, 6 passed, 0 failed
6 tests total, 6 passed, 0 failed
=====
Output: C:\jakamo\jakamo\test\output.xml
Log:    C:\jakamo\jakamo\test\log.html
Report: C:\jakamo\jakamo\test\report.html
C:\jakamo\jakamo\test>

```

KUVA 8. Robot Framework-testijoukon suorittaminen

Kuvan (8) testiajo on esimerkki kaikkein yksinkertaisimmasta tapauksesta, jossa *pybot*-komentosarjalle syötetään parametrina ainoastaan suoritettavien testijoukkojen nimi. Suoritettavia testitapauksia voidaan kuitenkin valikoida paljon monipuolisemmin oikeita parametreja käyttämällä. Robot Frameworkin testitapausten asetustaulukossa voidaan määritellä tunnisteita, joiden avulla tapauksia voidaan valikoida testejä ajettaessa parametreilla *include* ja *exclude*. Tunnisteiden avulla suoritettavia testejä voidaan myös määrittää kriittisiksi parametreilla *critical* ja *noncritical*. Robotissa kaikki testit ovat oletusarvoisesti kriittisiä, ellei edellä mainittuja parametreja ole käytetty. Lisäksi suoritettavia testijoukkoja voidaan valikoida parametrilla *suite* ja testitapauksia parametrilla *test*. Kaikkien parametrien arvot tukevat yleisesti tunnettuja asteriski- ja kysymysmerkkivillikortteja. (Robot Framework, 2014d)

### 3.7.4 Testiraportit

Kuvan (8) kolme alinta riviä osoittavat ajettujen testien raporttiedostoihin. Oletusarvoisesti Robot Framework kokoaa testien tulokset kolmeen tiedostoon: *output.xml*, *log.html* ja *report.html*. *Report.html* sisältää helppolukuisen yhteenvedon ja linkit yksi-

tyiskohtaisempaan *log.html*-tiedostoon. *Output.xml* pitää sisällään testiajon tiedot raamassa muodossa. (Robot Framework, 2014d)

Kuvassa (9) on kuvankaappaus testiryhmän raporttitiedostosta. Kun kaikki testitapaukset suoritetaan onnistuneesti, raportin taustaväri on vihreä.

## Test Test Report

Generated  
20150207 15:35:20 GMT +03:00  
3 days 6 hours ago

### Summary Information

Status:	All tests passed
Start Time:	20150207 15:30:32.841
End Time:	20150207 15:35:20.811
Elapsed Time:	00:04:47.970
Log File:	<a href="#">log.html</a>

### Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
<a href="#">Critical Tests</a>	6	6	0	00:04:48	<div></div>
<a href="#">All Tests</a>	6	6	0	00:04:48	<div></div>

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
<a href="#">fail</a>	3	3	0	00:02:22	<div></div>
<a href="#">success</a>	3	3	0	00:02:25	<div></div>

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
<a href="#">Test</a>	6	6	0	00:04:48	<div></div>
<a href="#">Test.Bulletin</a>	6	6	0	00:04:48	<div></div>
<a href="#">Test.Bulletin.New Bulletin Should Fail</a>	3	3	0	00:02:23	<div></div>
<a href="#">Test.Bulletin.New Bulletin Should Succeed</a>	3	3	0	00:02:25	<div></div>

### Test Details

Totals

Tags

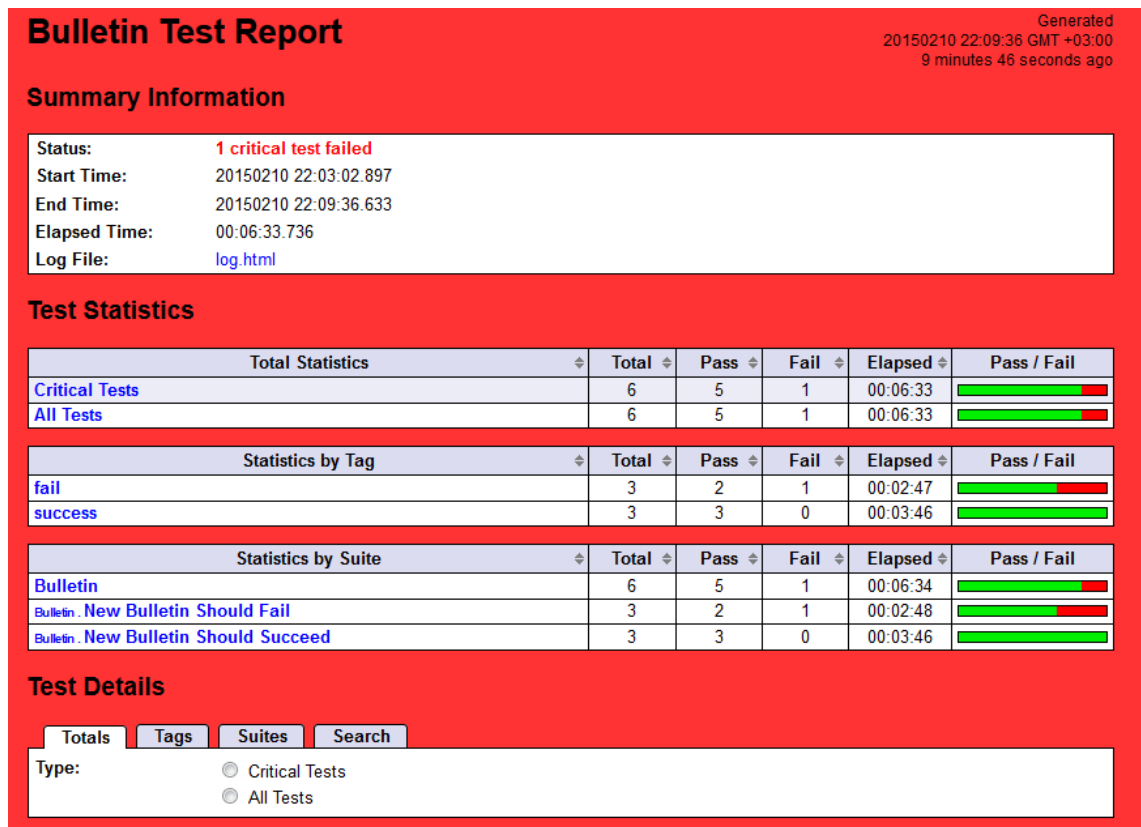
Suites

Search

Type:
☐ Critical Tests
☒ All Tests

KUVA 9. Raporttitiedoston sisältö, kun kaikki testitapaukset ovat onnistuneet

Jos yksikin testitapauksista epäonnistuu, raportin taustaväri näkyy punaisena, kuten kuvassa (10) nähdään. Esimerkkiraportista nähdään, että testiryhmästä *New bulletin should fail* yksi testitapaus on epäonnistunut.



KUVA 10. Robot Frameworkin testiraportti, kun yksi testitapaus on epäonnistunut

Raporttiedosto sisältää linkit yksityiskohtaisempaan lokitiedostoon, josta saadaan enemmän tietoa epäonnistumisen syystä. Kuvassa (11) on esitetty osa edellisen kuvan testiraportin lokitiedostosta.

[+] TEST CASE: New Bulletin Title Missing		00:00:46.566
[+] TEST CASE: New Bulletin Description Missing		00:00:47.964
Full Name: Bulletin.New Bulletin Should Fail.New Bulletin Description Missing		
Tags: fail		
Start / End / Elapsed: 20150210 22:03:49.765 / 20150210 22:04:37.729 / 00:00:47.964		
Status: <b>FAIL</b> (critical)		
Message: Location should have contained 'view' but it was 'http://localhost/bulletins/new'.		
[+] SETUP: resource_main.Open Browser To Login Page And Login		00:00:24.139
[+] KEYWORD: resource_bulletin.Go To New Bulletin Page		00:00:02.997
[+] KEYWORD: resource_main.Insert Item Basic Info \${TITLE}. \${EMPTY}		00:00:06.154
[+] KEYWORD: resource_main.Set Item Internal		00:00:04.129
[+] KEYWORD: resource_main.Submit Item		00:00:02.105
[+] KEYWORD: BuiltIn.Sleep 5s, Wait for Reply		00:00:05.000
[+] KEYWORD: Selenium2Library.Location Should Contain /view		00:00:02.337
Documentation: Verifies that current URL contains 'expected'.		
Start / End / Elapsed: 20150210 22:04:34.296 / 20150210 22:04:36.633 / 00:00:02.337		
[+] KEYWORD: Selenium2Library.Capture Page Screenshot		00:00:01.327
22:04:36.633 <b>FAIL</b> Location should have contained '/view' but it was 'http://localhost/bulletins/new'.		
[+] TEARDOWN: Selenium2Library.Close Browser		00:00:01.095
[+] TEST CASE: New Bulletin Sharing Options Missing		00:01:12.830
[+] TEST SUITE: New Bulletin Should Succeed		00:03:46.063

KUVA 11. Kuvankaappaus epäonnistuneen testiryhmän lokitiedostosta



## 4 TULOKSET

### 4.1 Valittu testaustyökalu

Opinnäytetyön tuloksena toimeksiantaja, Jakamo Osakeyhtiö, sai tärkeää tietoa tällä hetkellä markkinoilla olevista toiminnallisen testauksen automatisointityökaluista. Toimeksiantajan keskeisimpiä vaatimuksia olivat hankintakustannukset ja se, että työkalulla voisi mahdollisimman hyvin toteuttaa testit kaikille alustoille, joille Jakamo-sovellus on saatavilla. Muita valintakriteereitä olivat työkalun kirjastojen dokumentaation selkeys, eri selainten tukeminen, tukimateriaalin saatavuus, kehitysympäristön tehokkuus ja työkalun käytön oppimiskynnys.

Käytettävän testaustyökalun valinta suoritettiin kaksivaiheisen vertailun kautta. Ensimmäisessä vaiheessa työkaluja arvioitiin laadittujen valintakriteereiden perusteella, ja kunkin työkalun ominaisuudet pisteytettiin. Koska vaihtoehtoja oli alkuvaiheessa 11, ei jokaiseen olisi ollut mahdollisuutta tutustua kovin perusteellisesti. Tästä syystä työkaluja arvioitiin aluksi niistä saatavan tiedon perusteella. Ensimmäisen vertailuvaiheen tulokset kerättiin taulukkoon ja esiteltiin toimeksiantajalle, jonka kanssa vertailun toiseen vaiheeseen valittiin kaksi parasta vaihtoehtoa: Ranorex ja Robot Framework.

Toisessa vaiheessa lisättiin myös pintaa syvemmälle meneviä valintakriteereitä, minkä jälkeen työkalut asennettiin ja niillä toteutettiin pari yksinkertaista testiä, jotta saataisiin parempi kuva niiden käyttämisestä, kehitysympäristöstä ja dokumentaatiosta. Jälleen tulokset kerättiin taulukkoon ja esiteltiin toimeksiantajalle. Parhaiten Jakamon tarpeita vastaavaksi testaustyökaluksi valittiin Robot Framework.

### 4.2 Esimerkkitestit

Opinnäytetyön tarkoituksena oli myös tuottaa esimerkkitestejä valitulla testaustyökalulla, Robot Frameworkilla. Työssä ei lähdetty toteuttamaan koko järjestelmän kattavaa testausarkkitehtuuria, vaan ainoastaan muutaman testin toteuttaminen riitti. Tuloksena syntyi raportti Robot Frameworkin keskeisimmistä ominaisuuksista sekä yhteensä kuusi kappaletta valmiita testitapauksia. Samalla esiteltiin Jakamon ketterän kehityksen peri-

aatteita mukailevaan tuotekehitysprosessiin mielestäni hyvin sopiva käyttötapauksiin perustuva testien suunnitteluprosessi.

## 5 POHDINTA

Ennen opinnäytetyön aloittamista, minulla ei ollut yhtään kokemusta automaattisten testien toteuttamisesta, eikä ohjelmistotestauksesta muutenkaan juuri lainkaan. Työtä tehdessäni opin paljon erilaisista testaustyökaluista ja sain käytännön kokemusta automaattisten testien toteuttamisesta. Testejä suunnitellessani perehdyin myös käyttötapauksiin perustuvaan testisuunnitteluprosessiin.

Vaikka opinnäytetyön aihe ei ollutkaan minulla ennestään lainkaan tuttu, en koe että olisin työtä tehdessäni kohdannut paljon teknisiä haasteita. Työn suurimpana haasteena oli tekemisen aikatauluttaminen opiskelun, päivätyön ja vapaa-ajan lomassa, ja sen valmistuminen viivästyikin jonkun verran alkuperäisestä suunnitelmasta. Toimeksiantaja oli kuitenkin hyvin ymmärtäväinen, eikä asettanut kovia paineita työn valmistumisaikataululle. Luulen, että opinnäytetyö olisi valmistunut hyvinkin nopeasti, mikäli se olisi kuulunut niin sanottuihin normaaleihin työtehtäviini toimeksiantajayrityksessä.

Opinnäytetyön tarkoituksena oli etsiä toimeksiantajan tarpeisiin sopiva toiminnallisten testien automatisointityökalu ja toteuttaa sillä muutamia esimerkkitestejä. Työn ensimmäisessä vaiheessa, työkalujen vertailussa, lähteinä toimivat testaustyökalujen omat internetsivut. Koska sivut oli poikkeuksetta toteutettu tuotteen markkinointiin, tuli niitä lukea asianmukaisella lähdekriittisyydellä; joissakin tapauksissa oli hyvin vaikeaa saada todellista kuvaa siitä, mitä kyseessä olevalla testaustyökalulla oikeasti pystyy tekemään ja mille alustalle. Kuitenkin riittävän tarkasti lähteitä tutkimalla, löydettiin tarpeeksi tietoa työkalujen luotettavaa vertailua varten.

Opinnäytetyössä esitettiin teoria käyttötapauksiin perustuvasta testien suunnitteluprosessista, jota soveltamalla esimerkkitestit suunniteltiin. Teorian tarkoituksena oli esitellä toimintatapa, jolla testit voidaan tarvittaessa suunnitella todella kattavasti. Työssä päädyttiin käyttämään käyttötapauslähtöistä testisuunnittelua, jotta suunnittelu olisi mahdollisimman helposti omaksuttavissa myös niille, joilla ei välttämättä ohjelmointiosaamista. Esitetty testien suunnittelutapa tullaan ottamaan osaksi toimeksiantajan tuotekehitysprosessia.

Opinnäytetyön kuuden esimerkkitestin toteuttaminen osoitti, että testien automatisointi Robot Frameworkia käyttämällä oli helposti omaksuttavissa. Testikehyksen syntaksi ja

toimintaperiaate oli helppo käsittää jo pelkästään sen mukana tulleen esimerkin avulla. Vertailuvaiheessa Robot Frameworkille annetut pisteet tuntuivat vastaavan todellisuutta myös hiukan syvällisemmän tutustumisen jälkeen.

Toimeksiantajaa kiinnosti myös heidän mobiilisovelluksiensa testaaminen Robot Frameworkilla. Opinnäytetyöhön näiden testien toteuttamista ei kuitenkaan voitu sisällyttää. Robotiin on saatavilla Appium-kirjasto, jolla voidaan toteuttaa testejä Android- ja iOS-alustoille. Appium-kirjasto sisältää avainsanat kyseisten mobiilialustojen testaamista varten.

Tulevaisuudessa aloitettavaa testien toteutusta varten Jakamon käyttöliittymät olisi syytä käydä läpi, sillä johdonmukaisella käyttöliittymäelementtien nimeämisellä testien kirjoittamisesta saadaan huomattavasti sujuvampaa. Tällä tavoin myös testien uudelleenkäytettävyyttä voidaan parantaa merkittävästi.

Opinnäytetyön tulokset tullaan käymään läpi toimeksiantajan tuotekehitystiimin kanssa, minkä jälkeen niiden suunnittelu ja toteutus lisätään tehtävälistalle. Testien toteutus onärkevintä aloittaa Jakamon liiketoiminnan kannalta keskeisimmistä toiminnoista ja palalalta täydentää koko järjestelmän laajuiseksi. Suunnitteluun on kuitenkin syytä käyttää aikaa, sillä testien automatisoinnista on taloudellista hyötyä vain, jos niiden ylläpito-kustannukset ovat lähes olemattomat (Black 2007, 42).

## LÄHTEET

Singh, Y. 2012. Software Testing. Cambridge University Press.

Tian, J. 2005. Software Quality Engineering. John Wiley & Sons.

Black, R. 2002. Managing the Testing Process. Wiley Publishing.

Myers, G., Badgett, T., Sandler, C. 2012. The Art of Software Testing, 3rd Edition. John Wiley & Sons.

Tamres, L. 2002. Introducing Software Testing. Pearson Education.

Black, R. 2007. Pragmatic Software Testing. Wiley Publishing.

Selenium. 2015. SeleniumHQ. [Viitattu 26.2.2015] Saatavissa: <http://www.seleniumhq.org/>

Appium. 2015a. Mobile App Automation Made Awesome. [Viitattu 26.2.2015] Saatavissa: <http://appium.io/>

Appium. 2015b. The History of Appium. [Viitattu 26.2.2015] Saatavissa: <http://appium.io/history.html>

Sahi. 2015. Licensing and pricing. [Viitattu 26.2.2015] Saatavissa: <http://sahipro.com/pricing-licensing/>

Telerik. 2015. Purchase Telerik Software Development Tools. [Viitattu 26.2.2015] Saatavissa: <http://www.telerik.com/purchase.aspx>

Alexa. 2015. Stackoverflow.com Site Overview. [Viitattu 26.2.2015] Saatavissa: <http://www.alexa.com/siteinfo/stackoverflow.com>

Robot Framework. 2014a. Introduction. [Viitattu 18.12.2014] Saatavissa: <http://robotframework.org/#introduction>

Robot Framework. 2014b. Test libraries. [Viitattu 18.12.2014] Saatavissa: <http://robotframework.org/#test-libraries>

Robot Framework. 2014c. Tools. [Viitattu 18.12.2014] Saatavissa: <http://robotframework.org/#tools>

Python. 2014a. JythonFaq. [Viitattu 18.12.2014] Saatavissa: <https://wiki.python.org/jython/JythonFaq/GeneralInfo>

IronPython. 2014. IronPython. [Viitattu 18.12.2014] Saatavissa: <http://ironpython.net/>

Python. 2014b. Python2orPython3. [Viitattu 18.12.2014] Saatavissa: <https://wiki.python.org/moin/Python2orPython3>

PyPi. 2014. Python Package Index. [Viitattu 18.12.2014] Saatavissa: <https://pypi.python.org/pypi>

Python Software Foundation. 2014. Robot Framework 2.8.6. [Viitattu 18.12.2014] Saatavissa: <https://pypi.python.org/pypi/robotframework>

Haikala, I & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. Talentum.

Robot Framework. 2014d. User Guide. [Viitattu 30.1.2015] Saatavissa: <http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html>

LIITTEET

LIITE 1. ”Luo uusi tiedote”-käyttötapauksen testitapaustaulukko

1(2)

Luo uusi tiedote

ID	Skenaarion nimi ja kuvaus	Title *	Description *	**			HideSharing	Tags[]
				IsInternal	Relationship[]	IsPublic		
TT1.1	Skenaario 1 - Luo uusi tiedote	Tammikuun asiakastapaamiset	Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015, Asiakas 2 Oy - 15.1.2015, Asiakas 3 Oy - 16.1.2015, Asiakas 4 Oy	TRUE				
TT1.2	Skenaario 1 - Luo uusi tiedote	Tammikuun asiakastapaamiset	Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015, Asiakas 2 Oy - 15.1.2015, Asiakas 3 Oy - 16.1.2015, Asiakas 4 Oy		1, 2, 3, 4			
TT1.3	Skenaario 1 - Luo uusi tiedote	Tammikuun asiakastapaamiset	Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015, Asiakas 2 Oy - 15.1.2015, Asiakas 3 Oy - 16.1.2015, Asiakas 4 Oy			TRUE		
TT2.1	Skenaario 2 – Luo uusi tiedote, poikkeus: pakollisia tietoja puuttuu		Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015, Asiakas 2 Oy - 15.1.2015, Asiakas 3 Oy - 16.1.2015, Asiakas 4 Oy	TRUE				
TT2.2	Skenaario 2 – Luo uusi tiedote, poikkeus: pakollisia tietoja puuttuu	Tammikuun asiakastapaamiset		TRUE				
TT2.3	Skenaario 2 – Luo uusi tiedote, poikkeus: pakollisia tietoja puuttuu	Tammikuun asiakastapaamiset	Asiakastapaamiset: - 1.1.2015, Asiakas 1 Oy - 8.1.2015, Asiakas 2 Oy - 15.1.2015, Asiakas 3 Oy - 16.1.2015, Asiakas 4 Oy					

\* = pakollinen muuttuja, \*\* = yhdellä sarakkeeseen muuttujista oltava arvo

(jatkuu)

2(2)

..		..		Attachment[]	AllowAdd	AllowRemove	Lopputuloks	Muita
PublishNow	PublicationTime	NeverExpire	ExpireTime					
TRUE		TRUE					Uusi tiedote luodaan onnistuneesti	
TRUE		TRUE					Uusi tiedote luodaan onnistuneesti	
TRUE		TRUE					Uusi tiedote luodaan onnistuneesti	
TRUE		TRUE					Tiedotteen luominen ei onnistu	Pakollisella title-muuttujalla ei ole arvoa
TRUE		TRUE					Tiedotteen luominen ei onnistu	Pakollisella description-muuttujalla ei ole arvoa
TRUE		TRUE					Tiedotteen luominen ei onnistu	Jollakin isInternal-, relationship- ja isPublic-muuttujista täytyy olla arvo



## LIITE 2. Robot Framework-esimerkkitestien koodit

1(6)

resource\_main.txt

\*\*\* Settings \*\*\*

Documentation      Contains generally usable keywords and variables  
 Library            Selenium2Library

\*\*\* Variables \*\*\*

\${SERVER}            localhost  
 \${BROWSER}           Firefox  
 \${DELAY}             1  
 \${VALID EMAIL}       testi@example.com  
 \${VALID PASSWORD}    123456  
 \${FRONT PAGE URL}    http://\${SERVER}  
 \${LOGIN URL}          http://\${SERVER}/login  
 \${NEW BULLETIN URL} http://\${SERVER}/bulletins/new

\*\*\* Keywords \*\*\*

Should Be Logged In

Go To      \${FRONT PAGE URL}  
             Page Should Not Contain      id=sign\_in\_form

Go To Login Page And Login

Go To      \${LOGIN URL}  
             Set Selenium Speed      \${DELAY}  
 Login Page Should Be Open  
             Insert Credentials      \${VALID EMAIL}      \${VALID PASSWORD}  
             Submit Credentials  
             Should Be Logged In

Open Browser To Login Page And Login

Open Browser      \${LOGIN URL}      \${BROWSER}  
 Maximize Browser Window  
 Set Selenium Speed      \${DELAY}  
 Login Page Should Be Open  
             Insert Credentials      \${VALID EMAIL}      \${VALID PASSWORD}  
             Submit Credentials  
             Should Be Logged In

Login Page Should Be Open

Location Should Contain      /login

Insert Credentials

[Arguments]      \${EMAIL}      \${PASSWORD}  
 Input Text      username      \${EMAIL}  
                  Input Text      password      \${PASSWORD}

Submit Credentials

Click Element      sign\_in\_button

Insert Item Basic Info

[Arguments]      \${TITLE}      \${DESCRIPTION}  
                  Input Text      Title      \${TITLE}  
                  Input Text      Description      \${DESCRIPTION}

Set Item Internal

Select Checkbox      IsInternal

Set Item Public

Select Checkbox      IsPublic

(jatkuu)

```

Set Item Relationships
  [Arguments]      ${RELATIONSHIPS}
      Click Element    add_relation
  :FOR      ${ELEMENT}    IN      @${RELATIONSHIPS}
      \    ${id}=      Catenate      SEPARATOR=      relation_      ${ELE-
MENT}
      \    Click Element      ${id}

Submit Item
  Click Element      submitButton

```

2(6)

new\_bulletin\_should\_fail.txt

3(6)

## \*\*\* Settings \*\*\*

Documentation      A test suite for testing that a new bulletin can not  
be created when it shouldn't  
Resource            ../resource\_main.txt  
Resource            res/resource\_bulletin.txt  
Default Tags        fail

## \*\*\* Test Cases \*\*\*

## New Bulletin Title Missing

[Setup]      Open Browser To Login Page And Login  
              Go To New Bulletin Page  
Insert Item Basic Info    \${EMPTY}      \${DESCRIPTION}  
              Set Item Internal  
              Submit Item  
              Sleep      5s      Wait for reply  
              Location Should Contain      /bulletins/new  
              Page Should Contain Element      css=div.errors  
              Page Should Contain      Title is a required field.  
[Teardown]      Close Browser

## New Bulletin Description Missing

[Setup]      Open Browser To Login Page And Login  
              Go To New Bulletin Page  
Insert Item Basic Info    \${TITLE}      \${EMPTY}  
              Set Item Internal  
              Submit Item  
              Sleep      5s      Wait for Reply  
              Location Should Contain      /view  
              Page Should Contain Element      css=div.errors  
              Page Should Contain      Description for the bulletin is re-  
quired.  
[Teardown]      Close Browser

## New Bulletin Sharing Options Missing

[Setup]      Open Browser To Login Page And Login  
              Go To New Bulletin Page  
Insert Item Basic Info    \${TITLE}      \${DESCRIPTION}  
              Submit Item  
              Sleep      5s      Wait for Reply  
              Location Should Contain      /bulletins/new  
              Page Should Contain Element      css=div.errors  
              Page Should Contain      Bulletin needs to be set internal,  
public or shared with relations.  
[Teardown]      Close Browser

new\_bulletin\_should\_succeed.txt

4(6)

\*\*\* Settings \*\*\*

Documentation      A test suite for testing that a new bulletin can be created as it should

Resource            ../resource\_main.txt

Resource            res/resource\_bulletin.txt

Default Tags        success

\*\*\* Test Cases \*\*\*

New Bulletin Internal

[Setup]      Open Browser To Login Page And Login  
              Go To New Bulletin Page

Insert Item Basic Info    \${TITLE}      \${DESCRIPTION}

Set Item Internal

Submit Item

Sleep      5s      Wait for reply

Location Should Contain    /view

[Teardown]      Close Browser

New Bulletin Shared

[Setup]      Open Browser To Login Page And Login  
              Go To New Bulletin Page

Insert Item Basic Info    \${TITLE}      \${DESCRIPTION}

Set Item Relationships    \${RELATIONSHIPS}

Submit Item

Sleep      5s      Wait for Reply

Location Should Contain    /view

[Teardown]      Close Browser

New Bulletin Public

[Setup]      Open Browser To Login Page And Login  
              Go To New Bulletin Page

Insert Item Basic Info    \${TITLE}      \${DESCRIPTION}

Set Item Public

Submit Item

Sleep      5s      Wait for Reply

Location Should Contain    /view

[Teardown]      Close Browser

resource\_bulletin.txt

5(6)

\*\*\* Settings \*\*\*

Documentation      Bulletin related keywords

Library            Selenium2Library

Variables                variables\_bulletin.py

\*\*\* Keywords \*\*\*

Go To New Bulletin Page

    Go To      \${NEW BULLETIN URL}

variables\_bulletin.py

6(6)

```
TITLE = "Tammikuun asiakastapaamiset"
DESCRIPTION = "Asiakastapaamiset: \n- 1.1.2015, Asiakas 1 Oy\n- 8.1.2015, Asiakas 2 Oy\n- 15.1.2015, Asiakas 3 Oy\n- 16.1. 2015, Asiakas 4 Oy"
LIST__RELATIONSHIPS = [1, 43, 57]
```