Tuukka Jurvakainen

Ohjelmistorobotiikka

Tietotekniikan Pro gradu -tutkielma

10. marraskuuta 2018

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Tuukka Jurvakainen

Yhteystiedot: tuiljurv@student.jyu.fi

Ohjaaja: Vesa Lappalainen, Timo Tiihonen

Työn nimi: Ohjelmistorobotiikka

Title in English: Robotic Process Automation

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 68+29

Tiivistelmä: Ohjelmistorobotiikalla tarkoitetaan sovelluksia, jotka käyttävät järjestelmiä ihmisen tavoin. Ohjelmistorobotiikalla voidaan automatisoida käsintehtävää työtä, jota ei kannata kustannusten vuoksi automatisoida perinteisillä automaation keinoilla. Tässä tutkielmassa käymme läpi tieteellisen kirjallisuuden pohjalta ohjelmistorobotiikan perusteita ja määrittelemme kriteereitä ongelmille, jotka ovat soveltuvia ohjelmistorobotiikalla ratkaistaviksi. Lisäksi kartoitamme ohjelmistorobotiikan toteutukseen sopivia työkaluja. Tutkielman käytännön osassa toteutamme ohjelmistorobotin neljälle eri työkalulla todelliseen ongelmaan. Tutkielman johtopäätökseksi saimme, että ohjelmistorobotiikalla on mahdollista automatisoida tietynlaisia prosesseja. Prosessin tulee kuitenkin täyttää hyvin tarkat kriteerit, jotta se kannattaa automatisoida ohjelmistorobotiikalla. Automaatiolle sopivien tapausten löytyminen on haaste ohjelmistorobotiikan hyödyntämiselle.

Avainsanat: Ohjelmistorobotiikka, automaatio, UiPath, RPA Express, Selenium, SikuliX

Abstract: Robotic Process Automation (RPA) refers to applications that use computer systems in the same way as humans do. With RPA it is possible to automate manual tasks more cost-efficiently than using traditional system integration. In this thesis we give an overview of the basics of RPA based on recent scientific literature. Based on this we summarize the criteria for tasks and use cases that are suitable for RPA. We also introduce a variety of available tools for RPA. In the practical part of the thesis we construct, using different tools, four

different RPA implementations to solve a real problem. The main conclusion of the work is that RPA is a viable tool for automating processes but not for any problem and not with every tool. Identification and definition of suitable use cases requires good understanding of the processes as well as of the available tools.

Keywords: Robotic Process Automation, automation, UiPath, RPA Express, Selenium, SikuliX

Termiluettelo

Kuvatunnistus: Objektin tai elementin tunnistaminen mallikuvan avulla.

Ohjelmistorobotiikka: Teknologia, jossa automatisoidaan rutiinityötä ohjelmistorobottien avulla.

Ohjelmistorobotti: Tietokoneohjelma, joka käyttää muita tietokoneohjelmia ihmisen tavoin.

Ohjelmointirajapinta: Säännöstö, joka mahdollistaa ohjelmien välisen tiedon vaihdon ja pyyntöjen tekemisen.

RPA Express: Kaupallinen ohjelmistorobotiikan työkalu.

Selenium: Avoin työkalu web-sivujen testiautomaation tekemiseen. Voidaan käyttää ohjelmistorobottien luomiseen web-ympäristössä.

SikuliX: Avoin ohjelmistorobotiikan työkalu.

Tekstintunnistus: Teknologia, jonka avulla pystytään tunnistamaan tekstiä kuvasta ja kääntämään se sähköisesti muokattavaan muotoon.

UiPath: Kaupallinen ohjelmistorobotiikan työkalu.

Kuviot

Taulukot

Sisältö

1 Johdanto

Ohjelmistorobotiikalla (engl. robotic process automation, RPA) tarkoitetaan esimääriteltyjä ohjelmia, jotka suorittavat tiettyjen sääntöjen mukaan prosesseja, toimintoja ja tehtäviä yhdessä tai useammassa erillisessä järjestelmässä (IEEE 2017). Ohjelmistorobotteja voidaan käyttää automatisoimaan käsintehtävää työtä. Ohjelmistorobotiikka eroaa perinteisestä automaatiosta siinä, että ohjelmistorobotiikassa automatisoitaviin järjestelmiin ei usein tarvitse tehdä muutoksia (Asatiani ja Penttinen 2016). Sen sijaan ohjelmistorobotti on erillinen ohjelma, joka automatisoi työn tekemällä sen ihmisen tavoin. Tämä tekee ohjelmistorobotiikasta perinteistä automaatiota halvemman sekä nopeamman toteuttaa, sillä valmiiden järjestelmien lähdekoodin muokkaaminen on usein hidasta ja kallista (Wil M P van der, Bichler ja Heinzl 2018).

Tällä hetkellä puhutaan paljon automaation mahdollisuuksista, sillä laittamalla kone tekemään työtehtäviä ihmisen puolesta pystytään laskemaan kustannuksia (Fung 2014). Ohjelmistorobotiikalla kyetään tehostamaan työskentelyä ja vähentämään henkilötyövuosia, sillä ohjelmistorobotit pystyvät työskentelemään myös ihmistyöntekijän työajan ulkopuolella (Fung 2014). Ohjelmistoroboteilla on mahdollisuus säästää kustannuksia myös niiden tarkkuuden avulla (Alégroth, Feldt ja Ryrholm 2015) (Fung 2014). Ihmisille sattuu monotonisia töitä tehdessä virheitä, joita voidaan välttää antamalla työtehtävät robotin tehtäväksi. Esimerkiksi sopii tilanne, jossa ihminen kopioi huolimattomasti tietoa järjestelmästä toiseen. Hyvin tehty ohjelmistorobotti kykenee virhetarkistuksiin tai lopettamaan työtehtävän, kun jotain odottamatonta tapahtuu.

Ohjelmistorobotiikka on erittäin ajankohtainen aihe. Esimerkiksi IEEE:n standardi aiheesta käytettäville termeille julkaistiin syksyllä 2017 (IEEE 2017). On tärkeä kartoittaa millaisiin työtehtäviin ohjelmistorobotit soveltuvat sekä kuinka niitä tulisi toteuttaa.

Tässä Pro Gradu -tutkielmassa toteutetaan konstruktiivinen tutkimus. Konstruktiivinen tutkimus tarkoittaa tutkimusta, jossa toteutetaan käytännön ratkaisu todelliseen ongelmaan aiemman teoriatiedon pohjalta (Järvinen 2004). Tutkielmassa tehdään kirjallisuuskatsaus aiheesta kirjoitettuihin tieteellisiin julkaisuihin ja kartoitetaan ohjelmistorobotiikkaan sopivia

työkaluja. Tutkielman empiirisessä osassa löydettyä tietoa pyritään soveltamaan käytännössä rakentamalla ohjelmistorobotti ratkaisemaan todellinen ongelma. Tutkielman tarkoituksena on yhdistää käytäntö ja teoria toisiinsa, sekä esittää ohjelmistorobottien toteutukseen hyviä käytännön ratkaisumalleja.

Ohjelmistorobotiikalla ratkaistavaksi ongelmaksi valikoitui palkkion maksamiseen tarkoitetun lomakkeen täyttäminen. Jyväskylän yliopiston hallinto ja IT-palvelut tarjosivat tapausta, koska se oli todettu sellaiseksi, että siinä voitaisiin ohjelmistorobotiikan avulla saavuttaa merkittäviä kustannussäästöjä.

Tutkimus toteutettiin niin, että aluksi tutustuttiin työnkulkuun, jonka palkkiolaskulomakkeen käsittelijä joutuu tekemään. Tämän pohjalta hahmoteltiin ne vaiheet ja askeleet, jotka ohjelmistorobotti joutuu työssään tekemään. Itse ohjelmistorobotti toteutettiin sekä avoimilla että kaupallisilla robotiikkatyökaluilla. Ohjelmistorobotin onnistuneisuutta arvioitiin tutkimuksen lopuksi. Arviointiin kuuluivat seuraavat näkökohdat, jotka nousivat esiin teoriataustasta:

- tapauksen sopivuus ohjelmistorobotiikalla ratkaistavaksi (luku 3.1)
- nopeus verrattuna käsityöhön
- tarkkuus verrattuna käsityöhön
- ohjelmistorobotin toteutuksen kesto (arvio)
- toteutuksen tietoturva
- ohjelmistorobotin vakaus häiriötilanteissa

Luvussa 2 kaksi käydään läpi tieteellisestä kirjallisuudesta nousevaa teoriataustaa ohjelmistorobotiikasta. Luvussa käsitellään muun muassa ohjelmistorobotiikan ominaispiirteitä, etuja ja haasteita verrattuna muihin ratkaisuihin. Luvussa 3 määritellään kriteerit ongelmille, joita ohjelmistorobotiikalla voidaan tehokkaasti ratkaista. Luvussa 4 käydään läpi ohjelmistorobotiikan työkaluja. Luvussa 5 esitellään tarkemmin neljä eri työkalua, joita käytetään tutkielman käytännön osassa. Luvussa 6 esitetään tutkimuksen toteutus, josta saatuja tuloksia arvioidaan luvussa 7. Lopuksi tutkielman johtopäätökset vedetään yhteen luvussa 8.

2 Teoriataustaa ohjelmistorobotiikasta

Ohjelmistorobotiikalla (engl. robotic process automation, RPA) tarkoitetaan esimääriteltyjä ohjelmia, jotka suorittavat tiettyjen sääntöjen mukaan prosesseja, toimintoja ja tehtäviä yhdessä tai useammassa erillisessä järjestelmässä (IEEE 2017). Ohjelmistorobotit käyttävät järjestelmiä ihmisen tavoin eli usein graafisen käyttöliittymän välityksellä (Asatiani ja Penttinen 2016). Tässä tutkielmassa käsitämme ohjelmistorobotit ohjelmina, jotka käyttävät tietokonetta kuten ihminen.

Tässä luvussa käymme läpi ohjelmistorobotiikan perusteita tieteellisen kirjallisuuden pohjalta. Käymme läpi ohjelmistorobottien toimintaa sekä hyötyjä ja haasteita, joita ohjelmistorobottien tekemiseen liittyy.

2.1 Ohjelmistorobotiikan ominaispiirteitä

Ohjelmistorobotiikan ero perinteiseen automatisointiin verrattuna on, että siinä ohjelmistorobotti pyrkii käyttämään olemassa olevia järjestelmiä kuten ihminen (Asatiani ja Penttinen 2016). Perinteisessä automaatiossa pitää usein muuttaa olemassa olevia järjestelmiä yhteensopiviksi automaatiota varten. Perinteisen automaation toteuttamiseksi tarvitaan tietoa järjestelmän lähdekoodista, tietorakenteista tai ohjelmointirajapinnoista (engl. Application programming interface, API) (Penttinen, Kasslin ja Asatiani 2018). Esimerkiksi kahden järjestelmän yhteensovittaminen perinteisin menetelmin voi tapahtua tekemällä yhteensopiva ohjelmointirajapinta näiden välille. Rajapinnan avulla tiedonsiirto järjestelmien välillä voidaan automatisoida, mutta rajapinnan toteutus vaatii pääsyn järjestelmien lähdekoodiin, jollei yhteensopivia rajapintoja valmiiksi löydy. Ohjelmistorobotiikalla toteutetussa automaatiossa ei välttämättä tarvita mitään tietoa järjestelmien käyttäjälle näkymättömistä rakenteista. Itse työn suoritustapaa ei yritetä muokata, esimerkiksi lisäämällä järjestelmään ohjelmointirajapintoja, vaan ohjelmistorobotti suorittaa työn samaan tapaan kuin ihminen käyttämällä olemassa olevia käyttäjärajapintoja (Asatiani ja Penttinen 2016). Tästä seuraa, että ohjelmistorobotit voidaan toteuttaa omina erillisinä osinaan ja omilla työkaluilla. Automatisoitavien järjestelmien lähdekoodia ei tarvitse yleensä muokata.

Toinen oleellinen piirre ohjelmistoroboteissa on, että niiden rakentamiseen käytetään usein graafisia työkaluja, jolloin ohjelmointitaito ei ole välttämätöntä. Tämä on mahdollista siksi, että ohjelmistorobottien rakentamisessa tarvitaan usein tietoa vain käyttäjärajapinnasta, eikä järjestelmien lähdekoodin tai ohjelmointirajapintojen tunteminen ole välttämätöntä. Tämä mahdollistaa sen, että ohjelmistorobottien tekijöiden ei välttämättä tarvitse olla IT-ammattilaisia. Automatisointi on mahdollista helppokäyttöisillä ohjelmistorobotiikan työkaluilla. (Penttinen, Kasslin ja Asatiani 2018)

Tietotekniikan alalla on ollut käytössä jo vuosikymmeniä makrot ja skriptaus, joilla voidaan automatisoida säännönmukaisesti toistuva rutiinitoimenpide. Ohjelmistorobottien voidaan nähdä olevan makroista ja skriptauksesta kehittynyt teknologia. Ohjelmistorobotit tarjoavat kuitenkin mahdollisuuden paljon monimutkaisempien prosessien automatisointiin kuten järjestelmäintegraatioihin. (Penttinen, Kasslin ja Asatiani 2018)

Taulukko 1: Ohjelmistorobotiikan ominaispiirteet perinteiseen automaatioon verrattuna.

Ominaisuus	Perinteinen automaatio	Ohjelmistorobotiikka
Vaatii lähdekooditason tuntemusta järjestelmistä	Kyllä	Ei
Vaatii järjestelmien muokkaamista	Useimmiten	Ei
Vaatii ohjelmointitaitoa	Kyllä	Ei välttämättä

2.2 Ohjelmistorobotiikan etuja

Ohjelmistorobotit kommunikoivat toisten ohjelmien kanssa ihmismäisesti eli ne käyttävät ihmisen tavoin näiden ohjelmien käyttöliittymää. Asatianin tutkimusryhmän mukaan tällaisessa lähestymistavassa on monia etuja. Ensinnäkin ohjelmistorobotit pystyvät tämän takia käyttämään mitä tahansa järjestelmää, jota ihminenkin käyttää. Tämän vuoksi ohjelmistoroboteilla voidaan usein kiertää rajapinnoiltaan sopimattomien järjestelmien yhteensopivuusongelma ilman, että järjestelmien lähdekoodiin tarvitsee olla pääsyä. Esimerkiksi jos

yhden järjestelmän täytyy saada tietoa toisesta, mutta tieto ei välity suoraan järjestelmien välillä, voidaan tähän väliin toteuttaa ohjelmistorobotti, joka siirtää tietoa järjestelmästä toiseen. (Asatiani ja Penttinen 2016)

Toiseksi eduksi Asatianin tutkimusryhmä mainitsee ohjelmistorobottien nopean toteutuksen. Järjestelmien yhteensopivuusongelmien korjaaminen voi viedä kuukausista vuosiin, jotta toimiva rajapinta saadaan toteutettua. Asatanian tutkijaryhmän mukaan ohjelmistorobotin toteutus, jolla rajapintaongelma voidaan kiertää, kestää usein vain 2-4 viikkoa. (Asatiani ja Penttinen 2016)

Kolmanneksi Asatanian tutkimusryhmä mainitsee ohjelmistorobottien helpon muokattavuuden. Tutkijaryhmän mukaan jopa järjestelmän käyttäjät pystyvät mahdollisesti halutessaan tekemään muokkauksia robottien toimintaan. Perinteisien ohjelmien muokkaus vaatii usein kehittynyttä ohjelmointitaitoa, jos sen toimintaa halutaan muokata. RPA-ohjelmistot eivät usein vaadi edes ohjelmointitaitoa (Luku Ohjelmistorobotiikan työkalut), sillä niissä voi olla vain graafinen kaavio prosessista, jonka robotti suorittaa. Tätä kaaviota muokkaamalla käyttäjä itse voi muokata ohjelmistorobotin toimintaa helposti. (Asatiani ja Penttinen 2016) (Lacity ja Willcocks 2016)

Ohjelmistorobotit suorittavat myös työnsä ihmistä tarkemmin. Ihmisille sattuu töissään inhimillisiä virheitä, kuten esimerkiksi ylimääräiset näppäimen painallukset tai vastaavat tietoa syötettäessä. Ohjelmistorobotti suorittaa työtehtävänsä rutiininomaisesti ja ennalta määrättynä, joten se ei tee inhimillisiä virheitä. (Alégroth, Feldt ja Ryrholm 2015) (Fung 2014)

Ohjelmistorobotiikka tulee halvemmaksi kuin perinteinen automaatio. Ohjelmistorobotit eivät edellytä muutoksia automatisoitavissa järjestelmissä. Perinteisessä automaatiossa joudutaan tekemään usein muutoksia järjestelmien lähdekoodiin, mikä on hidasta ja kallista. Ohjelmistorobotiikka on kevyemmän tason automaatiota, joka mahdollistaa nopean käyttöönoton. (Wil M P van der, Bichler ja Heinzl 2018)

2.3 Ohjelmistorobotiikan haasteita

Ohjelmistoroboteilla on omat rajoituksensa. Ohjelmistorobottien toiminta perustuu sääntöihin, joten ne eivät mukaudu ympäristön muutoksiin itsestään tai osaa tehdä päätöksiä. Ne ovat tehokkaimpia yksinkertaisissa rutiinitehtävissä, jossa työmäärä on suuri. (Penttinen, Kasslin ja Asatiani 2018)

Vaikka ohjelmistorobotiikalla voidaan integroida järjestelmiä keskenään helposti, niin ohjelmistorobotit häviävät nopeudessa koodin puolella toteutetulle integraatiolle, jossa järjestelmät toimivat keskenään ilman välikäsiä. Ohjelmistorobotit ovat usein väliaikaisia ratkaisuja käsityön ja täysin yhteensopivien järjestelmien välillä. Todella suurilla toistuvien rutiinitoimenpiteiden määrällä perinteinen automaatio tulee ohjelmistorobotiikkaa kannattavammaksi. (Asatiani ja Penttinen 2016) (Wil M P van der, Bichler ja Heinzl 2018)

Ohjelmistorobotit ja automaatio mahdollisesti vähentävät työpaikkoja (Herbert 2016) (Chelliah 2017). Frey ym. (2016) esittävät artikkelissaan, että nykyisistä töistä 47 % automatisoituu Yhdysvalloissa seuraavan 20 vuoden aikana (Frey ja Osborne 2017). Tämän vuoksi työntekijät voivat nähdä ohjelmistorobotit uhkana omalle työlleen (Lacity ja Willcocks 2016). Lacity ym. (2016) toteavat tapaustutkimuksessaan tämän olleen työntekijöiden pelkona, kun ohjelmistorobotiikkaa otettiin käyttöön tutkittavassa olleessa yrityksessä. Pelot kuitenkin väistyivät, kun työntekijöiden työnkuva muuttuikin vain rutiinitehtävistä mielenkiintoisemmaksi (Lacity ja Willcocks 2016). Uhkakuvat työpaikkojen vähenemisestä ovat kuitenkin todellisia ja toimistotyöläisten tulee varautua työnkuvansa muuttumiseen automaation myötä (Chelliah 2017) (Frey ja Osborne 2017).

Penttinen ym. huomauttavat, että tietoturva on yksi haaste ohjelmistorobotteja käytettäessä. Erityishuomiota vaatii, mitä tietoja ohjelmistorobotille välitetään ja kuinka se niitä käsittelee. Ohjelmistorobotit esimerkiksi saattavat joutua kirjautumaan järjestelmiin, jolloin salasanojen suojaus vaatii oman huomionsa. (Penttinen, Kasslin ja Asatiani 2018)

2.4 Kuinka robotti käyttää graafista käyttöliittymää

Ohjelmistorobottien toteutuksen haaste liittyy siihen, kuinka ne pystyvät käyttämään käyttäjärajapintaa, joka on usein graafinen käyttöliittymä (Asatiani ja Penttinen 2016). Jos ohjelmistorobotin halutaan esimerkiksi hakevan lomakkeen tietystä kentästä tietty arvo, niin ohjelmistorobotin täytyy kyetä tunnistamaan graafisesta käyttöliittymästä oikea lomakkeen kenttä. Tämä voi tapahtua usealla eri tavalla.

Yksi vaihtoehto on, että ohjelmistorobotille opetetaan mallikuvan avulla, miltä etsittävä kenttä tai muu graafinen elementti näyttää. Ohjelmistorobotti etsii sitten tämän mallikuvan avulla oikean elementin. Tätä kutsutaan kuvatunnistukseksi (engl. image recognition) (Alégroth, Feldt ja Ryrholm 2015). Tällä tavalla toteutettu robotti pystyy periaatteessa toimimaan missä vain graafisessa järjestelmässä. Haittapuolena on, että jos mallikuva ei täysin vastaa etsittyä elementtiä, robotti ei toimi. Näin voi käydä esimerkiksi silloin, jos robotti kehitetään toisessa järjestelmässä, mutta sitä käytetään toisessa, jossa esimerkiksi käyttöliittymä elementit ovat erinäköisiä (vrt. Windows 10 ja Windows 7 painikkeet). Toinen ongelma ovat useat samanlaiset elementit. Robotin täytyy käyttää tällöin niin sanottuja ankkurikohteita, joiden avulla se etsii oikean elementin usean samanlaisen joukosta. Ankkurikohde on näytöllä sijaitseva uniikki kohde, joka voi sijaita esimerkiksi halutun elementin vieressä. Esimerkiksi lomakkeen kenttä voi olla haluttu elementti ja sen vieressä oleva selitysteksti ankkurikohde, jonka avulla oikea tekstikenttä löydetään usean samanlaisen tekstikentän joukosta.

Web-pohjaisiin järjestelmiin on usein helpompi toteuttaa ohjelmistorobotteja, sillä verkkosivun rakenne on saatavana tekstimuotoisena html-koodina. Kun ohjelmistorobotti käyttää web-pohjaista järjestelmää, sen ei välttämättä tarvitse tulkita näytön graafisia elementtejä, vaan elementtien löytäminen tapahtuu tutkimalla web-sivun html-koodia. Rakenteeseen perustuva etsintä on myös nopeampaa, kuin kuvatunnistukseen perustuva elementtien etsintä.

Joissakin järjestelmissä robotin on mahdollista myös saada käyttöliittymäelementtien “kahvat” selville, jolloin robotti voi käyttää käyttöliittymäelementtejä ilman, että sen täytyy kuvasta tunnistaa erikseen niitä. Tämä tekee robotin toiminnasta varmempaa ja nopeampaa. Esimerkiksi luvussa 4.2 esitellyn UiPathin avulla voidaan rakentaa ohjelmistorobotteja, jotka kykenevät tunnistamaan Windows-järjestelmissä erilaisia elementtejä ilman kuvatunnis-

tusta.

Ohjelmistorobotit voivat perustua mallikäyttäjän toimintojen nauhoitukseen ja toistamiseen. Tällöin ohjelmistorobotti toistaa vain useaan kertaan samaa mallikäyttäjän toimintoa. Alegrothin tutkijaryhmä mainitsee tällaista tekniikka käytettävän erimerkiksi käyttöliittymien testausautomaatiossa. Tutkijaryhmä ei kuitenkaan pidä tällaista toteutusta erityisen hyvänä, sillä se on herkkä käyttöliittymän ja koodin muutoksille, esimerkiksi kun käyttöliittymän elementit vaihtavat paikkaa, niin samaa nauhoitusta ei voida enää käyttää ohjelmistorobotin pohjana. Tällä tavalla toteutettu ohjelmistorobotti toimii siis vain tietyssä järjestelmässä hyvin rajoittuneesti. (Alégroth, Feldt ja Ryrholm 2015)

Nauhoitusta voidaan kuitenkin hyödyntää ohjelmistorobotin tekovaiheessa. Nykyajan ohjelmistorobotiikkatyökalut mahdollistavat käyttäjän toiminnan nauhoittamisen tavalla, jossa nauhoituksen aikana ohjelmistorobotiikkatyökalu tunnistaa eri elementtejä ja luo nauhoituksen pohjalta valmiiksi säännöt, joita ohjelmistorobotti noudattaa. Näistä säännöistä saadaan pohja, jonka avulla voidaan alkaa toteuttaa varsinaista robottia.

3 Ohjelmistorobotiikalla ratkaistavia ongelmia

Ohjelmistorobotiikka soveltuu tavallisen automaation tapaan erityisesti samalla kaavalla toistuviin työtehtäviin (Asatiani ja Penttinen 2016). Tässä luvussa käymme läpi täsmällisemmin, minkä tyyppisiä ongelmia ohjelmistorobotiikalla voidaan ratkaista.

3.1 Kriteerit automatisoitavalle prosessille

Fung esittää artikkelissaan yhdeksän kriteeriä, joiden perusteella voidaan arvioida, kannattaako työtehtävää automatisoida ohjelmistorobotiikalla (Fung 2014). Seuraavassa esitellään Fungin löytämät arviointikriteerit.

3.1.1 Prosessi on helppo jakaa selkeisiin samana toistuviin osiin

Prosessin pitää olla jaettavissa selkeisiin osiin, jotka noudattavat tiettyjä sääntöjä (Fung 2014). Tällöin se on mahdollista automatisoida ohjelmistorobotiikalla. Ohjelmistorobotti noudattaa tiettyjä sääntöjä tarkasti, joten jos tehtävän askelissa on tulkinnan varaa, niin se ei ole automatisoitavissa (Asatiani ja Penttinen 2016).

Asatianin ym. (2016) mukaan keskeinen sääntö automatisointia miettiessä on se, että pystytäänkö kaikki prosessin askeleet kirjaamaan ylös ja ottamaan huomioon kaikki mahdolliset tilanteet ja poikkeukset. Jos pystytään, niin prosessi on automatisoitavissa. (Asatiani ja Penttinen 2016)

3.1.2 Prosessin tapahtumien määrä

Prosessin tapahtumien määrä on keskeinen mietittäessä automaation kannattavuutta. Prosessin automatisointia mietittäessä on hyvä analysoida, kuinka monta kertaa samanlaisena toistuva tapahtumaa joudutaan suorittamaan. Mitä suurempi on samanlaisena toistuvien tapahtumien määrä, sitä kannattavampaa on prosessien automatisointi. (Asatiani ja Penttinen 2016) (Fung 2014)

Penttinen ym. (2018) esittävät, että erittäin suuri määrä tapahtumia puoltaa erityisesti perinteistä, järjestelmien muokkaamista vaativaa automaatiota. Tapahtumien määrä prosessissa on silloin niin suuri, että järjestelmän muokkaaminen kannattaa. Ohjelmistorobotiikkaa puoltaa taas suurehko tapahtumien määrä, jolloin automatisointi kannattaa käsityöhön verrattuna, mutta järjestelmää itsessään ei vielä välttämättä kannata lähteä muokkaamaan. (Penttinen, Kasslin ja Asatiani 2018) (Wil M P van der, Bichler ja Heinzl 2018)

3.1.3 Prosessin tapahtumien kustannus

Myös prosessin tapahtumien kustannusta pitää arvioida. Vaikka tapahtumien määrä olisi pieni, mutta kustannus manuaalityönä iso, niin prosessin automatisointia tulee harkita. Fung (2014) antaa artikkelissaan esimerkiksi rutiinitoimenpiteen, joka pitää suorittaa viikonloppuna. Työntekijän palkkaaminen toimistoaikojen ulkopuolelle voi olla kallista. Tällöin voidaan miettiä prosessin automatisointia ohjelmistorobotiikan avulla. (Fung 2014)

3.1.4 Usean tietojärjestelmän käyttäminen

Useiden järjestelmien käyttäminen jossakin työssä voi olla merkki, että työtehtävä kannattaisi ehkä automatisoida. Jos työntekijä joutuu käyttämään useita järjestelmiä, niin manuaalisesta työstä johtuvien mahdollisten virheiden määrä kasvaa. Tämä voi aiheuttaa suuriakin kuluja organisaatiolle. (Fung 2014)

Penttinen ym. (2018) toteavat usean tietojärjestelmän käyttämisen olevan erityisesti kriteeri ohjelmistorobotiikan puolesta, kun sitä verrataan perinteiseen automaatioon, joka vaatisi tällöin muutoksia useiden järjestelmien lähdekoodiin. (Penttinen, Kasslin ja Asatiani 2018)

3.1.5 Tietojärjestelmän ja ympäristön vakaus

Ohjelmistorobotti tarvitsee vakaan ympäristön toimiakseen, joten käytettävien tietojärjestelmien vakaus on yksi kriteeri mietittäessä ohjelmistorobotiikka (Fung 2014). Ohjelmistorobottia suunniteltaessa tulee osata ottaa huomioon, kaikki tilanteet ja poikkeukset, joita ohjelmistorobotti voi työssään kohdata (Asatiani ja Penttinen 2016). Vakaassa tietojärjestelmässä on luonnollisesti helpompi ottaa huomioon kaikki mahdolliset poikkeustilanteet.

Ympäristön vakautena voidaan pitää myös järjestelmän muuttumattomuutta. Penttinen ym. (2018) toteavat, että erityisesti käyttäjärajapinnan pysyminen samanlaisena on yksi kriteeri ohjelmistorobotiikkaa harkitessa. Taustajärjestelmän muutokset eivät ole ohjelmistorobotiikassa niin kriittisiä, kunhan käyttäjärajapinta pysyy samana. Käänteisesti perinteinen automaatio on herkkä taustajärjestelmän muutoksille, mutta ei niin herkkä käyttäjärajapinnan muutoksille. (Penttinen, Kasslin ja Asatiani 2018)

3.1.6 Vähäinen ihmismäisen ajattelun tarve

Ohjelmistorobotti ei pysty ihmismäiseen ajatteluun, vaan se seuraa tiettyjä selkeitä sääntöjä. Jos prosessissa vaaditaan ihmismäistä ajattelua ja valintaa intuition pohjalta, niin prosessi ei sovellu helposti ohjelmistorobotiikalla tehtäväksi. (Fung 2014)

Fung kuitenkin mainitsee artikkelissaan, että ihmismäistä ajattelua vaativat tehtävät eivät välttämättä sulje automaation mahdollisuutta pois, sillä uudet teknologiat sisältävät jo ihmismäistä tekoälyä (engl.artificial intelligence) (Fung 2014). Prosessi on kuitenkin helpommin automatisoitavissa, mikäli se ei vaadi lainkaan tämän tyyppistä tekoälyä.

3.1.7 Vähäinen poikkeuksien käsittely

Poikkeuskäsittelyn tulisi olla minimaalista ohjelmistoroboteille suunnatuissa tehtävissä. Mitä enemmän poikkeuksia käsitellään, sitä hitaammin ohjelmistorobotti suorittaa sille annetun tehtävän. (Fung 2014)

3.1.8 Manuaalisena tehdyn työn hinta

Kun mietitään prosessin automatisoimista, tulee aina arvioida manuaalisena tehdyn työn hinta. Vasta kun manuaalisen työn hinta on tiedossa, voidaan arvioida ohjelmistorobotiikkaan panostamisen kannattavuutta. (Fung 2014)

3.1.9 Manuaalityön aiheuttamien virheiden määrä ja kustannus

Kun mietitään työn automatisointia, on hyvä laskea myös manuaalityön aiheuttamien virheiden kulut. Parhaimmillaan ohjelmistorobotit pystyvät toimimaan paljon ihmistä tarkemmin, jolloin säästetään manuaalityön aiheuttamien virheiden kustannuksilta. (Fung 2014)

4 Ohjelmistotestaus ja -robotiikkatyökalut

Tässä luvussa kartoitetaan ja kuvaillaan teknologioita, jotka soveltuvat ohjelmistorobottien toteutukseen. Luvussa esitellään valikoidusti ohjelmistorobotiikan työkaluja sekä analysoidaan kaupallisten ja avointen työkalujen eroja. Työkalut ovat valikoituneet tarkempaan esittelyyn sen mukaan, mitä olemme tutkimuksen toteutusvaiheessa käyttäneet.

4.1 Avoimet ohjelmistorobotiikkatyökalut

Ohjelmistorobotiikan työkalut ovat pääosin kaupallisia ja usein ne ovat tarkoitettu yritysten liiketoimintaprosessien automatisointiin. Hyviä avoimia ohjelmistorobotiikkaan sopivia työkaluja on tarjolla vain muutama. Tässä luvussa esitellään niistä SikuliX ja Selenium. Tällä hetkellä avoimet ohjelmistorobotiikan työkalut on tarkoitettu vain yksittäisten robottien toteutukseen. Niistä ei löydy kaupallisten työkalujen tapaista hallintajärjestelmää, jonka avulla voidaan hallita ja kerätä dataa suuren robottijoukon toiminnasta.

4.1.1 SikuliX

Sikuli on alun perin kehitetty Massachusettsin teknillisen korkeakoulun (engl. Massachusetts Institute of Technology, MIT) tutkimusprojektiin. Sen kehittämistyötä SikuliX nimellä on jatkanut sittemmin Raimund Hocke. (SikuliX 2017)

SikuliX:lla tehdyt ohjelmistorobotit tukeutuvat toiminnassaan pelkästään kuvatunnistukseen (engl. image recognition). SikuliX IDE:ssä löytyy graafisia työkaluja työskentelyyn, mutta ohjelmistorobotit toteutetaan pääosin skriptejä kirjoittamalla. SikuliX toimii Windows-, macOS- ja Linux-järjestelmissä. Tarkempi esittely SikuliX:n ominaisuuksista löytyy luvusta 5.

4.1.2 Selenium

Selenium on alun perin web-käyttöliittymien testaukseen tarkoitettu avoimen lähdekoodin työkalu (Altaf ym. 2015). Seleniumin toimii testauksessa ihmismäisesti käyttäen web-sivua

graafisen käyttöliittymän välityksellä kuten ihminen. Siksi sitä voidaan käyttää myös ohjelmistorobottien tekoon web-ympäristössä.

Selenium IDE:ssä pystyy nauhoittamaan ja tekemään testitapauksia helposti. Monimutkaisempia testitapauksia voidaan ohjelmoida Seleniumin WebDriver-rajapinnan avulla. Selenium toimii Windows-, macOS- ja Linux-järjestelmissä. (Altaf ym. 2015)

Selenium ymmärtää vain web-sivun rakennetta, jonka avulla se etsii oikeat elementit. Kuvatunnistusta Seleniumissa ei ole. Kaupallisista ohjelmistoista WorkFusion RPA Express hyödyntää Seleniumia tai sen osia omassa toiminnassaan (Workfusion 2018).

4.2 Kaupalliset ohjelmistorobotiikkatyökalut

Kaupallisia ohjelmistorobotiikan työkaluja on tarjolla useita, mutta tässä tutkielmassa käymme läpi tarkemmin kahta: WorkFusion RPA Expressiä ja UiPathia. Nämä kaksi kaupallista työkalua valikoituivat mukaan, koska niistä oli tarjolla ilmaiset versiot testaukseen. Muita kaupallisia ohjelmistorobotiikan työkaluja on lueteltu alaluvussa 4.2.3.

4.2.1 WorkFusion RPA Express

WorkFusion RPA Express on ilmainen ohjelmistorobotiikan työkalu (Workfusion 2018). RPA Express kykenee kuvatunnistuksen lisäksi myös käyttöliittymäelementtien tunnistukseen muilla tavoin. Siinä on tuki muun muassa web-sivuille, Excelille ja tiedostojärjestelmille. Näitä se pystyy käyttämään ilman kuvatunnistusta.

RPA Express sisältää ohjelmistorobottien luomiseen tarkoitettujen työkalujen lisäksi niiden hallintaan tarkoitetun hallintajärjestelmän. Hallintajärjestelmällä pystyy esimerkiksi ajastamaan robottien toimintaa tai saamaan ne välittämään dataa keskenään. Hallintajärjestelmä on kuitenkin ilmaisversiossa asennettavissa vain samalla koneelle kehitystyökalujen kanssa. Maksullinen WorkFusion Smart Process Automation sisältää palvelimella toimivan hallintajärjestelmän, jossa se pääsee todella oikeuksiinsa.

RPA Express on melko raskas käyttää, sillä käyttäjän koneella pyörii samaan aikaan myös hallintajärjestelmä, joka on oikeastaan tarkoitettu toimimaan palvelimella. RPA Express on

näin tehty vähän kuin maistiaisiksi siitä, millaista olisi käyttää robottiarmeijan hallitsemiseen tarkoitettua ohjelmistorobotiikan työkalua, mutta käyttö rajoittuu vain yhdelle laitteelle. Lisäksi RPA Expressin OCR-lisenssissä on rajoitus, jonka vuoksi se vaatii uudelleen asennuksen kolmen kuukauden välein.

4.2.2 UiPath

UiPath on tämän hetken yksi käytetyimmistä ohjelmistorobotiikan työkaluista (Le Clair 2017). UiPath tukee kuvatunnistuksen lisäksi useita eri ympäristöjä, kuten esimerkiksi websivut, Office-tuotteet ja SAP-toiminnanohjausjärjestelmä. Näissä ympäristöissä se pystyy kuvatunnistuksen lisäksi hahmottamaan käyttöliittymäelementtejä myös muilla tavoin.

Tässä tutkielmassa käytimme UiPathin ilmaista Community Edition -versiota. Sitä ei saa käyttää suureen kaupalliseen käyttöön, eikä sillä toteutettuja robotteja voi yhdistää hallintajärjestelmään kuin testaamista varten. Se sopii kuitenkin yksittäisen ihmisen tarpeisiin ja opetuskäyttöön. (UiPath 2018)

UiPath sisältää palvelimella toimivan hallintajärjestelmän, *Orchestratorin*. Hallintajärjestelmän avulla voidaan hallita useassa eri laitteessa toimivia ohjelmistorobotteja yhdestä paikasta. Hallintajärjestelmän voi ostaa pilvipalveluna tai asentaa omalle laitteelleen. Hinnoittelu vaihtelee vaihtoehdosta riippuen. Hallintajärjestelmää käytetään web-käyttöliittymän välityksellä. (UiPath Orchestartor 2018)

4.2.3 Muita kaupallisia työkaluja

Kaupallinen tutkimusyhtiö Forrester listasi helmikuussa 2017 julkaistussa raportissaan 12 merkittävintä ohjelmistorobotiikan työkalujen tarjoajaa tällä hetkellä. Listaan kuuluivat edellä mainittujen UiPathin ja WorkFusionin lisäksi seuraavat kymmenen yhtiötä, joilla on tarjolla oma ohjelmistorobotiikkatyökalu. (Le Clair 2017)

- Automation Anywhere
- Blue Prism
- Contextor

- EdgeVerve Systems
- Kofax
- Kryon Systems
- NICE
- Pegasystems
- Redwood Software
- Softmotive

4.3 Työkalujen vertailua

Tässä alaluvussa käymme läpi Seleniumin, SikuliX:n, WorkFusion RPA Expressin ja UiPathin ominaisuuksien eroja. Yhteenveto vertailusta on koottu taulukkoon 1.

Taulukko 2: Ohjelmistorobotiikan työkalujen ominaisuudet.

	Selenium	SikuliX	WorkFusion	UiPath
Kuvatunnistus		x	x	x
Nauhoitus	x		x	x
Hallintajärjestelmä			x*	x
OCR		x	x*	x
Web-sivut	x		x	x
Excel			x	x
Windows-elementit				x
SAP/Citrix				x
Email				x

*Rajoituksin ilmaisversiossa.

Selenium on työkaluista ainoa, joka ei tue kuvatunnistusta. Tämän takia sen käyttö on rajattu web-sivuihin ja niihin liittyvään automatisointiin. Muilla työkaluilla toteutetut ohjelmistorobotit pystyvät kuvatunnistuksen avulla toimimaan lähes missä tahansa järjestelmässä, jossa on graafinen käyttöliittymä. Kuvatunnistus on kuitenkin hitaampaa verrattuna rakenteeseen

perustuvaan elementtien etsintään.

Nauhoitustoiminto löytyy SikuliX:ia lukuunottamatta kaikista työkaluista. SikuliX IDE:stä löytyy kyllä toimintoja, jotka helpottavat ohjelmistorobotin tekemistä. Tällaisia ovat esimerkiksi automaattisesti avautuva kuvankaappaustila. Kuitenkaan varsinaista nauhoitusta, joka generoisi ohjelmistorobotin pohjan, SikuliX:sta ei löydy.

Ohjelmistorobottien hallintajärjestelmä löytyy WorkFusion RPA Expressistä ja UiPathista. Hallintajärjestelmä auttaa ohjelmistorobottien hallinnassa ja yhteistoiminnan järjestämisessä. RPA Expressissä hallintajärjestelmä rajoittuu vain yhdellä laitteella toimiviin ohjelmistorobotteihin, mutta maksullisessa versiossa tämä rajoitus on poistettu. UiPathissa hallintajärjestelmä toimii palvelimella, ja hallintajärjestelmän avulla voidaan hallita myös eri laitteissa toimivia ohjelmistorobotteja.

Tekstintunnistusominaisuudet (engl. Optical Character Recognition, OCR) löytyvät Seleniumia lukuunottamatta kaikista työkaluista. RPA Expressissä OCR-lisenssi on rajattu ja toimii vain tietyn aikaa ilman uudelleen asennusta. Maksullisessa versiossa tätä rajoitusta ei ole.

Kuvatunnistuksen avulla pystytään käyttämään mitä tahansa järjestelmää, mutta rakenteellinen tunnistus nopeuttaa ja parantaa tunnistuksen tarkkuutta yleensä. Web-ympäristössä rakenteellista tunnistusta pystyvät käyttämään Selenium, RPA Express ja UiPath. Exceltaulukoita ilman kuvatunnistusta pystyvät käyttämään RPA Express ja UiPath. UiPath pystyy käyttämään rakenteellista tunnistusta myös Windows-elementeille sekä SAP- ja Citrix-toiminnanohjausjärjestelmissä.

5 Työkalujen käyttäminen

Tässä luvussa kuvataan tarkemmin ohjelmistorobottien rakentamiseen tarkoitettujen työkalujen käyttöä. Luvussa käydään läpi pääominaisuuksia työkaluista, jotka esiteltiin aiemmin lyhyesti luvussa 4.

5.1 SikuliX

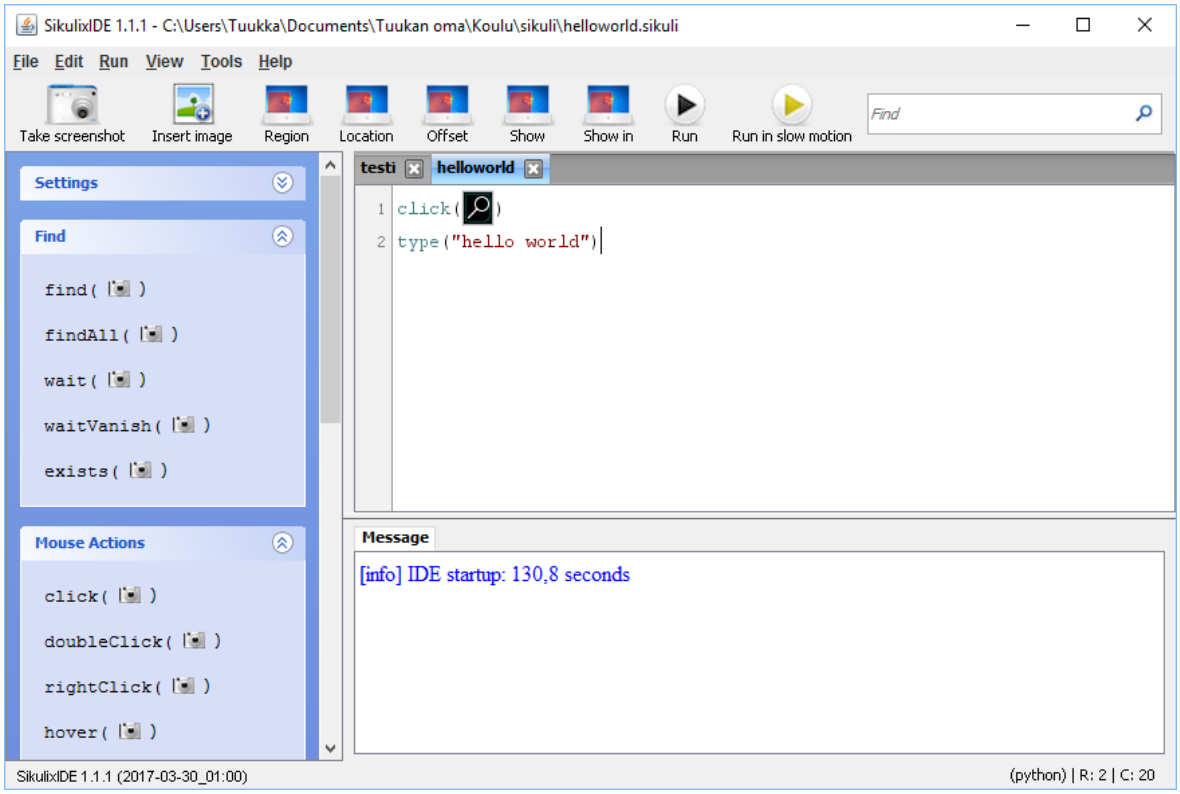
Luvussa esitellään tarkemmin ohjelmistorobottien tekemiseen suunnatun SikuliX:n toimintaa. Luvussa käydään läpi ohjelman käyttöä ja esitellään esimerkkien avulla ohjelmistorobotin luomista.

5.1.1 SikuliX IDE

SikuliX IDE on .sikuli-skriptien luomiseen ja ajamiseen tarkoitettu ohjelmointiympäristö. Sikuli IDE:llä voidaan kirjoittaa skriptejä kolmella kielellä: Pythonilla, Rubylla tai JavaScriptillä (SikuliX 2017). Tämän luvun esimerkeissä käytämme Pythonia.

5.1.1.1 Käyttöliittymä

Alla näkyy (Kuvio 1) SikuliX IDE:n käyttöliittymä, jossa on auki lyhyt Hello World -esimerkki. Yläpalkin valikoista löytyvät mahdollisuudet tiedostojen lataukselle ja tallennukselle sekä tekstin muokkaukselle. Yläpalkista löytyy myös mahdollisuus päästä säätämään IDE:n asetuksia. Esimerkiksi tekstintunnistusominaisuus (engl. Optical character recognition, OCR) tulee kytkeä erikseen päälle asetuksista. Tekstintunnistusominaisuus on myös täytynyt valita asennuksen yhteydessä asennettaviksi, jotta se toimii.



Kuvio 1. SikuliX IDE:n käyttöliittymä.

Yläpalkin alapuolella löytyvät pikavalinnat. Esimerkiksi kuvankaappausmahdollisuus mallikuville, joiden avulla ohjelmistorobotti käyttää käyttöliittymää, löytyy täältä. SikuliX IDE:n työkalujen avulla kuvankaappauksen ottaminen kuville, joita halutaan käyttää koodissa, on helppoa. SikuliX IDE osaa säilöä kuvat oikeaan projektikansioon, jolloin kuva on heti käytettävissä skriptiä kirjoittaessa, eikä sen sijaintia tarvitse miettiä. IDE myös näyttää käytetyn kuvan koodin seassa. Tämän ominaisuuden voi kytkeä pois asetuksista, jolloin koodissa näkyy vain kuvan tiedostonimi.

Pikavalinnoista löytyy myös koodin ajaminen normaalisti tai hidastetusti. Sen lisäksi löytyvät erilaiset aluevalinnat ja mallikuvan lisäys hakemistosta.

Vasemmasta sivupalkista saadaan SikuliX:ssa käytössä olevat peruskomennot. Näiden komentojen lisäksi SikuliX:sta löytyy muitakin komentoja, joista lisää SikuliX:n dokumentaatiossa (SikuliX 2017). Komentoja klikkaamalla IDE kirjoittaa komennon automaattisesti koodialueelle kohtaan, jossa kursori kulloinkin on. Jos komentoon sisältyy mallikuva, niin

IDE kirjoittaa komennon koodialueelle, pienentää IDE:n ja käynnistää sitten kuvankaappaustyökalun, jolla käyttäjä voi valita halutun kohteen. Kaapattu kuva lisätään välittömästi koodin sekaan. Tämä tekee peruskomentojen kanssa työskentelystä helppoa.

Esimerkkinä click-komento, jolla ohjelmistorobotti saadaan painamaan hiiren valintapainiketta halutun näköisen kohteen päällä. Click-komennolle pitää syöttää parametrinä mallikuva, jota sen tulisi painaa. Valitaan vasemmalta click-komento. IDE siirtyy kuvankaappaustilaan. Kuvankaappaustilassa valitsemme halutun kohteen rajaamalla sen suorakulmion muotoisen alueen sisään. Näkymä palaa takaisin IDE:hen, joka on kirjoittanut koodialueelle:

```
click(kaapattukuva.png)
```

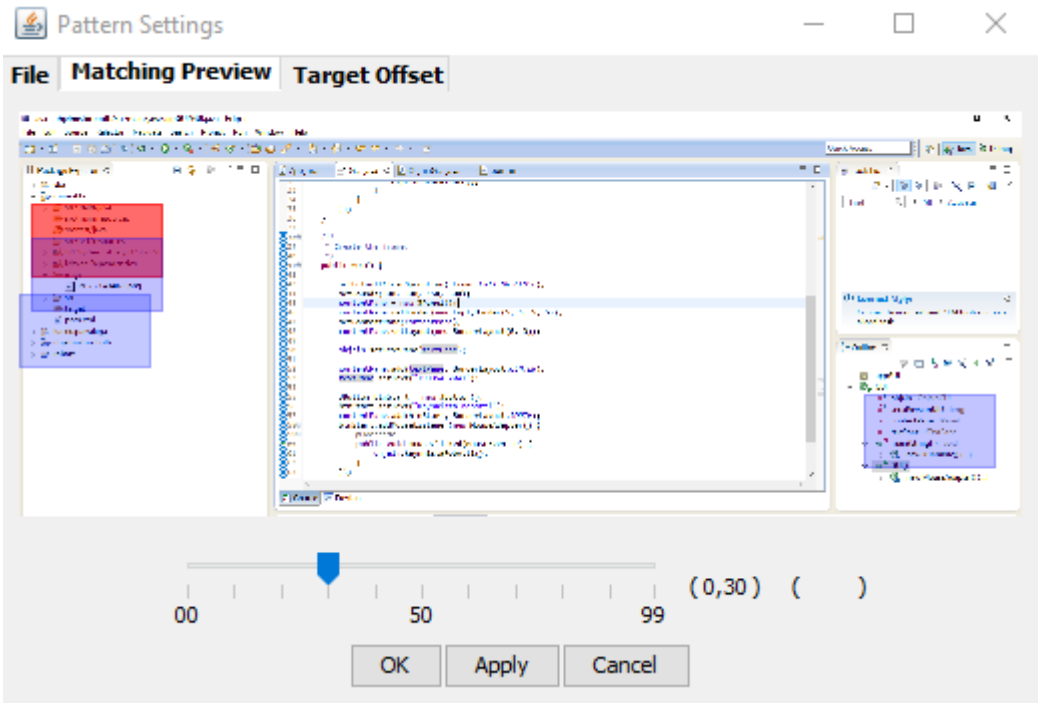
Koodia voi ja pitää kirjoittaa koodialueelle myös käsin. Tällöin kuvankaappaustila ei käynnisty IDE:ssä automaattisesti sitä vaativien komentojen kohdalla, vaan se pitää laittaa erikseen päälle.

Käyttöliittymän alalaidasta löytyy *Message*-alue, johon SikuliX kirjoittaa koodin ajoaikana lokia. Tietoa saadaan esimerkiksi ohjelman suoritusnopeudesta. Myös virhe- ja muut ilmoitukset ilmoitetaan *Message*-alueella.

5.1.1.2 Kuvatunnistuksen asetusten säätäminen

SikuliX IDE:llä kaiken voi kirjoittaa skripteillä, mutta IDE:ssä on myös graafisia työkaluja. Yksi graafinen työkalu on kuvatunnistuksen asetusten (engl. Pattern settings) säätäminen. Kun IDE:ssä mallikuvat näkyvät suoraan koodissa kuvan 1 tapaan, niin niitä klikkaamalla avautuu kuvatunnistuksen asetukset. Mallikuvat saa näkymään IDE:ssä valikosta *View->Show thumbnails*.

Kuvatunnistuksen asetuksista voidaan esimerkiksi säätää, kuinka tarkka kuvatunnistus on. Mitä tarkempi kuvatunnistus on, sitä kauemmin tunnistuksessa kestää. Oletuksena kuvatunnistuksen tarkkuus on 0.7 (SikuliX 2017). Työkalulla oikean tarkkuuden valitseminen helpottuu. Työkalu näyttää, mitkä kuvat kuvatunnistin tunnistaa samanlaisiksi rajaamalla ne sinisellä suorakulmiolla (Kuvio 2). Punaisella rajataan alue, josta mallikuva on otettu.

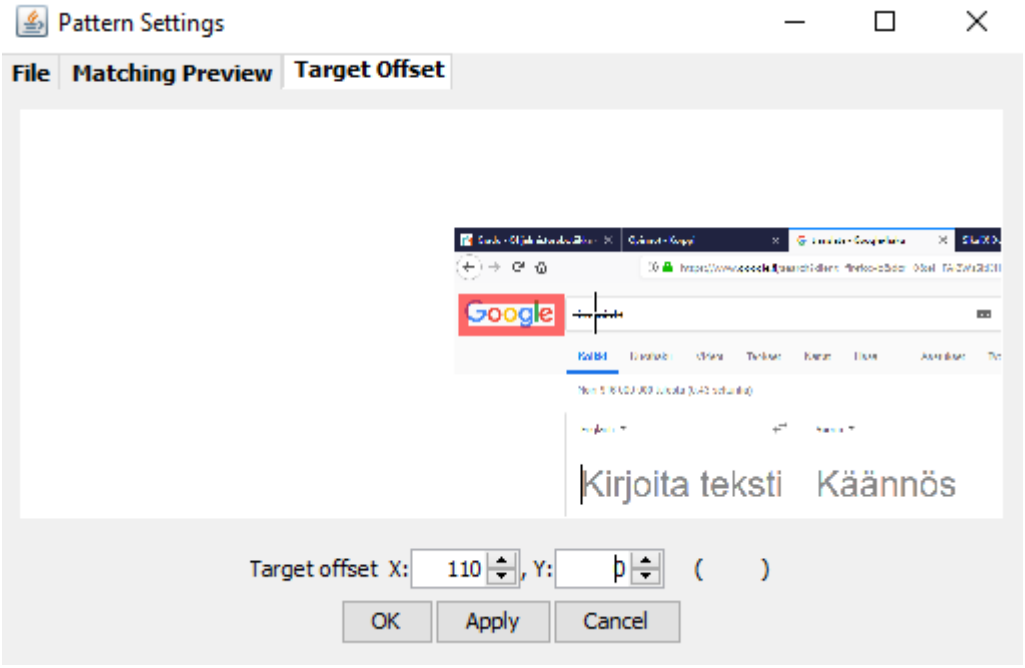


Kuvio 2. Kuvatunnistuksen asetusten säätäminen.

Kuvatunnistuksen tarkkuus voidaan määrittää myös suoraan koodissa näin:

```
click(Pattern("1518081130599.png").similar(0.40))
```

Kuvatunnistuksen asetuksista voidaan säätää myös kuvan painalluspiste (engl. Click point). Tästä on hyötyä, jos esimerkiksi halutaan etsiä kuvan avulla tietty lomakkeen kentän selite ja klikata kirjoituskenttää sen vierestä. Tällöin painalluspiste voidaan määrittää kuvasta tietyn verran sivuun. Kuvio 3 havainnollistaa painalluspisteen määrittämistä työkalun avulla.



Kuvio 3. Painalluspisteen määrittäminen mallikuvan suhteen.

Kuvan painalluspiste voidaan määrittää myös suoraan koodissa näin:

```
click(Pattern("1518083165984.png").targetOffset(110,0))
```

5.1.1.3 Tekstintunnistus

Tekstintunnistuksella (engl. Optical character recognition, OCR) tarkoitetaan teknologiaa, jonka avulla pystytään tunnistamaan tekstiä kuvasta ja kääntämään se sähköisesti muokattavaan muotoon (Menard 2008). SikuliX käyttää tekstintunnistukseen vapaan lähdekoodin tekstintunnistuskirjastoa nimeltä Tesseract (SikuliX 2017). SikuliX käyttää Tesseractin englanninkielistä versiota, joten esimerkiksi skandinaavisten merkkien tunnistus on puutteellista.

Tekstintunnistus kytketään SikuliX IDE:ssä päälle kohdasta *File->Preferences->General options->more options->TextSearch and OCR*. Tekstintunnistusominaisuudet on pitänyt valita asennettaviksi SikuliX:n asennusvaiheessa tai ne eivät toimi.

Seuraavassa yksinkertainen esimerkki tekstintunnistuksen käyttämisestä SikuliX:ssa. Esimerkissä olemme kirjoittaneet Muistioon tekstiä, joka luetaan ja kirjoitetaan uudelleen Muis-

tioon kursorin kohdalle. Koodi toimii, kun tarvittavat mallikuvat on otettu oikein.

```
click("1517390434071.png") #Avataan Muistio ikonia napsauttamalla
region = find("1517390540437.png")  #etsitään Muistion yläkulma kuvan avulla
                                    #ja tallennetaan yläkulman koordinaatit
                                    #Region-tyyppiseen olioon
region.y = region.y + 50 #liikutaan 50 pikseliä alas kirjoitusalueelle
region.w = 250 #asetetaan alueen leveys pikseleinä
region.h= 200 #asetetaan alueen korkeus pikseleinä
teksti = region.text() #luetaan alueen sisältämä teksti
paste(teksti) #liitetään teksti muistioon nähtäville
```

5.1.2 SikuliX API

SikuliX:a pystyy käyttämään myös Java-koodissa liittämällä Java-projektiin SikuliX API -kirjaston (SikuliX 2017). Tämä mahdollistaa SikuliX:n skriptien kirjoittamisen Javalla. SikuliX API mahdollistaa myös sen, että SikuliX ohjelmia voidaan ajaa muualtakin, kuin SikuliX IDE:stä. Ne voidaan esimerkiksi pakata suoritettaviksi .jar-tiedostoiksi. Lisäksi Java-ohjelmointiympäristö tarjoaa mahdollisuuden myös muiden Java-kirjastojen hyödyntämiseen. Ohjelmistorobotille on mahdollista tehdä vaikka oma käyttöliittymä esimerkiksi hyödyntämällä Javan omaa Swing-kirjastoa graafisen käyttöliittymän luomiseen.

Tässä alaluvussa käymme läpi SikuliX API:n käyttöä. Kirjoitamme Javaa Eclipse-ohjelmointiympäristössä ja havainnollistamme SikuliX-kirjaston liittämistä Java-projektiin.

5.1.2.1 SikuliX API -kirjaston lisääminen Java-projektiin

SikuliX:n asennuksen yhteydessä voidaan valita, asennetaanko mukana myös SikuliX API. Jos Sikulix API on valittu asennettavaksi, se löytyy samasta kansiosta kuin Sikulix IDE nimellä *sikulixapi.jar*. Kirjasto lisätään Eclipsessä Java-projektiin valitsemalla projekti hiiren oikealla painikkeella *Preferences->Add Library*. (SikuliX 2017)

SikuliX API voidaan lisätä myös automaattisesti Maven-projektiin. Maven on projektinhallintatyökalu Java-projektien hallintaan (Apache Maven 2018). SikuliX API:n lisääminen tapahtuu Maven-projektissa lisäämällä *pom.xml* tiedoston kohtaan *dependencies* seuraava koo-

di:

```
<dependency>
    <groupId>com.sikulix</groupId>
    <artifactId>sikulixapi</artifactId>
    <version>1.1.0</version>
</dependency>
```

(SikuliX 2017)

5.1.2.2 SikuliX API:n käyttäminen

SikuliX API lisätään ohjelmakoodiin muiden kirjastojen tapaan koodilla:

```
import org.sikuli.script.*;
```

Itse SikuliX API:n käyttäminen Java-koodissa on hyvin samanlaista kuin SikuliX-skriptien kirjoittaminen SikuliX IDE:ssä. Kielenä toimii nyt vain Java. Alla oleva esimerkki havainnollistaa Java-koodin kirjoittamista.

```
import org.sikuli.script.*;

public class TestSikuli {

        public static void main(String[] args) {
                Screen s = new Screen();
                try{
                        s.click("imgs/spotlight.png");
                        s.wait("imgs/spotlight-input.png");
                        s.click();
                        s.write("hello world#ENTER.");
                }
                catch(FindFailed e){
                        e.printStackTrace();
                }
        }

}
```

(SikuliX 2017)

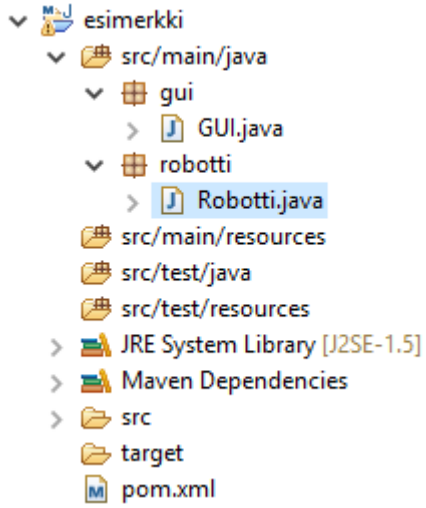
SikuliX IDE:n kanssa työskentelystä poiketen, SikuliX API:n komentoja käytettäessä on ensin luotava *Screen* -olio, joka vastaa näyttöruutua. *Screen* -oliolla on metodeinaan kaikki SikuliX:n peruskomennot. Tarkemmin SikuliX API:n ominaisuuksista voi lukea SikuliX API:n dokumentaatiosta (Sikulix API 2018).

5.1.2.3 Käyttöliittymä ohjelmistorobotille

Tässä alaluvussa esittelemme käyttöliittymän lisäämisen omalle ohjelmistorobotille Eclipse-ohjelmointiympäristössä. Toteutamme käyttöliittymän Javan mukana tulevalla Swing-käyttöliittymäkirjastolla.

SikuliX API käyttää Java AWT:n *Robot*-luokkaa, jota ei voida kutsua suoraan Swing-kontekstista. Tämän takia Swing-käyttöliittymä ja SikuliX API:n toiminnot tulee erottaa erillisiksi säikeiksi. Ohjelmistorobotin logiikka tulee sijoittaa luokkaan, joka toteuttaa *Runnable*-rajapintaa. Tätä luokkaa kutsutaan käyttöliittymästä, kun halutaan käynnistää ohjelmistorobotti.

Ensiksi luomme Maven-projektin, johon lisäämme kirjastoiksi mukaan SikuliX API:n (luku 5.2.1). Luomme erikseen käyttöliittymä-luokan ja robotti-luokan. Projektin kansiorakenne Eclipsessä on alla esitetyn (Kuvio 4) mukainen, jossa *GUI.java* on käyttöliittymä-luokka ja *Robotti.java* ohjelmistorobotin logiikan sisältävä luokka.



Kuvio 4. Kansiorakenne Eclipsessä, missä eroteltu käyttöliittymä ja robotin logiikka toisistaan.

Käyttöliittymäluokan (*GUI.java*) pohjan luomme WindowBuilderilla, joka on Eclipseen ladattava Java-käyttöliittymien muokkaukseen tarkoitettu työkalu. Se tulee usein mukana jo Eclipsen asennuksen yhteydessä. WindowBuilderista valitsemme luotavaksi *Swing Designerin* alta *JFramen*, joka luo meille valmiin käyttöliittymäikkunan pohjan. Lisäämme *Design*-tilassa *Käynnistä*-painikkeen käyttöliittymäikkunaan. Tämän jälkeen siirrymme *Design*-tilasta *Source*-tilaan muokkaamaan käyttöliittymäpohjamme koodia.

Lisäämme *GUI*-luokan alkuun koodin:

```
import robotti.Robotti;
```

Näin voimme kutsua *Robotti*-luokkaa, jossa sijaitsee ohjelmistorobotin logiikka. Lisäämme nyt luokan konstruktoriin tapauksen käsittelijän, kun *Käynnistä*-painiketta painetaan. Tapauksen käsittelijässä luomme uuden säikeen, jossa luomme *Robotti*-luokan olion, sekä käynnistämme säikeen:

```
btnKaynnistaRobotti.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Thread t = new Thread(new Robotti()); // Luodaan säie
        t.start(); // käynnistää säikeen
    }
```

```
});
```

Robotti-luokkamme taas toteuttaa *Runnable*-rajapintaa. Luokan pohja on seuraavanlainen:

```
package robotti;
import org.sikuli.script.*; //SikuliX API-kirjasto

public class Robotti implements Runnable{

    public void run() {
        //Koodi joka suoritetaan kun säie ajetaan:
        Screen s = new Screen(); //SikuliX API-kirjaston olio
        try{
            s.click("imgs/textbox.png"); //SikuliX API-kirjaston metodi
            s.type("Hello World!"); //SikuliX API-kirjaston metodi
        }
        catch(FindFailed e){
            e.printStackTrace();
        }

    }
}
```

Luokalla on siis yksi metodi nimeltä `run`. Kyseinen metodi suoritetaan, kun säie käynnistetään käyttöliittymässä komennolla `t.start()`. Tänne voimme kirjoittaa ohjelmistorobotin logiikan.

Nyt olemme toteuttaneet ohjelmistorobotin omalla käyttöliittymällä. Ajamalla koodi avautuu käyttöliittymä, jossa on yksi käynnistä-painike. Painiketta painamalla ohjelmistorobotti alkaa toimia eli tässä tapauksessa etsii tekstikentän mallikuvan avulla, klikkaa sitä ja kirjoittaa näppäinpainalluksia imitoiden “Hello World!”.

5.2 Selenium

Selenium on ilmainen ja alun perin web-käyttöliittymien testaukseen tarkoitettu ohjelmisto. Sillä on kuitenkin mahdollista toteuttaa ohjelmistorobotteja muuhunkin käyttöön web-

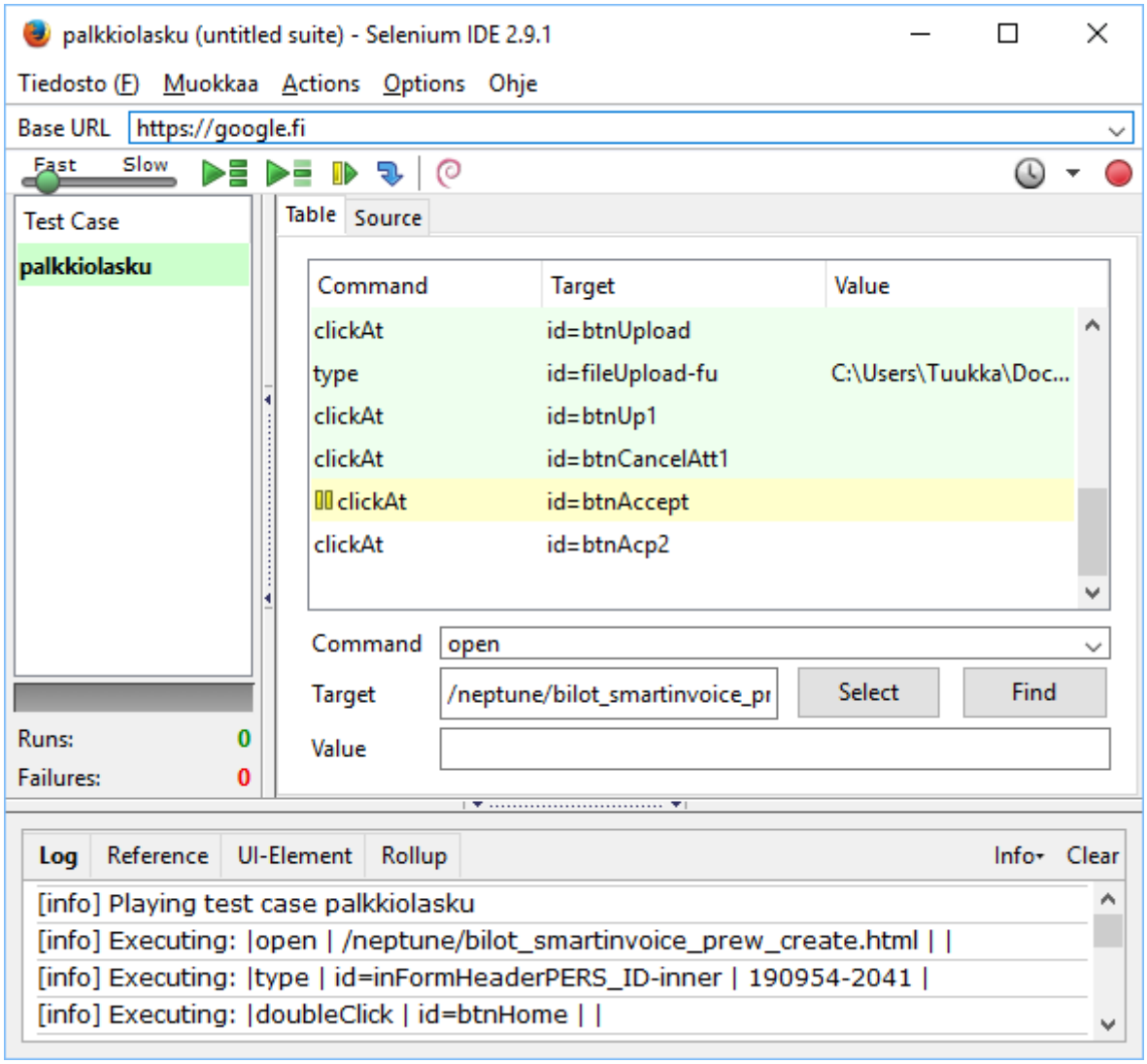
ympäristöissä. Seleniumia voi käyttää selaimen lisäosan, Selenium IDE:n, avulla tai ohjelmointiympäristössä Selenium WebDriverin avulla. Tässä alaluvussa käymme läpi Seleniumin peruskäyttöä.

5.2.1 Selenium IDE

Selenium IDE on selaimeen asennettava lisäosa, jolla on mahdollista luoda ohjelmistorobotteja. Selenium IDE toimii Google Chromessa ja Mozilla Firefoxissa. Selenium IDE sisältää nauhoitustoiminnon, jonka avulla käyttäjä voi luoda ohjelmistorobotin suorituskaavion pohjan nauhoittamalla omaa toimintaansa. Ohjelmistorobotti on mahdollista myös viedä Selenium IDE:stä myös ohjelmakoodiksi, jolloin ohjelmistorobottia voi muokata ohjelmointiympäristössä.

5.2.1.1 Käyttöliittymä

Alla näkyy (Kuvio 5) Selenium IDE:n käyttöliittymä. Yläpalkista löytyy tiedostojen hallinta ja mahdollisuus viedä ohjelmistorobotti ohjelmakoodiksi. Versiossa 2.9.1 tuettuja ohjelmointikieliä ovat Java, C#, Ruby ja Python. Yläpalkista voidaan säätää myös Selenium IDE:n asetuksia.



Kuvio 5. Selenium IDE:n käyttöliittymä.

Yläpalkin alta löytyvät ohjelmistorobotin suoritusnopeuden säätömahdollisuus ja muita ohjelmistorobotin suorittamiseen liittyviä painikkeita. Oikealla punainen ympyrä on nauhoituspainike, josta voidaan aloittaa nauhoitustoiminto, joka generoi suorituskaavion ohjelmistorobotille käyttäjän toimintojen mukaan.

Vasemmassa sivupalkissa näkyvät testitapaukset (engl. test case) eli yksittäiset ohjelmistorobotit. Selenium on tarkoitettu käyttöliittymätestaukseen, jossa yksi testipaketti (engl. test suite) voi sisältää useamman testitapauksen. Tässä esimerkissä käsittelemme yhtä testitapausta yhtenä yksittäisenä ohjelmistorobottina.

Keskellä on ohjelmistorobotin suorituskaavio, jonka voi valita näkyvän taulukkomuodossa tai XML-muotoisena ohjelmakoodina. Taulukkomuodossa suorituskaaviota voi muokata klikkaamalla haluttuja komentoja ja vaihtamalla komentojen arvoja suorituskaavion alla oleviin kenttiin. Komentoja voi lisätä suorituskaavioon hiiren oikealla painikkeella. Poistaminen tapahtuu delete-painikkeella. XML-muotoisessa näkymässä muokkaaminen, lisääminen ja poistaminen tapahtuvat XML-muotoista ohjelmakoodia kirjoittamalla.

Käyttöliittymän alalaidassa näytetään loki, jota kirjoitetaan kun ohjelmistorobotti ajetaan. Lokiin ilmestyvät suoritukseen liittyvät asiat, joita ovat esimerkiksi virhetilanteet.

5.2.1.2 Nauhoitus

Selenium IDE:n nauhoitustilassa ei ole juuri säätövaraa. Käyttäjä laittaa nauhoitustilan päälle ja alkaa suorittaa toimintoja selainikkunassa. Näistä toiminnoista muodostetaan ohjelmistorobotin suorituskaavio, jota voidaan muokata jälkikäteen Selenium IDE:ssä. Selenium käyttää elementtien tunnistukseen sivun HTML-koodia. Kuvatunnistus ei ole Seleniumissa mahdollinen. Nauhoitustilassa korostetaan hiiren alle jääviä käyttöliittymäelementtejä, kun käyttäjä suorittaa nauhoitusta.

Nauhoitustilan generoima ohjelmistorobotti ei aina toimi kuten se nauhoitettiin. Esimerkiksi jotkut elementtien painallukset eivät välttämättä toimi, vaan käyttäjä joutuu jälkikäteen muokkaaman painallustoimintoa esimerkiksi vaihtamalla toiminnon *click* toimintoon *clickAt*.

5.2.2 Selenium WebDriver

Selenium WebDriver mahdollistaa ohjelmistorobottien toteuttamisen ohjelmoimalla. WebDriver ohjelmointirajapinta tarjoaa pääsyn selaimen käyttämiseen. Selenium Webdriver tukee useita eri selaimia ja ohjelmointikieliä. Tuettuja ohjelmointikieliä ovat Java, C#, Python, Ruby, Javascript, Perl ja PHP. Tämän alaluvun esimerkeissä käytämme Javaa ja selaimena toimii Google Chrome. (Selenium 2018)

5.2.2.1 Seleniumin lisääminen Java-projektiin

Selenium käyttäminen Eclipsen Java-ympäristössä onnistuu luomalla Java Maven-projekti. Maven on projektinhallintatyökalu Java-projektien hallintaan (Apache Maven 2018). Seleniumin lisääminen projektiin tapahtuu lisäämällä *pom.xml* -tiedoston kohtaan *dependencies* seuraava koodi:

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-server</artifactId>
    <version>3.0.1</version>
</dependency>
```

(Selenium 2018)

5.2.2.2 Java-ohjelmointi

Seleniumia käytetään Java-koodissa lisäämällä koodin alkuun:

```
import org.openqa.selenium.*
```

Kuitenkaan koko kirjastoa ei kannata liittää projektiin, vaan pelkästään käytettävät ominaisuudet, sillä muuten projektista tulee raskas. Esimerkiksi jos haluamme käyttää Chrome-selaimen ohjaamiseen soveltuvaa *ChromeDriveria*, lisäämme koodin alkuun:

```
import org.openqa.selenium.chrome.ChromeDriver;
```

Selenium-kirjastoa käytetään ohjelmoinnissa kuten muitakin kirjastoja. Kirjaston metodit antavat poikkeuksen, jos niiden käytössä tulee virhe. Esimerkiksi jos käyttöliittymäelementtiä ei löydetä, niin metodi antaa poikkeuksen. Tämä takia poikkeuksienkäsittely on tärkeä muistaa Selenium-kirjastoa käytettäessä. Alla oleva ohjelmointiesimerkki havainnollistaa kirjaston käyttöä.

```
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.chrome.ChromeDriver;
```

```
public class esimerkki {

    public static void main(String[] args){
        //luodaan selaimen ohjain:
        ChromeDriver driver = new ChromeDriver(options);
        //Avataan oikea sivu:
        String baseUrl = "https://www.google.fi";
        //Klikataan nappia sivulla:
        try {
            driver.findElement(By.id("hplogo")).click();
        } catch (Exception e) {
            System.out.println("Tapahtui virhe");
            e.printStackTrace();
        }
        driver.close(); // suljetaan selaimen ohjain
    }
}
```

5.3 WorkFusion RPA Express

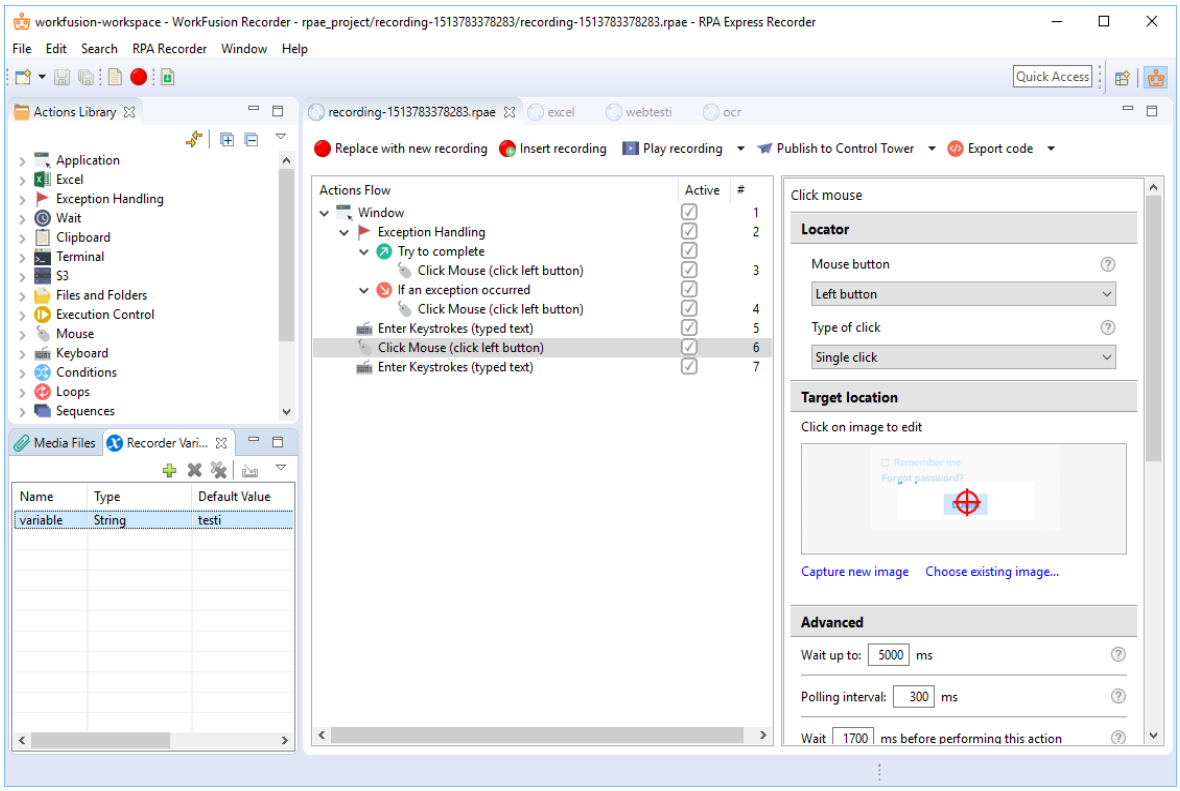
Tässä alaluvussa käsitellään WorkFusion RPA Expressin käyttöä. RPA Express on ilmainen ohjelmistorobotiikan työkalu, jossa on kuitenkin omat toiminnalliset rajoitteensa (luku 4.2.1). Sitä voi kuitenkin käyttää kaupalliseenkin käyttöön ilman rajoituksia. Maksullinen WorkFusion Smart Process Automation sisältää tarvittaessa enemmän toimintoja.

5.3.1 RPA Recorder

RPA Recorder on WorkFusion RPA Expressin ohjelmiston osa, jolla ohjelmistorobotit luodaan. Käyttäjä voi nauhoittaa toimiaan, minkä pohjalta RPA Recorder luo robotille kaavion säännöistä, joiden mukaan robotin tulee toimia. RPA Recorderissa tätä kaaviota voi myös muokata.

5.3.1.1 Käyttöliittymä

Alla näkyy (Kuvio 6) RPA Recorderin käyttöliittymä. Yläpalkista löytyvät tiedostojen hallinta sekä mahdollisuudet RPA Recorderin asetusten muokkaukselle. Vasemmasta palkista löytyvät RPA Recorderin osaamat komennot. Vasemmalla alhaalla on kuvassa 7 auki muuttujien hallinta. Ohjelmistorobotti voi suorituksensa aikana kirjoittaa muuttujiin tietoa tai lukea niiden arvoja.



Kuvio 6. RPA Recorderin käyttöliittymä.

Käyttöliittymän keskellä on ohjelmistorobotin suorituskaavio. Suorituskaavio kuvaa ohjelmistorobotin halutun toiminnan. Komennot suoritetaan järjestyksessä ylhäältä alas. Kaavioon raahataan komentoja vasemmasta sivupalkista.

Oikeasta sivupalkista voidaan tarkemmin määritellä yhden valitun komennon toimintaa. Sivupalkista voidaan esimerkiksi säätää kuvatunnistuksen asetuksia ja vaihtaa mallikuva. Jos halutaan käyttää rakenteeseen perustuvaa elementtien etsintää web-pohjaisissa järjestelmissä, niin voidaan määrittää elementtien XPath, joka kertoo etsittävän elementin sijainnin sivun

rakenteessa (W3C 2017).

Nauhoitustila avataan punaisesta ympyrästä, joko korvaamalla nykyinen suorituskaavio kohdasta “Replace with new recording” tai lisäämällä nauhoitettava osuus suorituskaavioon kohdasta “Insert recording”. Suorituskaavion voi toistaa kohdasta “Play Recording” ja sen voi julkaista hallintaohjelmistoon kohdasta “Publish to Control Tower”.

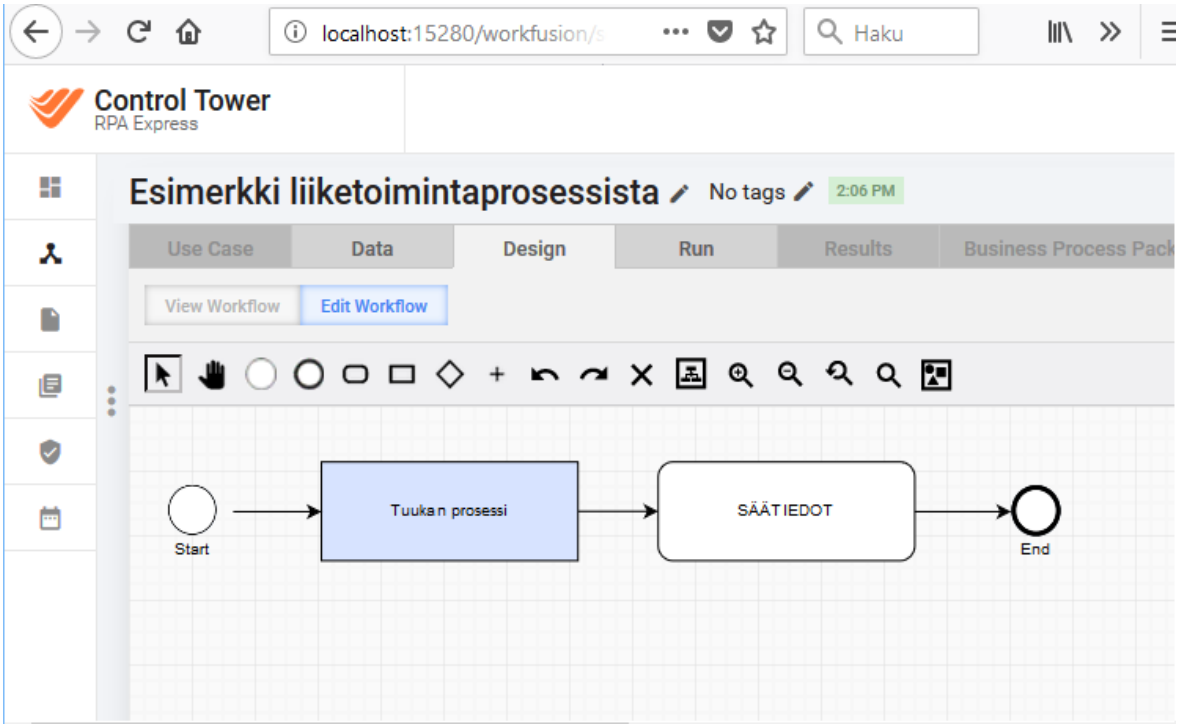
5.3.1.2 Nauhoitus

RPA Expressin nauhoitustila on yksinkertainen. Käyttäjä laittaa nauhoitustilan päälle ja alkaa suorittaa toimintoja, joita haluaa ohjelmistorobotin tekevän. Näistä toiminnoista muodostetaan ohjelmistorobotin suorituskaavio. RPA Express käyttää nauhoitustilassa kuvatunnistusta. Nauhoitustilassa ei siis voida tunnistaa elementtejä esimerkiksi lähdekoodin avulla. Jos halutaan tunnistaa elementtejä muilla tavoin, niin suorituskaaviota pitää muokata nauhoituksen jälkeen käsin.

5.3.2 Control Tower

Control Tower on ohjelmistorobottien hallintajärjestelmä WorkFusion RPA Expressissä. Control Tower toimii web-käyttöliittymän avulla. Sen avulla voidaan aikatauluttaa ohjelmistorobottien toimintaa ja luoda liiketoimintaprosesseja. Lisäksi Control Towerista saadaan tietoa ohjelmistorobottien tilasta. Control Tower toimii samassa laitteessa kuin muutkin RPA Expressin osat. Palvelimelle asennettava hallintajärjestelmä on tarjolla maksullisessa WorkFusion Smart Process Automationissa. (RPA Express 2018)

Business Process -välilehdeltä voidaan Control Towerissa luoda liiketoimintaprosesseja. Liiketoimintaprosessit tarkoittavat RPA Expressissä usean ohjelmistorobotin luomia kokonaisuuksia, joissa ohjelmistorobotteja ketjutetaan toisiinsa. Esimerkiksi yksi ohjelmistorobotti hakee tietoa pörssikursseista web-sivulta, joiden tiedot se välittää toiselle ohjelmistorobotille, joka analysoi tietoja ja tekee niistä yhteenvedon taulukkomuotoon, jota käyttäjä voi tarkastella.



Kuvio 7. Liiketoimintaprosessi luodaan piirtämällä työnkulkukaavio.

Liiketoimintaprosessit voivat sisältää täysin automaattisia tehtäviä (engl. bot task) ja manuaalisesti suoritettavia tehtäviä (engl. manual task). Liiketoimintaprosessissa voi esimerkiksi olla täysin automaattinen ohjelmistorobotin suorittama tehtävä, jossa ohjelmistorobotti muuntaa PDF-lomakkeen tiedot HTML-muotoon. Manuaalisessa tehtävässä käyttäjä tarkistaa, ovatko tiedot oikein ja hyväksyy tietojen lähetyksen eteenpäin. Manuaalisia tehtäviä voidaan suorittaa RPA Expressissä *WorkSpace* -tilassa.

Control Towerissa manuaalisia tehtäviä voidaan luoda *Manual Tasks* -välilehdellä. Manuaalinen tehtävä luodaan antamalla esimerkkidata ja kuinka se esitellään datan tarkistussivulla. Manuaaliset tehtävät liittyvät RPA Expressissä vain datan tarkistamiseen.

Ohjelmistorobottien ja liiketoimintaprosessien suorittamista voi aikatauluttaa *Schedules*-välilehdeltä. Tätä kautta ohjelmistorobotti voidaan laittaa käynnistymään vaikka kerran tunnissa. Ongelma RPA Expressissä on, että se toimii vain yhdellä koneella, joten tämä kone ja Control Towerin pitää olla päällä, jotta robottien aikataulutus toimii. Maksullisessa WorkFusion Smart Process Automationissa Control Tower toimii palvelimella, jolloin ohjelmistorobot-

tien ajaminen onnistuu, kunhan vain palvelinkone pyörii.

5.4 UiPath

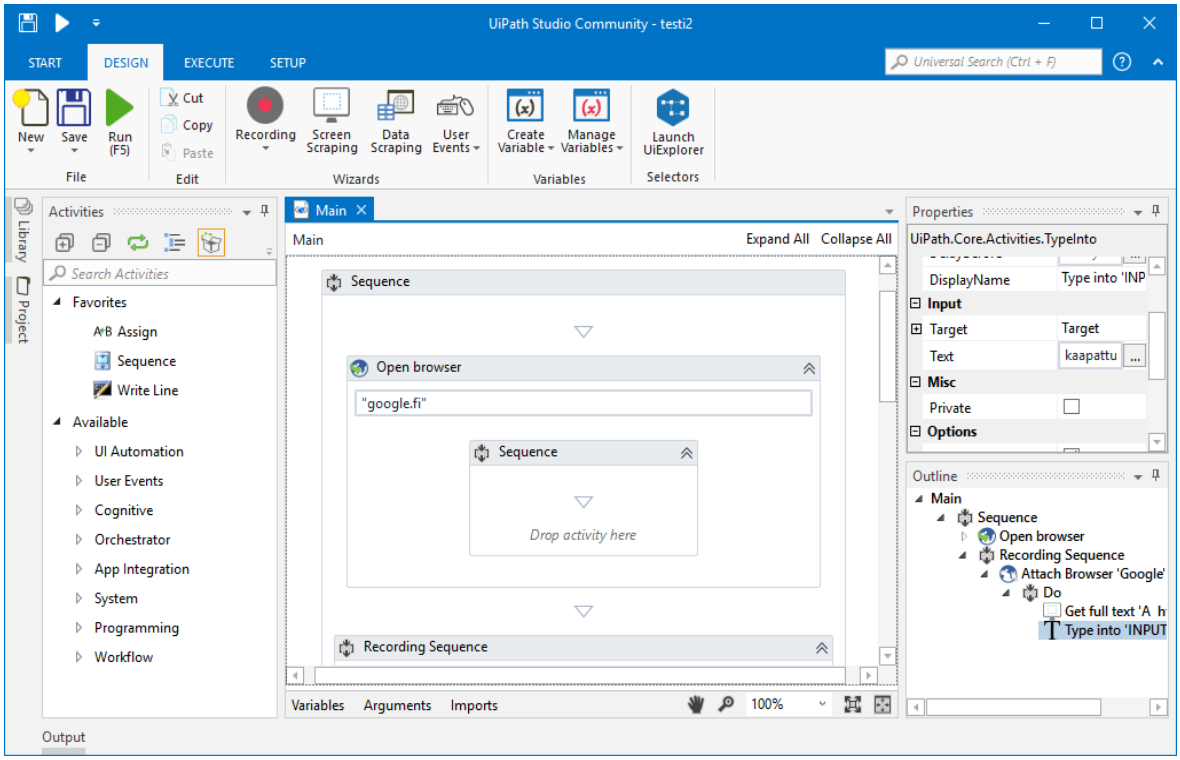
Luvussa esitellään ohjelmistorobottien tekoon tarkoitetun UiPathin käyttöä. UiPath on kaupallinen ohjelmistorobotiikan työkalu, josta on olemassa myös ilmainen *Community Edition* -versio. Ilmaisversiossa on kuitenkin rajoituksensa. Sitä ei saa käyttää suureen kaupalliseen käyttöön, eikä sillä toteutettuja robotteja voi yhdistää hallintajärjestelmään kuin testaamista varten. (UiPath 2018)

5.4.1 UiPath Studio

UiPath Studio on ohjelmistorobottien toteutukseen käytetty sovellus. Se sisältää työkaluja ohjelmistorobottien nauhoituksen, muokkaamisen ja testaamiseen.

5.4.1.1 Käyttöliittymä

Alla näkyy (Kuvio 8) UiPath Studion käyttöliittymä. Vasemmassa palkissa tarjolla ovat käytettävissä olevat komennot, keskellä on ohjelmistorobotin suorituskaavio ja oikeassa sivupalkissa voi säätää yksittäisen komennon parametrejä tarkemmin.



Kuvio 8. UiPath Studion käyttöliittymä.

Yläpalkista löytyvät tiedostojenhallinta, nauhoitus, muuttujien säätäminen ja vastaavat toimet. Ohjelmistorobotti on esimerkiksi mahdollista asettaa reagoimaan käyttäjän aiheuttamiin tapahtumiin. Käyttöliittymä on hyvin *Microsoftin Office* -tuotteiden henkinen, mikä voi olla peruskäyttäjälle tutun tuntuista.

5.4.1.2 Nauhoitus

Nauhoitus aloitetaan UiPath Studiossa valitsemalla millaisessa käyttöliittymä ympäristössä nauhoitusta tehdään. Valittava ympäristö voi olla tavallinen, web-käyttöliittymä, työpöytä tai Citrix. Ympäristön valinta vaikuttaa siihen, osaako UiPath tulkita käyttöliittymäelementtejä tietorakenteista vai tyytyykö se pelkkään kuvatunnistukseen.

Yksinkertaisimmillaan käyttäjä nauhoittaa nauhoitustilassa toimintansa, josta UiPath muodostaa ohjelmistorobotin suorituskaavion, jota voi muokata UiPath Studiossa. Nauhoitustila näyttää nauhoittaessa, mitä käyttöliittymäelementtejä se tunnistaa, kun käyttäjä vie hiiren

kursorin elementtien päälle. Tarvittaessa nauhoitustila osaa ehdottaa ankkurikohteiden määrittämistä, jos käyttöliittymäelementti ei ole yksiselitteinen.

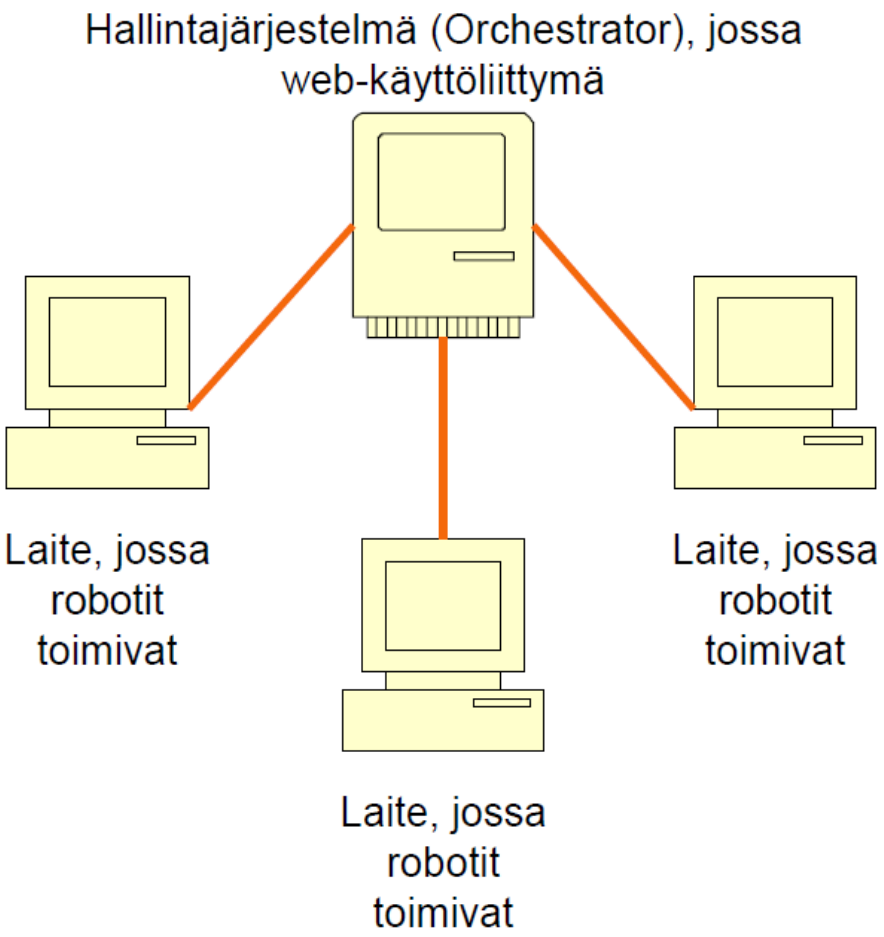
Nauhoitustilassa on mahdollista valita myös työkaluja työkaluvalikosta. Esimerkiksi jos halutaan kopioida jonkun kentän arvo, voidaan tämä vaihtoehto valita nauhoitustilan työkaluvalikosta.

Pelkkä nauhoitus ei useimmiten riitä tekemään toimivaa ohjelmistorobottia kuin erittäin yksinkertaisissa tapauksissa. Monimutkaisemmissa tapauksissa nauhoitusta muokataan UiPath Studiossa. Esimerkiksi poikkeuksienkäsittely tai muuttujien käyttäminen vaativat työskentelyä UiPath studiossa. Nauhoitus antaa lähinnä yksinkertaisen pohjan ohjelmistorobotin toiminnan rungoksi.

5.4.2 UiPath Orchestrator

UiPath Orchestrator on ohjelmistorobottien hallintajärjestelmä. Se toimii erillisellä palvelimella ja on käytettävissä web-käyttöliittymän välityksellä. Orchestratorin voi asentaa joko omalle palvelimelleen tai ostaa palveluna UiPathilta. UiPath Orchestrator sisältyy vain UiPathin maksulliseen versioon, joten sitä ei päästy testaamaan käytännössä tämän tutkielman puitteissa. UiPathin dokumentaatiosta saatavan tiedon perusteella voidaan kuitenkin kertoa sen yleisistä ominaisuuksista. (UiPath Orchestartor 2018)

UiPathin terminologia on hieman erilaista kuin tässä tutkielmassa olemme käyttäneet ja mitä esimerkiksi WorkFusion niillä ymmärtää. UiPath Orchestratorin dokumentaatio tarkoittaa prosesseilla yksittäistä ohjelmistorobottia, joka suorittaa jonkun tehtävän. Robotilla taas tarkoitetaan laitteessa toimivaa kehystä, jossa prosesseja eli tämän tutkielman terminologiassa ohjelmistorobotteja ajetaan.



Kuvio 9. UiPathissa hallintajärjestelmä toimii palvelimella ja on yhteydessä eri laitteissa oleviin robotteihin.

Orchestratorin ulkoasu on hyvin samanlainen kuin WorkFusion RPA Expressin Control Towerin. Orchestratorista löytyy Control Towerin tapaan työkalut robottien tilan seuraamiseen ja robottien hallintaan. Ohjelmistorobottien suorittamista pystyy esimerkiksi aikatauluttamaan. Orchestrator ei mahdollista kuitenkaan manuaalisten tehtävien luomista. Ohjelmistorobottien, eli UiPathin terminologiassa prosessien, ketjuttaminen toteutetaan jonojen (engl. queue) ja tapahtuminen (engl. transaction) avulla. Ohjelmistorobotti voi välittää dataa toisille ohjelmistoroboteilla näiden avulla. (UiPath Orchestartor 2018)

UiPath Orchestratorissa voidaan jakaa eri laitteessa toteutettuja ohjelmistorobotteja keskenään, jotta niitä voidaan suorittaa halutussa laitteessa. Pelkästään ihmisen apuna olevia ro-

botteja UiPath kutsuu läsnäoleviksi roboteiksi (engl. attended robots). Niitä ei voi ajaa Orchestratorin kautta, vaan laitteen käyttäjä ajaa niitä itse. Niiden tuottamia tuloksia ja tilaa voidaan kuitenkin seurata Orchestratorin kautta, ja ne voivat välittää Orchestratorin kautta dataa muille ohjelmistoroboteille. UiPath Orchestartorin avulla pystytään ajamaan niin kutsuttuja itsenäisiä robotteja (engl. unattended robots). Ne toimivat palvelimella virtuaaliympäristössä. (UiPath Orchestartor 2018)

6 Robotin suunnittelu ja toteutus

Tässä luvussa kuvaillaan ja taustoitetaan ongelma, jonka ratkaisemiseen toteutamme ohjelmistorobotin neljällä eri työkalulla. Robotin suunniteltu työnkulku kuvaillaan alaluvussa 6.2. Toteutuksen vaiheet eri työkaluilla kuvataan alaluvussa 6.3. Toteutuksien ohjelmakoodit löytyvät tutkielman liitteistä.

6.1 Ongelman kuvaus

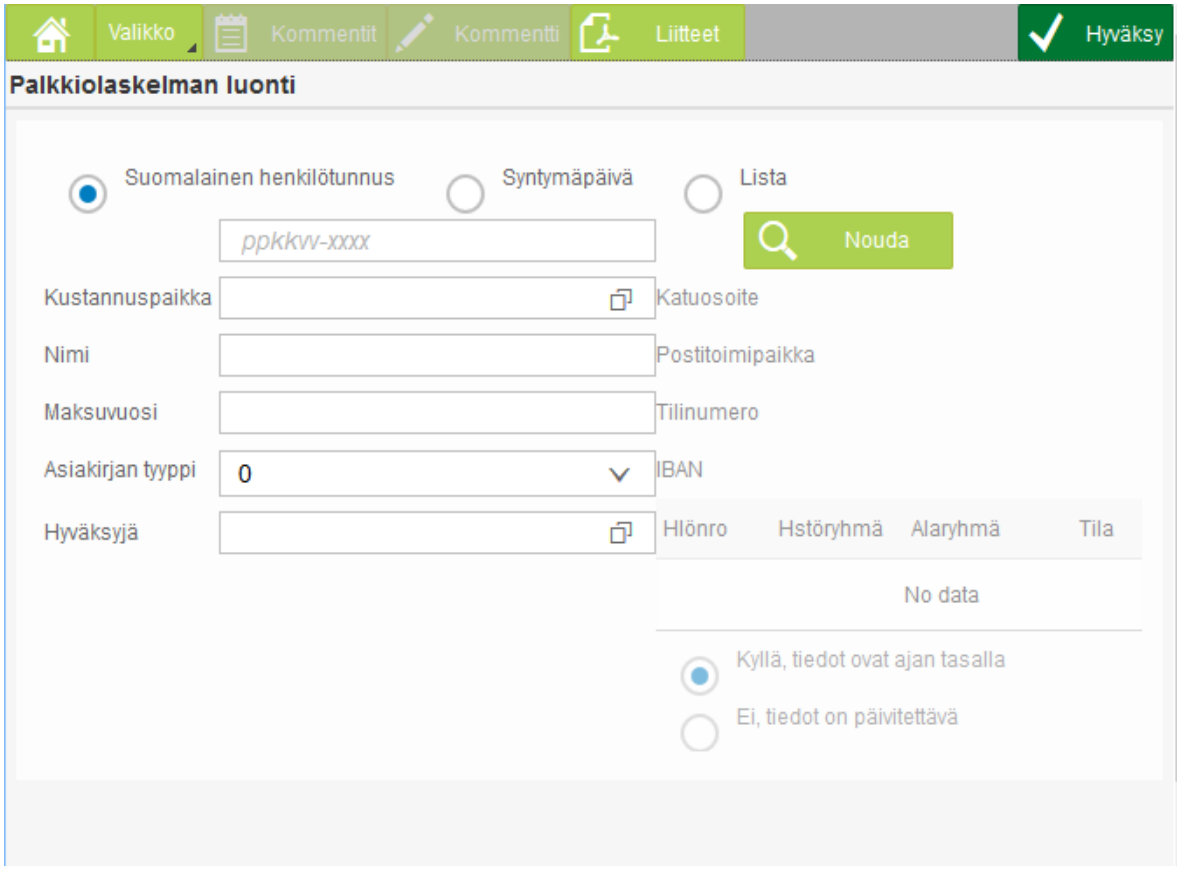
Tutkimuksen sovelluskohde on palkkioiden maksaminen Jyväskylän yliopistossa. Jyväskylän yliopisto hoitaa pienten ja kertaluontoisten tehtävien maksamisen palkkiolaskulomakkeella. Työn suorittanut henkilö toimittaa täytetyn ja käsin allekirjoitetun palkkiolaskulomakkeen Jyväskylän yliopiston palvelukeskukseen. Palvelukeskus tarkistaa, allekirjoittaa ja skannaa palkkiolomakkeen, jonka jälkeen se lähetetään web-käyttöliittymän läpi Certia Oy:lle, joka hoitaa maksamisen työntekijälle.

Yliopiston IT-palvelut toteuttaa omassa projektissaan palkkiolaskulomakkeen muuttamisen sähköisesti täytettäväksi, jotta paperilomakkeiden skannaamisesta päästään eroon. Sähköinen palkkiolaskulomake ei kuitenkaan poista rajapintaongelmaa, sillä palkkiolaskulomake on silti lähetettävä Certia Oy:lle web-käyttöliittymän avulla.

Tässä tutkimuksessa toteutetaan ohjelmistorobotti, joka automatisoi web-käyttöliittymän täyttämisen ja lomakkeen lähetyksen Certia Oy:lle. Ohjelmistorobotti saa tarvittavat tiedot web-käyttöliittymän täyttämiseen sähköiseltä palkkiolaskulomakkeelta.

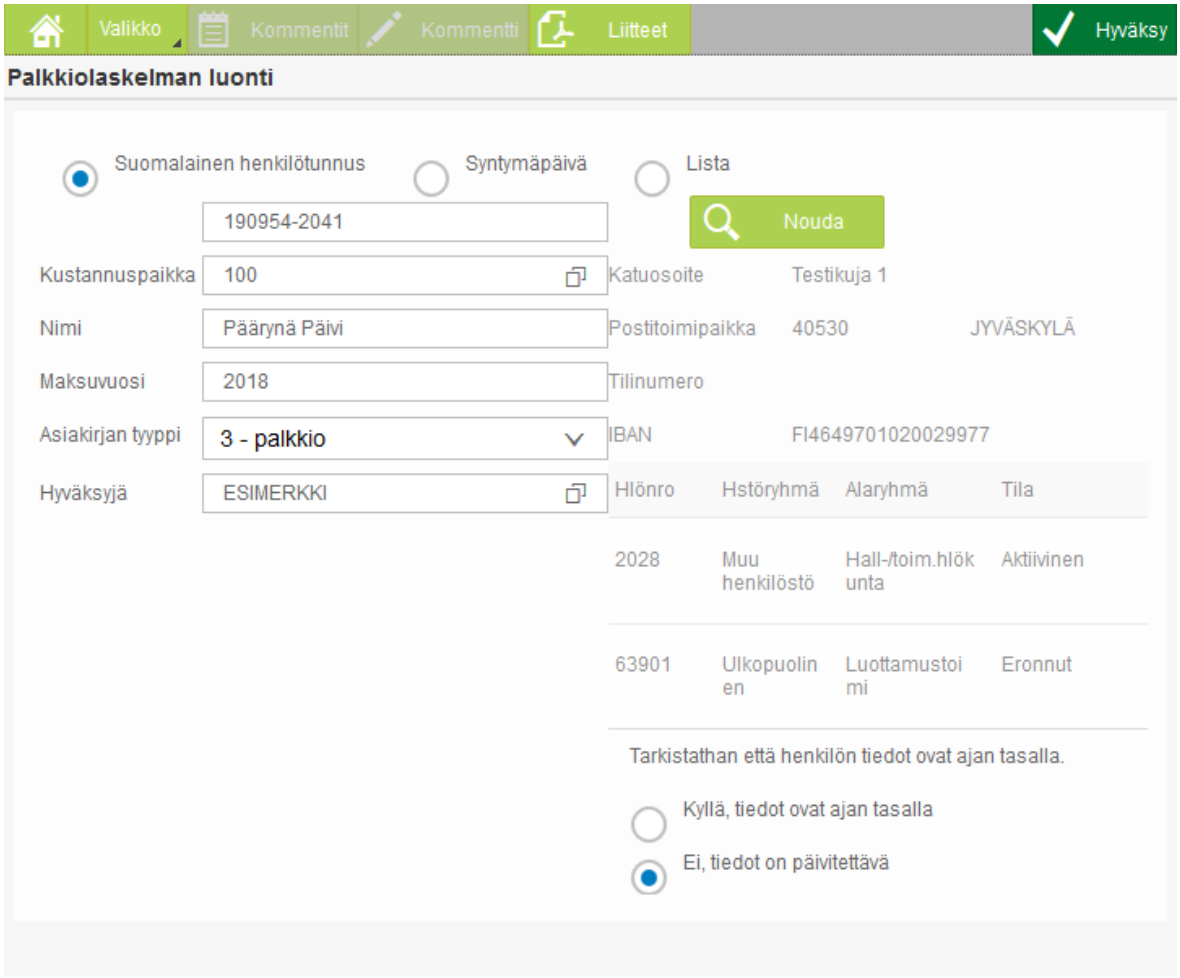
6.2 Lomakkeen täyttäminen

Kuviossa 10 on esitetty täytettävä web-lomake. Web-lomakkeen täytettävät tiedot saadaan sähköisestä palkkiolaskulomakkeesta. Web-lomakkeen liitteeksi tulee liittää palkkiolaskulomake PDF-muodossa.



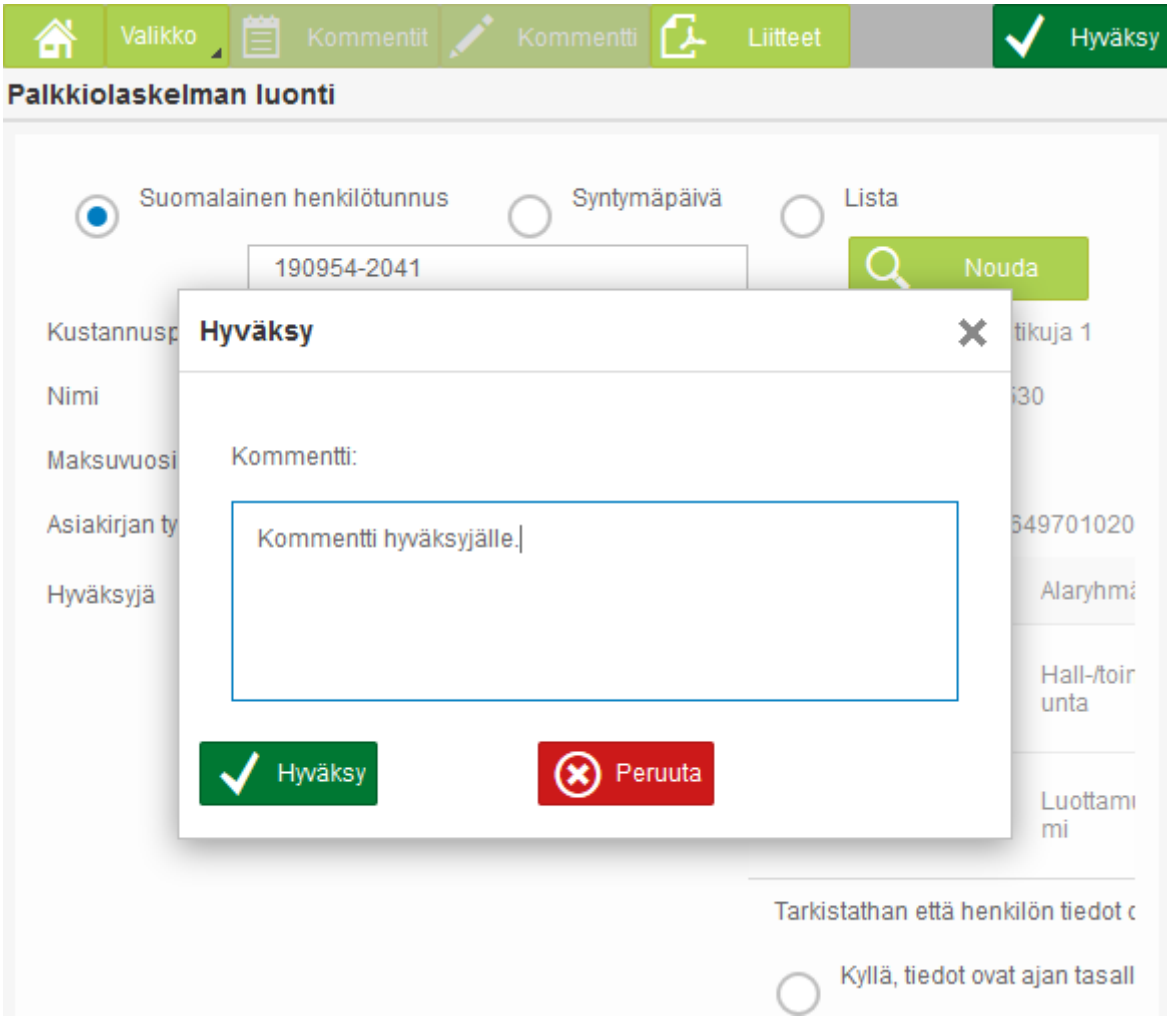
Kuvio 10. Lomake, joka robotin tulee täyttää.

Lomakkeelle täytetään ensin henkilötunnus. Henkilötunnuksen täyttämisen jälkeen painetaan *Nouda*-painiketta, joka noutaa henkilön nimen automaattisesti lomakkeelle. Henkilötiedot, jotka löytyvät Certian järjestelmästä, näkyvät painikkeen painamisen jälkeen lomakkeen oikealla sivulla. Lomakkeen täyttäjä tarkistaa tietojen oikeellisuuden ja valitsee henkilötietojen alta, ovatko tiedot ajan tasalla vai eivät. Tämän jälkeen lomakkeen täyttäjä täyttää kustannuspaikan, maksuvuoden, asiakirjatyypin ja hyväksyjän. Lopuksi web-lomakkeelle lisätään liitteeksi PDF-muotoinen palkkiolaskulomake kohdasta *Liitteet*. Kuviossa 11 on täytetty web-lomake.



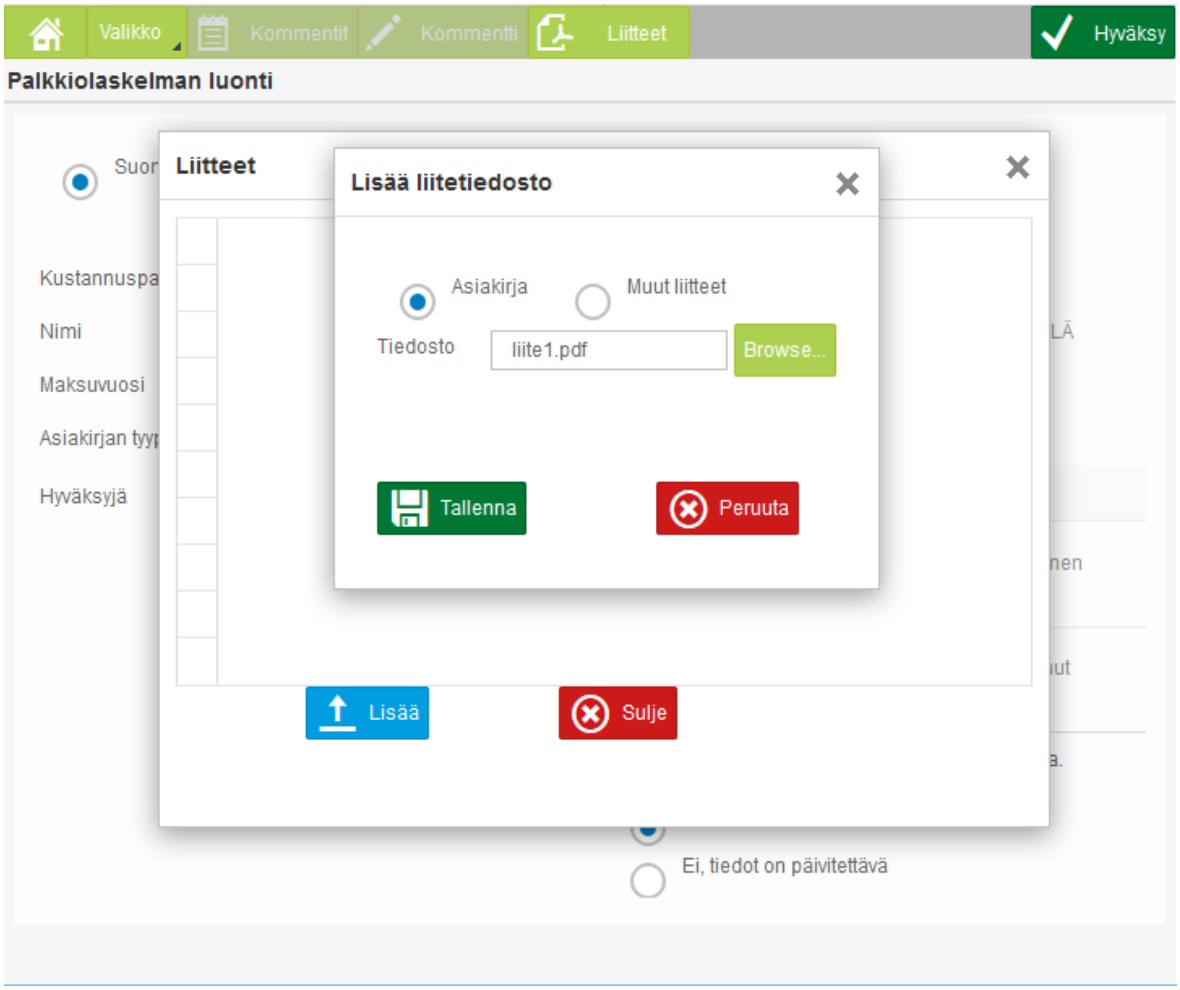
Kuvio 11. Täytetty web-lomake.

Tämän jälkeen käyttäjä hyväksyy lomakkeen *Hyväksy*-painikkeesta. *Hyväksy*-painike avaa vielä kommentti-ikkunan, jonne lomakkeen täyttäjä voi kirjoittaa kommentin hyväksyjälle (Kuvio 12). Kun kommentti-ikkunassa painetaan *Hyväksy*-painiketta, niin web-lomake lähetetään eteenpäin. Web-lomake estää lähetyksen ja antaa virheilmoituksen, jos täytetyissä tiedoissa on vikaa. Oikein täytetyn lomakkeen lähettämisen jälkeen web-lomake antaa onnistumisilmoituksen ja tyhjentää web-lomakkeen kentät uutta lomakkeen täyttöä varten.

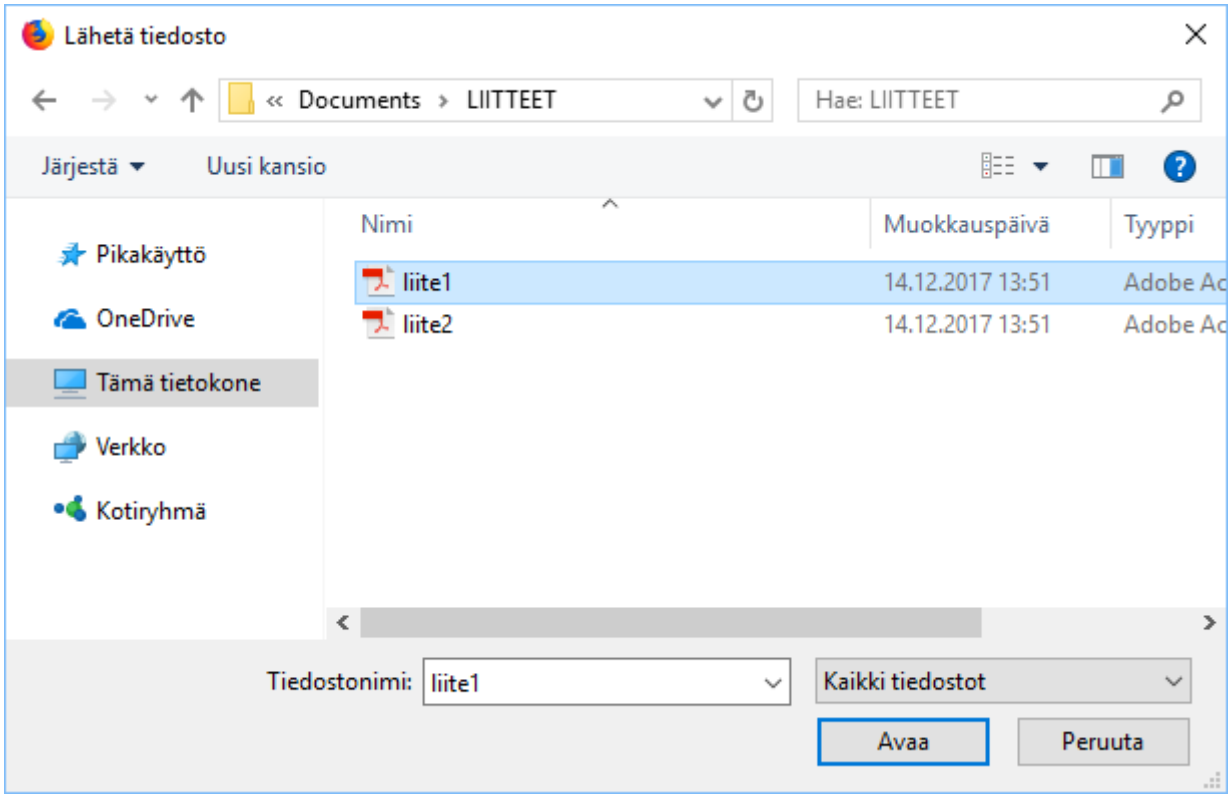


Kuvio 12. Hyväksymisvaiheen kommentti-ikkuna.

Alla on esitetty (Kuvio 13) PDF-muotoisen palkkiolaskulomakkeen lisääminen web-lomakkeen liitteeksi. Liitteen lisäys tapahtuu omassa ikkunassaan. Lisäksi liitteen hakeminen tiedoston-hallinnasta tapahtuu järjestelmän oman ikkunan välityksellä (Kuvio 14).



Kuvio 13. Liitteen lisääminen palkkiolaskulomakkeelle.



Kuvio 14. Liitteet lisätään valintaikkunan kautta, joka on järjestelmäriippuvainen.

6.3 Toteutus

Palkkiolaskulomakkeen täyttöön suunniteltu ohjelmistorobotti toteutettiin SikuliX:lla, Seleniumilla, RPA Expressillä ja UiPathilla. Toteutuksessa ohjelmistorobotti täytti lomakkeen testiarvoilla. Tuotantokäytön ohjelmistorobotit saisivat lomakkeelle täytettävät tiedot käynnistyksen yhteydessä argumenteissaan. Esimerkkitoteutus kertoo kuinka helppo ohjelmointirobotti oli toteuttaa, kuinka kauan toteutukseen meni, mitä haasteita toteutuksessa ilmeni, kuinka tarkka ohjelmistorobotista saatiin ja mikä oli kunkin ohjelmistorobotin suorituskyky ihmiseen verrattuna.

6.3.1 SikuliX-toteutus

SikuliX-toteutus tehtiin aluksi SikuliX IDE:n versiolla 1.1.1. SikuliX-toteutus perustui kuvatunnistukseen. SikuliX IDE:n kuvankaappaustyönkalu helpotti mallikuvien ottamista ja

IDE:ssä oli helppo testata robottia. Samoin IDE:ssä oli helppo säätää kuvatunnistuksen tarkkuuden asetuksia, joilla voidaan optimoida ohjelman suoritusnopeus. Kun pohja oli tehty SikuliX IDE:llä, niin sen pohjalta tehtiin Java-toteutus SikuliX API:n version 1.1.1 avulla. Tuotantoympäristössä Java-toteutus olisi parempi, koska ohjelmistorobotti voisi käyttää muita Java-kirjastoja ja Javalla tehty toteutus olisi käännettävissä suoritettavaksi *.jar*-tiedostoksi.

SikuliX IDE:ssä ei ole suoraa ominaisuutta, jolla toteutuksen voisi viedä suoraan Java-koodiksi. Tämä on huono puoli käytettävyyden kannalta. SikuliX IDE:ssä ohjelmistorobotti toteutettiin Python-koodina, joka on pienin muutoksin käännettävissä Javalle. Kuitenkin toteutuksen helppouden kannalta tämä on iso haaste. SikuliX-toteutus vaati väistämättä ohjelmointiosaamista.

SikuliX-toteutuksessa jouduttiin hieman rajoittamaan ohjelmistorobotin suoritusnopeutta laittamalla operaatioiden välille taukoja. Tämä johtui siitä, että selain lataa esimerkiksi uusia ilmestyvä elementtejä hetken verran, joten niiden ilmestymistä pitää odottaa.

SikuliX-toteutus käytti olemassa olevaa selainikkunaa toimintaansa. Niinpä ohjelmistorobotille ei tarvinnut antaa salasanoja, jotka olisi pitänyt kirjoittaa SikuliX:n ohjelmakoodiin. Käyttäjä kirjautuu ensin itse järjestelmään, jonka jälkeen ohjelmistorobotti suorittaa tehtävänsä järjestelmässä.

6.3.2 Selenium-toteutus

Selenium-toteutus nauhoitettiin ensin IDE:n versiolla 2.9.1 (liite 1). Selenium IDE:n versiolla 2.9.1 robotin suorituskaavion saa vietyä Ruby-, Python-, Java- tai C#-ohjelmakoodiksi. Ohjelmakoodiksi vieminen toimii suoraan tai pienin muokkauksin. Ohjelmakoodiksi vieminen mahdollistaa robotin paremman jatkokehityksen.

Aluksi Selenium toteutusta pidettiin mahdottomana, sillä liitetiedoston lisäämisen ei ajateltu toimivan ilman tiedostojenhallintaikkunan aukaisua. Selenium taas ei pysty käyttämään kuin web-sivuja, jolloin järjestelmän tiedostojenhallintaikkunan käyttö olisi mahdotonta. Weblomakkeen *Tiedosto*-nimiseen kenttään nimittäin ei pystynyt kirjoittamaan, vaan jos siihen kohdisti toimintaa, niin se avasi uuden ikkunan tiedostojenhallintaan. Lopulta Seleniumista löytyi *sendKeys* niminen metodi, joka mahdollisti tiedostopolun kirjoittamisen lomakekent-

tään. Näin Selenium toteutuskin saatiin lopulta onnistumaan.

Toteutuksessa piti ottaa huomioon, että lomakkeen kentän täyttämisen jälkeen ohjelmistorobotin täytyi simuloida ENTERin painaminen. Ilman tätä lomakkeen lähetys ei onnistunut, vaan lomake antoi virheilmoituksen täyttämättömistä kentistä. Ilmeisesti robotin täyttäessä kentät lomakkeen tarkistus ei huomioinut imitoituja näppäimen painalluksia ennen kuin painetaan ENTER.

Selenium IDE:llä toteutettu ohjelmistorobotti ei suoraan toimisi tuotantokäytössä, vaan se vaatii vielä muokkauksia ohjelmoimalla. Selenium IDE:llä toteutettu ohjelmistorobotille ei esimerkiksi pysty syöttämään argumentteja ulkopuolelta. Siksi ohjelmointitoteutus on välttämätön, jotta robotti saa tarvittavat tiedot lomakkeen täyttöön. Java-toteutuksen saa paketoitua myös *.jar*-muotoisiksi ajattaviksi tiedostoiksi. Kuitenkin Selenium IDE:n nauhoituksesta saa pohjan ohjelmointitoteutukselle.

Ohjelmointitoteutus tehtiin Javalla (liite 2). Selenium WebDriverin versio Java-toteutuksessa oli 3.11.0. Selenium IDE:stä saatua nauhoitusta jouduttiin vielä muokkaamaan jonkun verran. Ongelmia oli esimerkiksi liian nopeassa suoritusnopeudessa, jolloin tiettyjen painikkeiden painaminen ei rekisteröitynyt. Selenium IDE:ssä nauhoituksen nopeutta pystyi säätämään. Ohjelmointitoteutuksessa robotti toimi täydellä nopeudella, eikä säätömahdollisuutta ollut, joten tiettyjen komentojen väliin jouduttiin laittamaan tauko.

Suurin ongelma toteutuksessa oli, että Seleniumin WebDriverissa ei pysty käyttämään avoinna olevia ikkunoita. Aina kun robotti laitetaan päälle se avasi uuden selainikkunan. Selain avasi ikkunan myös uudella profiililla, jolloin kirjautumistiedot hävisivät. Lopulta robotti saatiin käyttämään selaimen oletusprofiilia, jolle oli tallennettu olemassa oleva kirjautumistiedot. Kuitenkin tuotantokäytössä tämä voisi olla ongelmallista, että robotti ei pysty käyttämään jo valmiiksi avointa selainikkunaa, vaan sen pitää avata uusi WebDriver-pohjainen selainikkuna toimintaansa varten. Myös tietoturvan kannalta voi olla ongelmallista, että kirjautumistiedot ovat tallennettu selaimelle, mistä ne voidaan saada helposti selville.

6.3.3 RPA Express -toteutus

RPA Express -toteutus tehtiin ohjelmiston versiolla 1.1.8. RPA Expressissä on tuki websivujen nopeammalle käytölle, koska se tunnistaa elementit html-rakenteen avulla. Tällaisessa toteutuksessa ongelmaksi ilmeni kuitenkin, ettei RPA Express pystynyt käyttämään avoinna olevaa selainta, vaan sen piti avata oma selaimensa. Tämä taas olisi vaatinut joka kerta uutta kirjautumista järjestelmään ja lisännyt tietoturvariskiä, sillä robotille olisi pitänyt antaa käyttäjätunnukset selkokielisenä. Toteutus päätettiin siksi tehdä kuvatunnistuksella, jolloin robotti pystyy käyttämään auki olevaa selainikkunaa, jossa käyttäjä oli valmiiksi kirjautunut järjestelmään.

RPA Express -toteutuksessa ensin suoritettiin nauhoitus, jonka jälkeen nauhoitusta muokattiin sopivaksi. Ohjelmistorobotti oli helppo toteuttaa. Ohjelmistorobotin käyttäminen asettaa kuitenkin haasteita. Jokaisen ohjelmistorobottia käyttävän pitää asentaa raskas RPA Express -ohjelmisto, jossa ohjelmistorobotteja voi ajaa. Robotin hallintajärjestelmäkin toimii vain yhdellä koneella. Jos tarkoituksena olisi tehdä useampia ohjelmistorobotteja käyttäjien tueksi, voisi RPA Express olla hyödyllinen. Kuitenkin vain yhden apurobotin tekeminen lomakkeen täyttöä varten vaikuttaa liioittelulta. RPA Express ohjelmistona syö ison osan käytettävän laitteen resursseista.

6.3.4 UiPath-toteutus

UiPath toteutus tehtiin UiPath 2017 Community Edition versiolla 2017.1.6522. Toteutus noudatti samoja periaatteita kuin RPA Express -toteutus eli ensin tehtiin nauhoitus lomakkeen täytöstä, jonka pohjalta ohjelmistorobotti rakennettiin. RPA Expressistä poiketen UiPath-toteutus käytti elementtien tunnistukseen html-rakennetta eikä kuvatunnistusta. Nauhoitustilassa html-elementin tunnistus onnistui hyvin. Myös liitetiedoston lisäämisessä pystyttiin käyttämään Windows-järjestelmien kahvoja, eikä kuvatunnistusta tarvittu. Tämä teki UiPath-toteutuksesta nopean.

UiPath toteutuksessa piti ottaa samalla tavalla ENTER-painalluksien simulointi huomioon kuin Selenium-toteutuksessa. Ilman ENTER-painallusten simulointia yhden kentän täyttämisen välissä täyttö ei onnistunut, vaan lomake antoi virheilmoituksen täyttämättömistä ken-

tistä.

UiPath toteutusta ajettaisiin ohjelmistorobottien ajamiseen tarkoitetun UiPath Robotin kautta. Tämä vaatisi ohjelmistorobotin käyttäjältä UiPath Robotin asennuksen omalle koneelleen. Tämä ei kuitenkaan ole yhtä raskas koneelle, kuin RPA Express, jossa pitää asentaa kehitysympäristö, tietokanta, OCR-moottori ja robottien ajamiseen tarvittava ohjelmisto samalla koneelle ennen robotin ajamista. Ohjelmistorobotti voidaan toki laittaa työskentelemään myös palvelimelle itsekseen. UiPath ohjelmiston hintaan vaikuttaa kuinka monessa koneessa on on käytössä kehitysympäristö, ohjelmistorobottien ajo-ohjelmisto sekä palvelimella pyörivien virtuaalirobottien määrä.

UiPath käytti toteutuksessa avoinna olevaa selainikkunaa, jossa käyttäjä oli kirjautunut sisälle järjestelmään. Näin UiPathille ei tarvinnut antaa järjestelmän salasanoja, mikä teki ratkaisusta tietoturvallisemman.

7 Arviointi

Tässä luvussa arvioidaan saatuja tuloksia. Alaluvussa 7.1 arvioidaan sitä kuinka hyvin tutkielman luvuissa 6.1 ja 6.2 esitelty tapaus soveltui ohjelmistorobotiikalla ratkaistavaksi. Tämän jälkeen arvioidaan robottien nopeutta, tarkkuutta ja niiden toteutuksen kestoa. Lopuksi arvioidaan kuinka alttiita ohjelmistorobotit ovat toiminnan häiriöille.

7.1 Tapauksen sopivuus ohjelmistorobotiikalla ratkaistavaksi

Luvussa 3 esiteltiin Fungin (2014) kriteerit tapaukselle, joka olisi järkevää ratkaista ohjelmistorobotiikalla. Näiden kriteerien ja toteutusvaiheessa saatujen kokemusten perusteella arvioidaan, kuinka hyvin luvussa 6 esitelty web-lomakkeen täyttäminen soveltui ohjelmistorobotiikalla ratkaistavaksi ongelmaksi.

Web-lomakkeen täyttäminen toistui samanlaisena ilman poikkeuksia eikä sen täyttäminen vaatinut ihmismäistä ajattelua (3.1.1) (3.1.6) (3.1.7). Lomaketta joudutaan täyttämään satoja kertoja vuodessa (3.1.2). Jyväskylän yliopiston palveluskeskus halusi automatisoida prosessin, koska se sitoo henkilöresursseja eli aiheuttaa tuntuvia kustannuksia (3.1.3) (3.1.8). Palkkiolaskuprosessissa sähköisen palkkiolaskulomakkeen tietoja pitää siirtää web-käyttöliittymän kautta eteenpäin, joten tapauksessa on kaksi yhteensopimatonta tietojärjestelmää (3.1.4). Web-lomake on vakaa ympäristö (3.1.5), mutta sen hallinta ei ole Jyväskylän yliopiston käsissä. Muutokset web-lomakkeessa aiheuttavat siis muutoksia ohjelmistorobotissa. Kuitenkin oletettavissa oli, ettei web-lomake tule muuttumaan hetkeen, joten ympäristön vakauden kriteerin voidaan olettaa täyttyvän.

Manuaalityön aiheuttamien virheiden määrästä ja kustannuksesta ei ole tietoa tai arviota. Kuitenkin muut luvussa 3 esitellyt kriteerit täyttyivät, joten näiden kriteerien perusteella tapaus oli sopiva ohjelmistorobotiikalla ratkaistavaksi.

Toteutuksessa ei ollut myöskään pääsyä Certia Oy:n järjestelmiin, joten omaa rajapintaa järjestelmien välille ei voitu toteuttaa lähdekoodia muokkaamalla, vaan sellainen olisi pitänyt ostaa Certialta. Tämän vuoksi ohjelmistorobotiikka oli tässä tapauksessa perinteistä auto-

maatiota kannattavampi ratkaisu, koska se voitiin itse toteuttaa ilman järjestelmien lähdekoodiin tehtäviä muutoksia.

Ohjelmistorobotti ei kuitenkaan olisi toiminut ilman yliopiston IT-palvelujen toteuttamaa sähköistä palkkionlaskulomaketta, josta se sai tiedot web-lomakkeen täyttöön. Jos web-lomakkeelle tulevat tiedot olisi pitänyt lukea skannatusta palkkiolaskulomakkeesta, olisi se ollut ohjelmistorobotille hyvin vaikea tai jopa mahdoton tehtävä. Ohjelmistorobotti ei pysty käsittelemään tietoa, joka ei ole säännönmukaista. Sähköinen palkkiolaskulomake mahdollisti sen, ettei ohjelmistorobotin tarvinnut käyttää tekstintunnistusominaisuuksia skannatun palkkiolaskulomakkeen lukemiseen. Ohjelmistorobotti sai tiedot sähköisesti ja sen päätehtäväksi jäi täyttää web-lomake. Jokaisella ohjelmistorobotiikan työkalulla pystyttiinkin toteuttamaan ohjelmistorobotti, joka pystyi täyttämään web-lomakkeen ilman virheitä manuaalista työtä nopeammassa ajassa.

7.2 Toteutusten nopeus verrattuna käsityöhön

Kaikilla työkaluilla tehdyt ohjelmistorobotit vähintään puolittavat ajan, joka yhden lomakkeen täyttämiseen menisi käsityönä (Taulukko 3). SikuliX- ja RPA Express -toteutukset toimivat kuvatunnistuksella ja olivat siksi selvästi hitaampia kuin UiPath- ja Selenium-toteutukset, jotka pystyivät käyttämään web-sivun elementtejä analysoimalla web-sivun rakennetta.

Taulukko 3: Lomakkeen täyttönopeus.

Työkalu	Aika per täytetty lomake
SikuliX	26s
Selenium	7s
RPA Express	33s
UiPath	12s
Manuaalisesti	68s

Yhden lomakkeen täyttöaika laskettiin sivun latauksen aloittamisesta seuraavan sivun lataamisen alkuun asti. Suurin osa työkaluista käytti olemassa olevaa selainikkunaa, jossa oltiin

valmiina kirjauduttu sisään järjestelmään. Selenium-toteutuksessa jouduttiin kuitenkin ensin kirjaamaan käyttäjä sisään, ennen kuin lomaketta alettiin täyttää, sillä Selenium ei pystynyt käyttämään avointa selainikkunaa, vaan sen piti avata uusi selainikkuna. Kirjautuminen kesti Seleniumilla 7 sekuntia. Kuitenkin itse lomakkeen täyttö vei sillä vain 7 sekuntia. Kirjautumisaika on kuitenkin merkityksetön tuloksia ajatellen. Jos robotin esimerkiksi pitäisi täyttää 100 lomaketta, niin robotti kirjautuisi sisään, mikä veisi 7 sekuntia, ja täyttäisi sitten 100 lomaketta, joihin kuluisi 700 sekuntia. Yhteensä tähän kuluisi siis 707 sekuntia.

Tuloksista voidaan päätellä, että ohjelmistorobotti nopeuttaa työntekoa suurilla lomakemäärillä. Mikäli ohjelmistorobotti saa ennen käynnistystä usean lomakkeen tiedot, eikä sitä täydy erikseen käynnistää jokaisen lomakkeen täyttöön, niin laskennallinen aika lomakkeiden täyttämiseen on lomakkeiden määrä kertaa yhden lomakkeen täyttämiseen kulunut aika. Esimerkiksi 100 lomakkeen täyttämiseen kuluisi aikaa käsityönä laskennallisesti noin 113 minuuttia eli lähes kaksi tuntia. Hitaimmallakin ohjelmistorobotilla (RPA Express) päästäisiin alle tuntiin (noin 55 minuuttia). Nopeimmalla Selenium-toteutuksella aikaa kuluisi noin 12 minuuttia eli ohjelmistorobotilla menisi vain hieman yli kymmenesosa siitä ajasta, mitä manuaaliseen lomakkeen täyttöön menisi.

Laskennallisessa arviossa ei otettu huomioon lomaketietojen tarkistusta, vaan pelkästään web-lomakkeen täyttämiseen kulunut aika. Lomakkeiden tiedot pitää tarkastaa palvelukeskuksen vastuuhenkilön toimesta ennen niiden lähettämistä Certia Oy:lle. Tämä pitää tehdä joka lomakkeelle riippumatta siitä, täytetäänkö web-lomake manuaalisesti vai automaattisesti. Manuaalisessa työnkulussa vastuuhenkilö tarkastaa lomakkeen, jonka jälkeen hän täyttää web-lomakkeen käsin. Ohjelmistorobotiikan ratkaisussa vastuuhenkilö tarkastaa tiedot, jonka jälkeen sähköinen lomake menee hyväksyttyjen kansioon, josta ohjelmistorobotti saa tiedot. Vastuuhenkilö voi siis hyväksyä useamman palkkionlaskun kerralla, jonka jälkeen ohjelmistorobotti täyttää usean web-lomakkeen kerralla.

Aikaa säästetään, jos ohjelmistorobottia ei tarvitse valvoa koko sen suorituksen ajan. Työntekijöiden resursseja vapautuu muuhun käyttöön, kun heidän ei tarvitse edes seurata web-lomakkeen täyttämistä. Tällöin säästetty aika on manuaaliseen lomakkeen täyttämiseen aiemmin kulunut aika.

7.3 Toteutusten tarkkuus verrattuna käsityöhön

Kaikilla työkaluilla pystyttiin luomaan ohjelmistorobotti, joka täytti kentät luotettavasti testiarvoilla. Ohjelmistorobottien tarkkuutta on kuitenkin hankala arvioida ilman suurempaa koetta oikealla datalla ja tuotantokäytössä. Myös käsityön tarkkuuden arviointi on hankalaa ilman järjestettyä koetta. Voisi kuitenkin olettaa, etteivät ohjelmistorobotit tee tarkkuutensa puolesta virheitä, kun taas ihmiset tekevät niitä huolimattomuuttaan (Alégroth, Feldt ja Ryrholm 2015) (Fung 2014). Tämä on kuitenkin yksi jatkotutkimukseen sopiva tutkimusaihe.

7.4 Ohjelmistorobotin toteutuksen kesto

Ohjelmistorobottien toteutuksen kestoon kulunutta aikaa ei mitattu tarkasti tässä tutkielmassa. Joitain arvioita voidaan kuitenkin antaa. Työkaluihin tutustumisen jälkeen toteutukset olivat UiPathilla ja RPA Expressilla todella nopeita. Molemmissa meni alle yksi työpäivä kussakin. SikuliX- ja Selenium-toteutuksissa meni noin kaksi työpäivää per toteutus. Osaltaan tätä selittää, että ohjelmistorobotteja jouduttiin siirtämään Java-ohjelmakoodiksi, kun ne oli ensin tehty Selenium IDE:llä ja SikuliX IDE:llä. Seleniumilla iso ajasta meni kamppaillessa WebDriverin ja kirjautumisen kanssa. SikuliX-toteutus oli taas ensimmäinen kaikista toteutuksista, joten ohjelmistorobotin toteutukseen kului hieman pidempi aika.

Voidaan sanoa yleistetysti, että itse ohjelmistorobotin toteutus on todella nopeaa, kun työvälineet ovat hallussa. Suurin osa ajasta kuluu ennemmin vaatimusmäärittelyn tekemiseen, missä pitää hahmotella vaiheet, mitä robotin tulee tehdä ja millä tavalla sen halutaan toimivan. Siinä mielessä voidaan yhtyä Asatianin tutkijaryhmän arvioon siitä, että ohjelmistorobotin toteutukseen kuluu yleensä noin 2-4 viikkoa, kun kierretään tämän tyylistä rajapintaongelmaa (Asatiani ja Penttinen 2016). Tämän tutkimuksen perusteella voidaan sanoa, että suurin osa tästä ajasta kuluu kuitenkin itse ongelman hahmottamiseen ja vaatimusmäärittelyyn, eikä niinkään itse ohjelmistorobotin tekniseen toteutukseen.

Ohjelmistorobotin toteutuksen kesto sopii hyvin jatkotutkimusaiheeksi. Robotin toteutuksesta voitaisiin järjestää todellinen koeasetelma, jotta saataisiin tarkkoja mittaustuloksia vain näiden hieman epämääräisten arvioiden lisäksi. Samoin vaatimusmäärittelyn kestoa olisi hyvä arvioida täsmällisemmin, sillä siihen kuluu tämän tutkielman perusteella suurempi aika

kuin itse toteutukseen.

7.5 Toteutusten tietoturva

Jokaisessa ohjelmistorobotin toteutuksessa jouduttiin ottamaan huomioon tietoturva, koska järjestelmään jouduttiin kirjautumaan sisään, ennen kuin sitä voitiin käyttää. Millekään ohjelmistorobotille ei annettu suoraan selkokielisenä tunnuksia järjestelmään. SikuliX-, RPA Express- ja UiPath-toteutuksissa käyttäjä kirjautui ensin selaimella sisään, ennen kuin käynnisti ohjelmistorobotit. Ohjelmistorobotit käyttivät avoinna olevaa selainikkunaa, joten niiden ei tarvinnut tietää mitään salasanoista.

Selenium-toteutuksessa oli haasteensa, sillä se ei voinut käyttää avointa selainikkunaa, vaan sen piti käynnistyksen yhteydessä avata omansa. Selenium-toteutuksessakaan salasanaa ei kuitenkaan annettu selkokielisenä ohjelmistorobotille, vaan se tallennettiin selaimen profiiliin, jolloin Selenium onnistui kirjautumaan järjestelmään ilman tietoa salasanasta. Tämä ei välttämättä ole kuitenkaan sen tietoturvallisempi ratkaisu, sillä salasana on tiedossa selaimen muistissa. Niinpä Seleniumin käyttöä tulee harkita tarkkaan, kun mietitään automatisoinnin tietoturvallisuutta.

7.6 Toiminnan häiriöt ja niiden korjaaminen

Tässä tutkielmassa haluttiin tutkia myös ohjelmistorobottien korjaamiseen menevää aikaa, kun käyttäjärajapinta muuttuu. Penttinen ym. (2018) totesivat artikkelissaan, että ohjelmistorobotiikka on herkkä käyttäjärajapinnan muutoksille, mutta taustajärjestelmien muutokset eivät siihen juuri vaikuta. (Penttinen, Kasslin ja Asatiani 2018)

Lomakkeen selitekenttä *“Asiakirjan tyyppi”* muutettiin muotoon *“Tyyppi”*. Ohjelmistorobotin toiminta häiriintyi tästä ja se ilmoitti virheestä. Korjauksen tekeminen tälle tapaukselle kesti SikuliX:lla 2 minuuttia. Kyseinen selitekenttä esiintyi SikuliX-koodissa yhdessä kohtaa, joten sen korjaaminen oli suoraviivaista. Virheilmoitus antoi suoraan rivin, missä virhe tapahtui. SikuliX:n työkaluilla otettiin uusi kuvankaappaus selitekentästä ja lisättiin sen osoite koodiin vanhan selitekentän kuvan osoitteen paikalle.

Yksittäisen elementin ulkoasun muuttumisesta aiheutuva korjaustoimenpide on siis melko suoraviivainen operaatio kuvatunnistukseen perustuvissa roboteissa, sillä mallikuva vaihdetaan vain uuteen. Koko ulkoasun muuttuminen aiheuttaa jo enemmän vaivaa. Kuitenkaan, jos ohjelman suorituslogiikkaa ei tarvitse muuttaa, ulkoasun muutos tarkoittaa vain uusien mallikuvien ottamista. Vasta jos työtehtävän suoritustapa muuttuu, joudutaan ohjelmistorobotti suunnittelemaan alusta lähtien uudelleen.

Virheen havaitsemiseen menevä aika on kuitenkin tässä tapauksessa suurempi, kuin korjaukseen menevä aika. Siksi ohjelmistorobottien tilaa pitää pystyä valvomaan säännöllisesti. Erityisesti ongelma koskee robotteja, jotka voidaan laittaa työskentelemään pitkiksi ajoiksi itsenäisesti. Kaupallisissa työkaluissa hallintajärjestelmät auttavat havaitsemaan häiriöt ohjelmistorobottien toiminnassa nopeasti. Esimerkiksi UiPathin kaltaisessa kaupallisessa ohjelmistossa on mahdollisuus sähköpostihälytyksiin (UiPath Orchestartor 2018). Jos ohjelmistorobotin toiminta häiriintyy jostain syystä, niin UiPath osaa lähettää automaattisen hälytyksen asiasta sähköpostilla. Avoimilla ohjelmistorobotiikan työkaluilla (SikuliX ja Selenium) toteutetuissa ohjelmistoroboteissa ei ole valmista hallintajärjestelmää ohjelmistoroboteille, joten virhetilanteista ilmoittamiseen pitää kiinnittää erityistä huomiota, kun näillä työkaluilla toteutetaan ohjelmistorobotteja.

8 Johtopäätökset

Tässä Pro Gradu -tutkielmassa käytiin läpi, mitä on ohjelmistorobotiikka ja millaisia ongelmia sillä voidaan ratkaista. Tutkielmassa kartoitettiin sopivia avoimia ja kaupallisia ohjelmistorobotiikan työkaluja. Käytännön osassa toteutettiin ohjelmistorobotti neljällä eri työkalulla todelliseen web-lomakkeiden täyttöön liittyvään ongelmaan. Tarkoituksena oli tuottaa tietämystä ohjelmistorobottien suunnitteluun ja toteutukseen.

Ohjelmistorobotiikan työkaluja kartoittaessa kävi ilmi, että suurin osa työkaluista on kaupallisia. Tarjolla oli muutama avoin ohjelmistorobotiikan työkalu, joista tutkielmassa testattiin SikuliX:a ja Seleniumia. Kaupallisista työkaluista testattiin UiPathia ja RPA Expressiä, joista oli tarjolla ilmaiset versiot kokeiluun. Avoimien työkalujen keskeinen ero kaupallisiin oli, että niistä puuttui hallintajärjestelmä usean ohjelmistorobotin hallitsemiseen.

Kaikilla neljällä työkalulla saatiin aikaan toimiva ohjelmistorobotti. Ohjelmistorobotit toimivat myös manuaalista työtä nopeammin. Nopeimmalla Selenium-toteutuksella kului vain hieman yli kymmenesosa manuaalityöhön kuluneesta ajasta. Hitain RPA Express -toteutus puolitti manuaalityöhön kuluneen ajan. Ohjelmistorobotit voidaan myös jättää työskentelemään itsekseen, jolloin manuaalityöhön kulunut aika vapautuu ihmiseltä kokonaan muuhun käyttöön.

Ohjelmistorobotit eivät kuitenkaan korvanneet ihmistä täysin. Ihmisiä tarvitaan edelleen esimerkiksi tietojen oikeellisuuden tarkistamiseen. Samoin ohjelmistorobotin toiminnan häiriintyessä ihmistä tarvitaan korjaamaan virhetilanne. Ohjelmistorobotit toimivat kuitenkin hyvin ihmisten avustajina automatisoiden rutiinityötä.

Luvussa 6 esitelty ongelma täytti luvussa 3 annetut kriteerit automatisoitavalle prosessille. Siksi sen automatisointi onnistui hyvin, mutta tapaus saattaa antaa liian optimistisen kuvan ohjelmistorobotiikan mahdollisuuksista. Automatisoitavien tapausten löytäminen voi olla haastavaa, sillä kriteereitä on paljon. Ei riitä, että työn tekeminen manuaalisesti on kallista, vaan työtehtävän tulee noudattaa selvää säännönmukaisuutta. Ohjelmistorobotti pystyy toimimaan vain hyvin säännönmukaisessa työssä, jossa tieto on saatavissa sähköisesti ja mielellään tekstimuotoisena. Ihmismäiseen soveltamiseen ohjelmistorobotit eivät kykene. Auto-

maatioon ladataan usein kovia odotuksia, mutta kuitenkin sopivia tapauksia sen hyödyntämiseen on rajallinen määrä. Oikeanlaisten ongelmien löytäminen voi olla ohjelmistorobotiikan hyödyntämisen suurin haaste.

Tutkimuksessa havaittiin, että suurin aika ohjelmistorobottien toteutuksessa menee vaatimusmäärittelyn tekemiseen. Itse ohjelmistorobottien toteutukseen kulunut aika on hyvin lyhyt siihen verrattuna. Tässä tutkielmassa ei kuitenkaan mitattu tarkasti kokeellisesti, kuinka pitkä aika ohjelmistorobottien toteutukseen kuluu, joten tarkempi mittaaminen on hyvä jatkotutkimuksen aihe.

Tutkielman aikana ilmeni myös, että lisää tutkimusta tarvittaisiin ohjelmistorobottien tarkkuudesta verrattuna käsityöhön. Sitä olisi hyödyllistä testata kokeellisesti suurilla datamäärillä.

Lähteet

Alégroth, Emil, Robert Feldt ja Lisa Ryrholm. 2015. “Visual gui testing in practice: challenges, problemsand limitations”. *Empirical Software Engineering* 20 (3): 694–744.

Altaf, Insha, Jawad Ahmad Dar, Firdous ul Rashid ja Mohd Rafiq. 2015. “Survey on selenium tool in software testing”: 1378–1383.

Apache Maven. 2018. “Apache Maven Project”. Viitattu 26. tammikuuta 2018. `https://maven.apache.org/`.

Asatiani, Aleksandre, ja Esko Penttinen. 2016. “Turning robotic process automation into commercial success: Case OpusCapita”. *Journal of Information Technology Teaching Cases* 6 (2): 67–74. doi:`10.1057/jittc.2016.5`.

Chelliah, John. 2017. “Will artificial intelligence usurp white collar jobs?” *Human Resource Management International Digest* 25 (3): 1–3.

Frey, Carl Benedikt, ja Michael A. Osborne. 2017. “The future of employment: How susceptible are jobs to computerisation?” *Technological Forecasting and Social Change* 114:254–280. doi:`https://doi.org/10.1016/j.techfore.2016.08.019`. `http://www.sciencedirect.com/science/article/pii/S0040162516302244`.

Fung, H. P. 2014. “Criteria, Use Cases and Effects of Information Technology Process Automation (ITPA)”. *Advances in Robotic and Automation* 3 (3): 1–11.

Herbert, Ian. 2016. “The future of professional work: Will you be replaced or will you be sitting next to a robot?” *Management Services* 60 (2): 22–27.

IEEE. 2017. “IEEE Guide for Terms and Concepts in Intelligent Process Automation”. *IEEE Std 2755-2017:* 1–16. doi:`10.1109/IEEESTD.2017.8070671`.

Järvinen, Pertti. 2004. *Tutkimustyön metodeista.* [Uud. p.] Toimittanut Annikki Järvinen. Tampere: Opinpajan kirja.

Lacity, Mary, ja Leslie Willcocks. 2016. “Robotic Process Automation at Telefónica O2”. *MIS Quarterly Executive* 15 (1): 21–35.

Le Clair, Craig. 2017. “The Forrester Wave: Robotic Process Automation, Q1 2017”.

Menard, Bruno. 2008. “Optical Character Recognition” [kielellä English]. *Quality* 47, numero 5 (toukokuu): 3–18S, 19S, 20S. https://search-proquest-com.ezproxy.jyu.fi/docview/235228621?accountid=11774.

Penttinen, Esko, Henje Kasslin ja Aleksandre Asatiani. 2018. “How to Choose between Robotic Process Automation and Back-End System Automation?” (Kesäkuu).

RPA Express. 2018. “RPA Express User Guide”. Viitattu 15. toukokuuta 2018. https://kb.workfusion.com/display/RPAe/.

Selenium. 2018. “Selenium Documentation”. Viitattu 15. toukokuuta 2018. https://www.seleniumhq.org/docs/.

SikuliX. 2017. “Sikuli / SikuliX Documentation for version 1.1+ (2014 and later)”. Viitattu 9. tammikuuta 2018. http://sikulix-2014.readthedocs.io/en/latest/index.html.

Sikulix API. 2018. “Javadoc of SikuliX”. Viitattu 29. toukokuuta 2018. http://doc.sikuli.org/javadoc/index.html.

UiPath. 2018. “Community Edition License Agreement”. Viitattu 5. syyskuuta 2018. https://www.uipath.com/community-license-agreement.

UiPath Orchestartor. 2018. “The UiPath Orchestrator Guide”. Viitattu 16. toukokuuta 2018. https://orchestrator.uipath.com/.

W3C. 2017. “XML Path Language (XPath) 3.1”. Viitattu 5. syyskuuta 2018. https://www.w3.org/TR/2017/REC-xpath-31-20170321/.

Wil M P van der, Aalst, Martin Bichler ja Armin Heinzl. 2018. “Robotic Process Automation” [kielellä English]. Copyright - Business Information Systems Engineering is a copyright of Springer, (2018). All Rights Reserved; Last updated - 2018-07-11, *Business Information Systems Engineering* 60, numero 4 (elokuu): 269–272. https://search-proquest-com.ezproxy.jyu.fi/docview/2038480309?accountid=11774.

Workfusion. 2018. “WorkFusion RPA Express Frequently Asked Questions”. Viitattu 1. maaliskuuta 2018. https://www.workfusion.com/rpaexpress-faq.

Liitteet

Liite 1

SikuliX Java-toteutus

```
package robotti;

import org.sikuli.script.*;

/**
 * Ohjelmistorobotti palkkiolaskun käsittelyyn.
 * Käyttää SikuliX:a hyödykseen(SikuliXAPI-kirjasto).
 * @author Tuukka Jurvakainen
 *
 */
public class App implements Runnable
{
    private String[][] tiedot;

    /**
     * Luokan konstruktori. Käynnistyessä robotti saa täytettävät tiedot
     * kaksi ulotteisessa taulukossa.
     */
    public App(String[][] taytettavatTiedot){
        this.tiedot = taytettavatTiedot;
    }


    /**
     * Web-lomakkeen täyttö.
     * Robotin logiikka on tässä aliohjelmassa.
     * Heittää FindFailedin, jos kuvaa ei löydy näytöltä.
     */
    public void teeProsessi() throws FindFailed{
        Screen s = new Screen();
```

```
for(int i = 0; i < tiedot.length; i++){
    //Hetun perusteella kentän täyttö:
    s.click("imgs/1513606222592.png");
    s.wait("imgs/1513251749214.png",10);
    s.wait(0.5);//lisäodotus
    s.click("imgs/1513251749214.png");
    s.type(this.tiedot[i][0]);
    s.click("imgs/1513251821990.png");
    s.wait("imgs/1515583401120.png",5);
    s.wait(0.5);//lisäodotus
    s.click("imgs/1515583401120.png");

    //Muiden kenttien täyttö:
    s.click(new Pattern("imgs/1515584096760.png").targetOffset(100,0));
    s.type(this.tiedot[i][1]);
    s.click(new Pattern("imgs/1515584403995.png").targetOffset(100,0));
    s.type(this.tiedot[i][2]);
    s.click("imgs/1515584487497.png");
    s.click(new Pattern("imgs/1515584487497.png").targetOffset(100,0));
    s.type(this.tiedot[i][3]);
    s.click("imgs/1515584536711.png");
    s.click(new Pattern("imgs/1515584536711.png").targetOffset(100,0));
    s.type(this.tiedot[i][4]);

    //Liitteen lisääminen:
    s.click("imgs/1513252009949.png");
    s.wait("imgs/1513252045542.png");
    s.click("imgs/1513252045542.png");
    s.wait("imgs/1513252089645.png");
    s.click("imgs/1513252105064.png");
    s.wait("imgs/1515587979245.png", 5);
    s.click(new Pattern("imgs/1515587979245.png").targetOffset(79,0));
    s.paste(this.tiedot[i][5]);
    s.click("imgs/1515588027151.png");
    s.wait("imgs/1513253148449.png",5);
    s.click("imgs/1513252089645.png");
    s.wait("imgs/1513252045542.png",5);
```

```
            s.click("imgs/1513252787249.png");
            s.click("imgs/1513252808046.png");
            s.wait("imgs/1513253270569.png",5);
            Region region = s.find("imgs/1513265370885.png").below(300);
            Match target = region.find("imgs/1513252808046.png");
            s.click(target);
            //odotetaan merkkiä että asiakirja lähetettiin:
            s.wait("imgs/1513251749214.png",10);


        }

    }

    /**
     * Toteutetaan Runnable-rajapintaa. Siksi run-funktio-joka ajaa säikeen.
     */
    public void run() {
        try {
            this.teeProsessi();
        } catch (FindFailed e) {
            // virhetilanteessa antaa virheilmoituksen
            e.printStackTrace();
        }

    }

}
```

Liite 2

Selenium Java-toteutus

```
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.chrome.ChromeDriver;
```

```
import org.openqa.selenium.chrome.ChromeOptions;

public class robotti {
    public static void main(String[] args){

        // Haetaan profiilikansiosta profiili, josta saadaan kirjautumistiedot,
        // jotta ei tarvitse täyttää kirjautumistietoja joka kerta uudestaan:
        System.setProperty("webdriver.chrome.driver",
        "C:/Users/Tuukka/Downloads/chromedriver.exe");
        ChromeOptions options = new ChromeOptions();
        options.addArguments("user-data-dir=
        C:\\Users\\Tuukka\\AppData\\Local\\Google\\Chrome\\User Data");
        options.addArguments("--start-maximized");
        ChromeDriver driver = new ChromeDriver(options);

        //Kirjautuminen:
        driver.get("https://qaswebdp1.certia.fi:8193/irj");
        driver.findElement(By.name("uidPasswordLogon")).click();

        //Avataan oikea sivu:
        String baseUrl = "https://qaswebdp1.certia.fi:8187";
        driver.get(baseUrl + "/neptune/bilot_smartinvoice_prew_create.html");

        //Täytetään lomake:
        try {
            driver.findElement(By.id("inFormHeaderPERS_ID-inner"))
            .sendKeys("190954-2041" + Keys.ENTER);
            driver.findElement(By.id("btnGetHr")).click();
            Thread.sleep(500);
            driver.findElement(By.id("rgUpdateNeeded-0-Button")).click();
            driver.findElement(By.id("inFormHeaderCOST_CENTER-inner"))
            .sendKeys("2110002" + Keys.ENTER);
            driver.findElement(By.id("inFormHeaderPAYMENT_YEAR-inner"))
            .sendKeys("2018" + Keys.ENTER);
            driver.findElement(By.id("inFormHeaderDOCUMENT_TYPE-label"))
            .click();
            driver.findElement(By.id("__item3")).click();
```

```
            driver.findElement(By.id("inFormHeaderAPPROVER-inner"))
            .sendKeys("21TUNNUS3" + Keys.ENTER);
            driver.findElement(By.id("btnAttachments")).click();
            driver.findElement(By.id("btnUpload")).click();
            driver.findElement(By.id("fileUpload-fu")).clear();
            driver.findElement(By.id("fileUpload-fu"))
            .sendKeys("C:\\Users\\Tuukka\\Documents\\LIITTEET\\liite2.pdf");
            driver.findElement(By.id("btnUp1")).click();
            Thread.sleep(1000);
            driver.findElement(By.id("btnCancelAtt1")).click();
            Thread.sleep(500);
            driver.findElement(By.id("btnAccept")).click();
            Thread.sleep(500);
            driver.findElement(By.id("btnAcp2")).click();
            System.out.println("Tietojen täyttäminen onnistui");
        } catch (Exception e) {
            // Virheilmoitus
            System.out.println("Tietojen täyttäminen epäonnistui");
            e.printStackTrace();
        }
        driver.close();

    }
}
```

Liite 3

WorkFusion RPA Express toteutus Palkkiolasku.rpae

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<recorder:ActionFlow xmlns:recorder="http://www.workfusion.com/recorder/v1.1.8"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ignoreDelays="false"
xmi:version="2.0">
    <actions actionDetails="" active="true" awaitTimeout="5000"
clazz="MozillaWindowClass" delay="0" modeType="SELECTED" pollingInterval="300"
timeoutMs="0" title="Palkkiolaskelman luonti - Mozilla Firefox"
```

varType="java.lang.Boolean" xsi:type="recorder:SwitchWindowAction">
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1"
controlClassName="MozillaWindowClass" controlName="ppkkvv-xxxx" controlText=""
delay="0" fullImageName="1519128864519.png"
imageName="1519128864519-anchor.apng" modifiers="256"
paramString="NativeMouseEvent{type=RELEASE, button=1, position=(150, 209),
clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932,
title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0),
height=865, width=810}, control=ControlData {hControl=197932,
name='ppkkvv-xxxx', className='MozillaWindowClass', value='', position=(123,
195), height=24, width=250, displayed=true, enabled=false, itemType='',
itemStatus='', frameworkId='', ancestors=''}, modifiers=256,
when=1519128864621}" pollingInterval="300" searchInFrames="false"
when="1519128864621" x="150" xsi:type="recorder:MouseClickAction" y="209"/>
        <actions actionDetails="(typed text)" active="true"
controlClassName="MozillaWindowClass" controlName="ppkkvv-xxxx" controlText=""
delay="0" keyChar="49" keyCode="2" keyLocation="1"
paramString="NativeKeyEvent{type=TYPE, keyData=(1, 49, 2, 1, 1, 0)}
NativeInputEvent{window=WindowData {pId=15064, hWnd=197932,
title='Palkkiolaskelman luonti - Mozilla Firefox',
className='MozillaWindowClass', position=(-5, 0), height=865, width=810},
control=ControlData {hControl=197932, name='ppkkvv-xxxx',
className='MozillaWindowClass', value='', position=(123, 195), height=24,
width=250, displayed=true, enabled=false, itemType='', itemStatus='',
frameworkId='', ancestors=''}, modifiers=0, when=1519128865892}" rawCode="49"
text="190954-2041" type="KEY_TYPE" when="1519128867565"
xsi:type="recorder:KeyboardAction"/>
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1"
controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0"
fullImageName="1519303161633.png"
imageName="1519303161633-anchor-1519303161720.apng" modifiers="256"
paramString="NativeMouseEvent{type=RELEASE, button=1, position=(98, 204),
clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932,
title='Palkkiolaskelman luonti

- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(7, 140), height=368, width=769, displayed=true, enabled=true, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128874517}" pollingInterval="300" searchInFrames="false" when="1519128874517" x="98" xsi:type="recorder:MouseClickAction" y="204"/>

<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0" fullImageName="1519128875577.png" imageName="1519128875577-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(192, 232), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(123, 227), height=24, width=250, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128875626}" pollingInterval="300" searchInFrames="false" when="1519128875626" x="192" xsi:type="recorder:MouseClickAction" y="232"/>

<actions actionDetails="(typed text)" active="true" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0" keyChar="49" keyCode="2" keyLocation="1" paramString="NativeKeyEvent{type=TYPE, keyData=(1, 49, 2, 1,
1, 0)} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(123, 227), height=24, width=250, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=0, when=1519128879233}" rawCode="49" text="2110002" type="KEY_TYPE" when="1519128881498" xsi:type="recorder:KeyboardAction"/>

<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="100" fullImageName="1519128882709.png" imageName="1519128882709-anchor.apng"

modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(80, 221), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(7, 140), height=368, width=769, displayed=true, enabled=true, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128882765}" pollingInterval="300" searchInFrames="false" when="1519128882765" x="80" xsi:type="recorder:MouseClickAction" y="221"/>

<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0" fullImageName="1519128885574.png" imageName="1519128885574-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(495, 207), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(423, 191), height=32, width=120, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128885643}" pollingInterval="300" searchInFrames="false" when="1519128885643" x="495" xsi:type="recorder:MouseClickAction" y="207"/>

<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="500" fullImageName="1519128888953.png" imageName="1519128888953-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(403, 570), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(387, 558), height=22, width=22, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128889021}" pollingInterval="300" searchInFrames="false" when="1519128889021" x="403"

```
xsi:type="recorder:MouseClickAction" y="570"/>
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1"
controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0"
fullImageName="1519128890953.png" imageName="1519128890953-anchor.apng"
modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1,
position=(167, 299), clickCount=1} NativeInputEvent{window=WindowData
{pId=15064, hWnd=197932, title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0),
height=865, width=810}, control=ControlData {hControl=197932, name='',
className='MozillaWindowClass', value='', position=(123, 291), height=24,
width=250, displayed=true, enabled=false, itemType='', itemStatus='',
frameworkId='', ancestors=''}, modifiers=256, when=1519128891036}"
pollingInterval="300" searchInFrames="false" when="1519128891036" x="167"
xsi:type="recorder:MouseClickAction" y="299"/>
        <actions actionDetails="(typed text)" active="true"
controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0"
keyChar="50" keyCode="3" keyLocation="1" paramString="NativeKeyEvent{type=TYPE,
keyData=(2, 50, 3, 2,
1, 0)} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932,
title='Palkkiolaskelman luonti - Mozilla Firefox',
className='MozillaWindowClass', position=(-5, 0), height=865, width=810},
control=ControlData {hControl=197932, name='', className='MozillaWindowClass',
value='', position=(123, 291), height=24, width=250, displayed=true,
enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''},
modifiers=0, when=1519128891894}" rawCode="50" text="2018" type="KEY_TYPE"
when="1519128892721" xsi:type="recorder:KeyboardAction"/>
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1"
controlClassName="MozillaWindowClass" controlName="Tilinumero" controlText=""
delay="100" fullImageName="1519128894563.png"
imageName="1519128894563-anchor.apng" modifiers="256"
paramString="NativeMouseEvent{type=RELEASE, button=1, position=(438, 301),
clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932,
title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0),
height=865, width=810}, control=ControlData {hControl=197932, name='Tilinumero',
```

className='MozillaWindowClass', value='', position=(373, 291), height=24, width=100, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128894660}" pollingInterval="300" searchInFrames="false" when="1519128894660" x="438" xsi:type="recorder:MouseClickAction" y="301"/>
<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="0" controlText="" delay="0" fullImageName="1519128895823.png" imageName="1519128895823-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(222, 344), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='0', className='MozillaWindowClass', value='', position=(0, 0), height=0, width=0, displayed=false, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128895885}" pollingInterval="300" searchInFrames="false" when="1519128895885" x="222" xsi:type="recorder:MouseClickAction" y="344"/>
<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="3 - palkkio" controlText="" delay="300" fullImageName="1519128897208.png" imageName="1519128897208-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(201, 455), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='3 - palkkio', className='MozillaWindowClass', value='', position=(123, 430), height=40, width=250, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128897257}" pollingInterval="300" searchInFrames="false" when="1519128897257" x="201" xsi:type="recorder:MouseClickAction" y="455"/>
<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="100"

fullImageName="1519128898485.png" imageName="1519128898485-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(241, 355), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(123, 354), height=32, width=250, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128898541}" pollingInterval="300" searchInFrames="false" when="1519128898541" x="241" xsi:type="recorder:MouseClickAction" y="355"/>

<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0" fullImageName="1519128898962.png" imageName="1519128898962-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(248, 369), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(123, 358), height=24, width=250, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128899024}" pollingInterval="300" searchInFrames="false" when="1519128899024" x="248" xsi:type="recorder:MouseClickAction" y="369"/>

<actions actionDetails="(typed text)" active="true" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0" keyChar="50" keyCode="3" keyLocation="1" paramString="NativeKeyEvent{type=TYPE, keyData=(2, 50, 3, 2, 1, 0)} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(123, 358), height=24, width=250, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=0, when=1519128903000}" rawCode="50" text="21TUNNUS3" type="KEY_TYPE" when="1519128903375" xsi:type="recorder:KeyboardAction"/></actions>

```
    <actions actionDetails="" active="true" awaitTimeout="5000"
clazz="MozillaWindowClass" delay="0" modeType="SELECTED" pollingInterval="300"
timeoutMs="0" title="Palkkiolaskelman luonti - Mozilla Firefox"
varType="java.lang.Boolean" xsi:type="recorder:SwitchWindowAction">
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1"
controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0"
fullImageName="1519130710964.png" imageName="1519130710964-anchor.apng"
modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1,
position=(404, 95), clickCount=1} NativeInputEvent{window=WindowData {pId=15064,
hWnd=197932, title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0),
height=865, width=810}, control=ControlData {hControl=197932, name='',
className='MozillaWindowClass', value='', position=(324, 75), height=32,
width=100, displayed=true, enabled=false, itemType='', itemStatus='',
frameworkId='', ancestors=''}, modifiers=256, when=1519130711016}"
pollingInterval="300" searchInFrames="false" when="1519130711016" x="404"
xsi:type="recorder:MouseClickAction" y="95"/>
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1"
controlClassName="MozillaWindowClass" controlName="" controlText="" delay="500"
fullImageName="1519130713589.png" imageName="1519130713589-anchor.apng"
modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1,
position=(263, 602), clickCount=1} NativeInputEvent{window=WindowData
{pId=15064, hWnd=197932, title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0),
height=865, width=810}, control=ControlData {hControl=197932, name='',
className='MozillaWindowClass', value='', position=(216, 582), height=32,
width=76, displayed=true, enabled=false, itemType='', itemStatus='',
frameworkId='', ancestors=''}, modifiers=256, when=1519130713700}"
pollingInterval="300" searchInFrames="false" when="1519130713700" x="263"
xsi:type="recorder:MouseClickAction" y="602"/>
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1" controlClassName=""
controlName="Ei valittua tiedostoa." controlText="" delay="500"
fullImageName="1519130716100.png" imageName="1519130716100-anchor.apng"
modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1,
```

position=(511, 458), clickCount=1} NativeInputEvent{window=WindowData
{pId=15064, hWnd=197932, title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0),
height=865, width=810}, control=<NA>, modifiers=256, when=1519130716224}"
pollingInterval="300" searchInFrames="false" when="1519130716224" x="511"
xsi:type="recorder:MouseClickAction" y="458"/>
    </actions>
    <actions actionDetails="" active="true" awaitTimeout="5000" clazz="#32770"
delay="2100" modeType="SELECTED" pollingInterval="300" timeoutMs="0"
title="Lähetä tiedosto" varType="java.lang.Boolean"
xsi:type="recorder:SwitchWindowAction">
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1" controlClassName="DirectUIHWND"
controlName="Nimi" controlText="liite2" delay="500"
fullImageName="1519129800833.png"
imageName="1519129800833-anchor-1519129800894.apng" modifiers="256"
offsetX="-88" offsetY="-21" paramString="NativeMouseEvent{type=RELEASE,
button=1, position=(219, 169), clickCount=1} NativeInputEvent{window=WindowData
{pId=15064, hWnd=3672086, title='Lähetä tiedosto', className='#32770',
position=(1, 0), height=480, width=800}, control=ControlData {hControl=591976,
name='Nimi', className='DirectUIHWND', value='liite2', position=(205, 149),
height=22, width=236, displayed=true, enabled=false, itemType='', itemStatus='',
frameworkId='DirectUI', ancestors=''}, modifiers=256, when=1519128918949}"
pollingInterval="300" searchInFrames="false" when="1519128918949" x="219"
xsi:type="recorder:MouseClickAction" y="169"/>
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" delay="0" fullImageName="1519130168586.png"
imageName="1519130168586-anchor-1519130168656.apng" offsetX="-51" offsetY="20"
pollingInterval="300" searchInFrames="false"
xsi:type="recorder:MouseClickAction"/></actions>
    <actions actionDetails="" active="true" awaitTimeout="5000"
clazz="MozillaWindowClass" delay="1700" modeType="SELECTED"
pollingInterval="300" timeoutMs="0" title="Palkkiolaskelman luonti - Mozilla
Firefox" varType="java.lang.Boolean" xsi:type="recorder:SwitchWindowAction">
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1"
controlClassName="MozillaWindowClass" controlName="" controlText="" delay="0"

fullImageName="1519128920934.png" imageName="1519128920934-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(318, 557), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(259, 535), height=32, width=92, displayed=true, enabled=false, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128921033}" pollingInterval="300" searchInFrames="false" when="1519128921033" x="318" xsi:type="recorder:MouseClickAction" y="557"/>

<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="300" fullImageName="1519128923626.png" imageName="1519128923626-anchor.apng" modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(404, 596), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(372, 582), height=32, width=73, displayed=true, enabled=true, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128923725}" pollingInterval="300" searchInFrames="false" when="1519128923725" x="404" xsi:type="recorder:MouseClickAction" y="596"/>

<actions actionDetails="(click left button)" active="true" awaitTimeout="5000" button="1" clickCount="1" controlClassName="MozillaWindowClass" controlName="" controlText="" delay="300" fullImageName="1519128927148.png" imageName="1519128927148-anchor.apng" modifiers="256" offsetX="24" paramString="NativeMouseEvent{type=RELEASE, button=1, position=(734, 84), clickCount=1} NativeInputEvent{window=WindowData {pId=15064, hWnd=197932, title='Palkkiolaskelman luonti - Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0), height=865, width=810}, control=ControlData {hControl=197932, name='', className='MozillaWindowClass', value='', position=(693, 75), height=32, width=89, displayed=true, enabled=true, itemType='', itemStatus='', frameworkId='', ancestors=''}, modifiers=256, when=1519128927195}"

```
pollingInterval="300" searchInFrames="false" when="1519128927195" x="734"
xsi:type="recorder:MouseClickAction" y="84"/>
        <actions actionDetails="(click left button)" active="true"
awaitTimeout="5000" button="1" clickCount="1"
controlClassName="MozillaWindowClass" controlName="" controlText="" delay="500"
fullImageName="1519128929585.png" imageName="1519128929585-anchor.apng"
modifiers="256" paramString="NativeMouseEvent{type=RELEASE, button=1,
position=(249, 577), clickCount=1} NativeInputEvent{window=WindowData
{pId=15064, hWnd=197932, title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0),
height=865, width=810}, control=ControlData {hControl=197932, name='',
className='MozillaWindowClass', value='', position=(203, 556), height=32,
width=89, displayed=true, enabled=true, itemType='', itemStatus='',
frameworkId='', ancestors=''}, modifiers=256, when=1519128929708}"
pollingInterval="300" searchInFrames="false" when="1519128929708" x="249"
xsi:type="recorder:MouseClickAction" y="577"/>
        <actions actionDetails="" active="true" awaitTimeout="5000"
controlClassName="MozillaWindowClass" controlName="Päivitä" controlText=""
delay="400" fullImageName="1519128933542.png"
imageName="1519128933542-anchor.apng" paramString="NativeMouseEvent{type=MOVE,
button=0, position=(87, 59), clickCount=0} NativeInputEvent{window=WindowData
{pId=15064, hWnd=197932, title='Palkkiolaskelman luonti
- Mozilla Firefox', className='MozillaWindowClass', position=(-5, 0),
height=865, width=810}, control=ControlData {hControl=197932, name='Päivitä',
className='MozillaWindowClass', value='', position=(70, 34), height=40,
width=32, displayed=true, enabled=false, itemType='', itemStatus='',
frameworkId='', ancestors=''}, modifiers=0, when=1519128933542}"
pollingInterval="300" when="1519128933542" x="87"
xsi:type="recorder:MouseMoveAction" y="59"/></actions>
    <actions actionDetails="for 5000 ms" active="true" delay="5000"
xsi:type="recorder:WaitAction"/> </recorder:ActionFlow>
```


Liite 4

UiPath toteutus Main.xaml

```
<Activity mc:Ignorable="sap sap2010 sads" x:Class="Main"
mva:VisualBasic.Settings="{x:Null}" sap2010:WorkflowViewState.IdRef="Main_1"
 xmlns="http://schemas.microsoft.com/netfx/2009/xaml/activities"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:mva="clr-namespace:Microsoft.VisualBasic.Activities;assembly=
System.Activities"
 xmlns:sads="http://schemas.microsoft.com/netfx/2010/xaml/activities/
 debugger"

xmlns:sap="http://schemas.microsoft.com/netfx/2009/xaml/activities/presentation"

xmlns:sap2010="http://schemas.microsoft.com/netfx/2010/xaml/activities/
presentation"
 xmlns:scg="clr-namespace:System.Collections.Generic;assembly=mscorlib"
 xmlns:sco="clr-namespace:System.Collections.ObjectModel;assembly=mscorlib"
 xmlns:ui="http://schemas.uipath.com/workflow/activities"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <TextExpression.NamespacesForImplementation>
    <sco:Collection x:TypeArguments="x:String">
      <x:String>System.Activities</x:String>
      <x:String>System.Activities.Statements</x:String>
      <x:String>System.Activities.Expressions</x:String>
      <x:String>System.Activities.Validation</x:String>
      <x:String>System.Activities.XamlIntegration</x:String>
      <x:String>Microsoft.VisualBasic</x:String>
      <x:String>Microsoft.VisualBasic.Activities</x:String>
      <x:String>System</x:String>
      <x:String>System.Collections</x:String>
      <x:String>System.Collections.Generic</x:String>
      <x:String>System.Data</x:String>
      <x:String>System.Diagnostics</x:String>
      <x:String>System.Drawing</x:String>
      <x:String>System.IO</x:String>
      <x:String>System.Linq</x:String>
      <x:String>System.Net.Mail</x:String>
      <x:String>System.Xml</x:String>
```

```
      <x:String>System.Xml.Linq</x:String>
      <x:String>UiPath.Core</x:String>
      <x:String>UiPath.Core.Activities</x:String>
      <x:String>System.Windows.Markup</x:String>
    </sco:Collection>
  </TextExpression.NamespacesForImplementation>
  <TextExpression.ReferencesForImplementation>
    <sco:Collection x:TypeArguments="AssemblyReference">
      <AssemblyReference>System.Activities</AssemblyReference>
      <AssemblyReference>Microsoft.VisualBasic</AssemblyReference>
      <AssemblyReference>mscorlib</AssemblyReference>
      <AssemblyReference>System.Data</AssemblyReference>
      <AssemblyReference>System</AssemblyReference>
      <AssemblyReference>System.Drawing</AssemblyReference>
      <AssemblyReference>System.Core</AssemblyReference>
      <AssemblyReference>System.Xml</AssemblyReference>
      <AssemblyReference>System.Xml.Linq</AssemblyReference>
      <AssemblyReference>UiPath.Core</AssemblyReference>
      <AssemblyReference>UiPath.Core.Activities</AssemblyReference>
      <AssemblyReference>PresentationFramework</AssemblyReference>
      <AssemblyReference>WindowsBase</AssemblyReference>
      <AssemblyReference>PresentationCore</AssemblyReference>
      <AssemblyReference>System.Xaml</AssemblyReference>
      <AssemblyReference>System.ComponentModel.Composition</AssemblyReference>
      <AssemblyReference>System.ServiceModel</AssemblyReference>
      <AssemblyReference>System.Runtime.WindowsRuntime</AssemblyReference>
    </sco:Collection>
  </TextExpression.ReferencesForImplementation>
  <Sequence DisplayName="Recording Sequence"
sap2010:WorkflowViewState.IdRef="Sequence_3">
    <ui:BrowserScope Browser="{x:Null}" SearchScope="{x:Null}"
TimeoutMS="{x:Null}" UiBrowser="{x:Null}" BrowserType="Firefox"
DisplayName="Attach Browser 'Palkkiola Page'"
sap2010:WorkflowViewState.IdRef="BrowserScope_2"
InformativeScreenshot="112ec4867927506bdcb3dad6e90da35a" Selector="&lt;html
app='firefox.exe' htmlwindowname='WIDxDefaultExternal1518691788034'
title='Palkkiolaskelman luonti' /&gt;">
```

```
      <ui:BrowserScope.Body>
        <ActivityAction x:TypeArguments="x:Object">
          <ActivityAction.Argument>
            <DelegateInArgument x:TypeArguments="x:Object" Name="ContextTarget"
/>
          </ActivityAction.Argument>
          <Sequence DisplayName="Do"
sap2010:WorkflowViewState.IdRef="Sequence_4">
            <ui:TypeInto DelayBefore="{x:Null}" DelayBetweenKeys="{x:Null}"
DelayMS="{x:Null}" Activate="True" ClickBeforeTyping="False" DisplayName="Type
into 'INPUT  inFormHeaderPERS...'" EmptyField="False"
sap2010:WorkflowViewState.IdRef="TypeInto_5" SendWindowMessages="False"
SimulateType="True" Text="190954-2041">
              <ui:TypeInto.Target>
                <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="696532ef324929bdd691d5aa0cd00842"
Selector="&lt;webctrl id='inFormHeaderPERS_ID-inner' tag='INPUT' /&gt;"
WaitForReady="COMPLETE" />
              </ui:TypeInto.Target>
            </ui:TypeInto>
            <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'BUTTON  btnGetHr'"
sap2010:WorkflowViewState.IdRef="Click_10" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
              <ui:Click.CursorPosition>
                <ui:CursorPosition OffsetX="61" OffsetY="13" Position="TopLeft"
/>
              </ui:Click.CursorPosition>
              <ui:Click.Target>
                <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="cbe804a8f7fd2bd13d18a295c9208a64"
Selector="&lt;webctrl id='btnGetHr' tag='BUTTON' /&gt;" WaitForReady="COMPLETE"
/>
              </ui:Click.Target>
            </ui:Click>
            <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'DIV  rgUpdateNeeded-0-B...'"
```

```
sap2010:WorkflowViewState.IdRef="Click_11" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
              <ui:Click.CursorPosition>
                <ui:CursorPosition OffsetX="7" OffsetY="5" Position="TopLeft" />
              </ui:Click.CursorPosition>
              <ui:Click.Target>
                <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="78d703c7147b5aa268b1e2bb08b886cf"
Selector="&lt;webctrl id='rgUpdateNeeded-0-Button' tag='DIV' /&gt;"
WaitForReady="COMPLETE" />
              </ui:Click.Target>
            </ui:Click>
            <ui:TypeInto DelayBefore="{x:Null}" DelayBetweenKeys="{x:Null}"
DelayMS="{x:Null}" Activate="True" ClickBeforeTyping="False" DisplayName="Type
into 'INPUT  inFormHeaderCOST...'" EmptyField="False"
sap2010:WorkflowViewState.IdRef="TypeInto_6" SendWindowMessages="False"
SimulateType="False" Text="[&quot;2110002&quot;  + [k(enter)]]">
              <ui:TypeInto.Target>
                <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="005c6667f7270c338d0afbc02d1efcde"
Selector="&lt;webctrl id='inFormHeaderCOST_CENTER-inner' tag='INPUT' /&gt;"
WaitForReady="COMPLETE" />
              </ui:TypeInto.Target>
            </ui:TypeInto>
            <ui:TypeInto DelayBefore="{x:Null}" DelayBetweenKeys="{x:Null}"
DelayMS="{x:Null}" Activate="True" ClickBeforeTyping="False" DisplayName="Type
into 'INPUT  inFormHeaderPAYM...'" EmptyField="False"
sap2010:WorkflowViewState.IdRef="TypeInto_7" SendWindowMessages="False"
SimulateType="False" Text="[&quot;2018&quot; + [k(enter)]]">
              <ui:TypeInto.Target>
                <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="a262f0ec735027f8d678159a24488846"
Selector="&lt;webctrl id='inFormHeaderPAYMENT_YEAR-inner' tag='INPUT' /&gt;"
WaitForReady="COMPLETE" />
              </ui:TypeInto.Target>
            </ui:TypeInto>
            <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
```

```
ClickType="CLICK_SINGLE" DisplayName="Click 'LABEL  inFormHeaderDOCU...'"
sap2010:WorkflowViewState.IdRef="Click_13" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
              <ui:Click.CursorPosition>
                <ui:CursorPosition OffsetX="25" OffsetY="17" Position="TopLeft"
/>
              </ui:Click.CursorPosition>
              <ui:Click.Target>
                <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="6524a85d37557197a163b4d5dd8c6e81"
Selector="&lt;webctrl id='inFormHeaderDOCUMENT_TYPE-label' tag='LABEL' /&gt;"
WaitForReady="COMPLETE" />
              </ui:Click.Target>
            </ui:Click>
            <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'LI  __item3'"
sap2010:WorkflowViewState.IdRef="Click_14" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
              <ui:Click.CursorPosition>
                <ui:CursorPosition OffsetX="77" OffsetY="14" Position="TopLeft"
/>
              </ui:Click.CursorPosition>
              <ui:Click.Target>
                <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="70d659fe94b08fa442c25e29977cb120"
Selector="&lt;webctrl id='__item3' tag='LI' /&gt;" WaitForReady="COMPLETE" />
              </ui:Click.Target>
            </ui:Click>
            <ui:TypeInto DelayBefore="{x:Null}" DelayBetweenKeys="{x:Null}"
DelayMS="{x:Null}" Activate="True" ClickBeforeTyping="False" DisplayName="Type
into 'INPUT  inFormHeaderAPPR...'" EmptyField="False"
sap2010:WorkflowViewState.IdRef="TypeInto_8" SendWindowMessages="False"
SimulateType="False" Text="[&quot;21TUNNUS3&quot; +[k(enter)]]">
              <ui:TypeInto.Target>
                <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="a2b00c92595f09275811c9ba15cb97f0"
Selector="&lt;webctrl id='inFormHeaderAPPROVER-inner' tag='INPUT' /&gt;"
```

```
WaitForReady="COMPLETE" />
              </ui:TypeInto.Target>
            </ui:TypeInto>
          </Sequence>
        </ActivityAction>
      </ui:BrowserScope.Body>
    </ui:BrowserScope>
    <Sequence DisplayName="Recording Sequence"
sap2010:WorkflowViewState.IdRef="Sequence_6">
      <ui:BrowserScope Browser="{x:Null}" SearchScope="{x:Null}"
TimeoutMS="{x:Null}" UiBrowser="{x:Null}" BrowserType="Firefox"
DisplayName="Attach Browser 'Palkkiola Page'"
sap2010:WorkflowViewState.IdRef="BrowserScope_3"
InformativeScreenshot="410b00ea126b0c83cfa47fedf89e0050" Selector="&lt;html
app='firefox.exe' htmlwindowname='WIDxDefaultExternal1518691788034'
title='Palkkiolaskelman luonti' /&gt;">
        <ui:BrowserScope.Body>
          <ActivityAction x:TypeArguments="x:Object">
            <ActivityAction.Argument>
              <DelegateInArgument x:TypeArguments="x:Object"
Name="ContextTarget" />
            </ActivityAction.Argument>
            <Sequence DisplayName="Do"
sap2010:WorkflowViewState.IdRef="Sequence_5">
              <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'BUTTON  btnAttachments'"
sap2010:WorkflowViewState.IdRef="Click_15" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
                <ui:Click.CursorPosition>
                  <ui:CursorPosition OffsetX="74" OffsetY="20"
Position="TopLeft" />
                </ui:Click.CursorPosition>
                <ui:Click.Target>
                  <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="801bacc7ff7c4c37f61c7a1402a972b4"
Selector="&lt;webctrl id='btnAttachments' tag='BUTTON' /&gt;"
WaitForReady="COMPLETE" />
```

```
                </ui:Click.Target>
              </ui:Click>
              <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'BUTTON  btnUpload'"
sap2010:WorkflowViewState.IdRef="Click_16" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
                <ui:Click.CursorPosition>
                  <ui:CursorPosition OffsetX="61" OffsetY="14"
Position="TopLeft" />
                </ui:Click.CursorPosition>
                <ui:Click.Target>
                  <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="94ffd0f2008f1ce4ee3479376b63e539"
Selector="&lt;webctrl id='btnUpload' tag='BUTTON' /&gt;" WaitForReady="COMPLETE"
/>
                </ui:Click.Target>
              </ui:Click>
              <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'INPUT  fileUpload-fu'"
sap2010:WorkflowViewState.IdRef="Click_17" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
                <ui:Click.CursorPosition>
                  <ui:CursorPosition OffsetX="2165" OffsetY="19"
Position="TopLeft" />
                </ui:Click.CursorPosition>
                <ui:Click.Target>
                  <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="1090826352eacb2dbe3de8fb37d6b91a"
Selector="&lt;webctrl id='fileUpload-fu' tag='INPUT' /&gt;"
WaitForReady="COMPLETE" />
                </ui:Click.Target>
              </ui:Click>
            </Sequence>
          </ActivityAction>
        </ui:BrowserScope.Body>
      </ui:BrowserScope>
    </Sequence>
```

```
    <Sequence DisplayName="Recording Sequence"
sap2010:WorkflowViewState.IdRef="Sequence_7">
      <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'editable text  Nimi'"
sap2010:WorkflowViewState.IdRef="Click_18" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="False">
        <ui:Click.CursorPosition>
          <ui:CursorPosition OffsetX="65" OffsetY="11" Position="TopLeft" />
        </ui:Click.CursorPosition>
        <ui:Click.Target>
          <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="19e9f19320a69dcdd2f7ec61bacd5f70"
Selector="&lt;wnd app='firefox.exe' cls='#32770' title='Lähetä tiedosto'
/&gt;&lt;wnd aaname='Resurssienhallintaruutu' cls='DirectUIHWND' /&gt;&lt;wnd
ctrlid='1121' title='ShellView' /&gt;&lt;wnd aaname='Kohdenäkymä'
cls='DirectUIHWND' /&gt;&lt;ctrl name='Kohdenäkymä' role='list' /&gt;&lt;ctrl
automationid='1' /&gt;&lt;ctrl automationid='System.ItemNameDisplay' /&gt;"
WaitForReady="INTERACTIVE" />
        </ui:Click.Target>
      </ui:Click>
    </Sequence>
    <Sequence DisplayName="Recording Sequence"
sap2010:WorkflowViewState.IdRef="Sequence_8">
      <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'Button'"
sap2010:WorkflowViewState.IdRef="Click_19" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="False">
        <ui:Click.CursorPosition>
          <ui:CursorPosition OffsetX="55" OffsetY="16" Position="TopLeft" />
        </ui:Click.CursorPosition>
        <ui:Click.Target>
          <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="b97d3c31ffc3d840f307e63ef5851d70"
Selector="&lt;wnd app='firefox.exe' cls='#32770' title='Lähetä tiedosto'
/&gt;&lt;wnd ctrlid='1' title='&amp;amp;Avaa' /&gt;" WaitForReady="INTERACTIVE"
/>
        </ui:Click.Target>
```

```
      </ui:Click>
    </Sequence>
    <Sequence DisplayName="Recording Sequence"
sap2010:WorkflowViewState.IdRef="Sequence_10">
      <ui:BrowserScope Browser="{x:Null}" SearchScope="{x:Null}"
TimeoutMS="{x:Null}" UiBrowser="{x:Null}" BrowserType="Firefox"
DisplayName="Attach Browser 'Palkkiola Page'"
sap2010:WorkflowViewState.IdRef="BrowserScope_4"
InformativeScreenshot="691fe149e13e0601c1db60f0ce3094b9" Selector="&lt;html
app='firefox.exe' htmlwindowname='WIDxDefaultExternal1518691788034'
title='Palkkiolaskelman luonti' /&gt;">
        <ui:BrowserScope.Body>
          <ActivityAction x:TypeArguments="x:Object">
            <ActivityAction.Argument>
              <DelegateInArgument x:TypeArguments="x:Object"
Name="ContextTarget" />
            </ActivityAction.Argument>
            <Sequence DisplayName="Do"
sap2010:WorkflowViewState.IdRef="Sequence_9">
              <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'BUTTON  btnUp1'"
sap2010:WorkflowViewState.IdRef="Click_20" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
                <ui:Click.CursorPosition>
                  <ui:CursorPosition OffsetX="55" OffsetY="15"
Position="TopLeft" />
                </ui:Click.CursorPosition>
                <ui:Click.Target>
                  <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="58c9a0b58ba02b46c58fa736420a3f7c"
Selector="&lt;webctrl id='btnUp1' tag='BUTTON' /&gt;" WaitForReady="COMPLETE" />
                </ui:Click.Target>
              </ui:Click>
              <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'BUTTON  btnCancelAtt1'"
sap2010:WorkflowViewState.IdRef="Click_21" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
```

```
                <ui:Click.CursorPosition>
                  <ui:CursorPosition OffsetX="33" OffsetY="8" Position="TopLeft"
/>
                </ui:Click.CursorPosition>
                <ui:Click.Target>
                  <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="2ff6c4b89b36848b13b4356201c54113"
Selector="&lt;webctrl id='btnCancelAtt1' tag='BUTTON' /&gt;"
WaitForReady="COMPLETE" />
                </ui:Click.Target>
              </ui:Click>
              <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'BUTTON  btnAccept'"
sap2010:WorkflowViewState.IdRef="Click_22" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
                <ui:Click.CursorPosition>
                  <ui:CursorPosition OffsetX="24" OffsetY="18"
Position="TopLeft" />
                </ui:Click.CursorPosition>
                <ui:Click.Target>
                  <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="90138ef4e36202338b21385e223fa59e"
Selector="&lt;webctrl id='btnAccept' tag='BUTTON' /&gt;" WaitForReady="COMPLETE"
/>
                </ui:Click.Target>
              </ui:Click>
              <ui:Click DelayBefore="{x:Null}" DelayMS="{x:Null}"
ClickType="CLICK_SINGLE" DisplayName="Click 'BUTTON  btnAcp2'"
sap2010:WorkflowViewState.IdRef="Click_23" KeyModifiers="None"
MouseButton="BTN_LEFT" SendWindowMessages="False" SimulateClick="True">
                <ui:Click.CursorPosition>
                  <ui:CursorPosition OffsetX="68" OffsetY="12"
Position="TopLeft" />
                </ui:Click.CursorPosition>
                <ui:Click.Target>
                  <ui:Target ClippingRegion="{x:Null}" Element="{x:Null}"
TimeoutMS="{x:Null}" InformativeScreenshot="49a10bc95b07232a23bf0a23fcbb728d"
```

```
Selector="&lt;webctrl id=’btnAcp2’ tag=’BUTTON’ /&gt;" WaitForReady="COMPLETE"
/>
                </ui:Click.Target>
              </ui:Click>
            </Sequence>
          </ActivityAction>
        </ui:BrowserScope.Body>
      </ui:BrowserScope>
    </Sequence>

<sads:DebugSymbol.Symbol>dzdDOlxVc2Vyc1xUdXVra2FcRG9jdW1lbnRzXFVpUGF0aFxwYWxra2l
</sads:DebugSymbol.Symbol>
  </Sequence>
  <sap2010:WorkflowViewState.ViewStateManager>
    <sap2010:ViewStateManager>
      <sap2010:ViewStateData Id="TypeInto_5"
sap:VirtualizedContainerService.HintSize="314,134" />
      <sap2010:ViewStateData Id="Click_10"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Click_11"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="TypeInto_6"
sap:VirtualizedContainerService.HintSize="314,134" />
      <sap2010:ViewStateData Id="TypeInto_7"
sap:VirtualizedContainerService.HintSize="314,134" />
      <sap2010:ViewStateData Id="Click_13"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Click_14"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="TypeInto_8"
sap:VirtualizedContainerService.HintSize="314,134" />
      <sap2010:ViewStateData Id="Sequence_4"
sap:VirtualizedContainerService.HintSize="336,1364">
        <sap:WorkflowViewStateService.ViewState>
          <scg:Dictionary x:TypeArguments="x:String, x:Object">
            <x:Boolean x:Key="IsExpanded">True</x:Boolean>
          </scg:Dictionary>
```

```
        </sap:WorkflowViewStateService.ViewState>
      </sap2010:ViewStateData>
      <sap2010:ViewStateData Id="BrowserScope_2"
sap:VirtualizedContainerService.HintSize="414,1510" />
      <sap2010:ViewStateData Id="Click_15"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Click_16"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Click_17"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Sequence_5"
sap:VirtualizedContainerService.HintSize="336,522">
        <sap:WorkflowViewStateService.ViewState>
          <scg:Dictionary x:TypeArguments="x:String, x:Object">
            <x:Boolean x:Key="IsExpanded">True</x:Boolean>
          </scg:Dictionary>
        </sap:WorkflowViewStateService.ViewState>
      </sap2010:ViewStateData>
      <sap2010:ViewStateData Id="BrowserScope_3"
sap:VirtualizedContainerService.HintSize="414,668" />
      <sap2010:ViewStateData Id="Sequence_6"
sap:VirtualizedContainerService.HintSize="414,51">
        <sap:WorkflowViewStateService.ViewState>
          <scg:Dictionary x:TypeArguments="x:String, x:Object">
            <x:Boolean x:Key="IsExpanded">False</x:Boolean>
            <x:Boolean x:Key="IsPinned">False</x:Boolean>
          </scg:Dictionary>
        </sap:WorkflowViewStateService.ViewState>
      </sap2010:ViewStateData>
      <sap2010:ViewStateData Id="Click_18"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Sequence_7"
sap:VirtualizedContainerService.HintSize="414,51">
        <sap:WorkflowViewStateService.ViewState>
          <scg:Dictionary x:TypeArguments="x:String, x:Object">
            <x:Boolean x:Key="IsExpanded">False</x:Boolean>
            <x:Boolean x:Key="IsPinned">False</x:Boolean>
```

```
          </scg:Dictionary>
        </sap:WorkflowViewStateService.ViewState>
      </sap2010:ViewStateData>
      <sap2010:ViewStateData Id="Click_19"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Sequence_8"
sap:VirtualizedContainerService.HintSize="414,51">
        <sap:WorkflowViewStateService.ViewState>
          <scg:Dictionary x:TypeArguments="x:String, x:Object">
            <x:Boolean x:Key="IsExpanded">False</x:Boolean>
            <x:Boolean x:Key="IsPinned">False</x:Boolean>
          </scg:Dictionary>
        </sap:WorkflowViewStateService.ViewState>
      </sap2010:ViewStateData>
      <sap2010:ViewStateData Id="Click_20"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Click_21"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Click_22"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Click_23"
sap:VirtualizedContainerService.HintSize="314,106" />
      <sap2010:ViewStateData Id="Sequence_9"
sap:VirtualizedContainerService.HintSize="336,668">
        <sap:WorkflowViewStateService.ViewState>
          <scg:Dictionary x:TypeArguments="x:String, x:Object">
            <x:Boolean x:Key="IsExpanded">True</x:Boolean>
          </scg:Dictionary>
        </sap:WorkflowViewStateService.ViewState>
      </sap2010:ViewStateData>
      <sap2010:ViewStateData Id="BrowserScope_4"
sap:VirtualizedContainerService.HintSize="414,814" />
      <sap2010:ViewStateData Id="Sequence_10"
sap:VirtualizedContainerService.HintSize="414,51">
        <sap:WorkflowViewStateService.ViewState>
          <scg:Dictionary x:TypeArguments="x:String, x:Object">
            <x:Boolean x:Key="IsExpanded">False</x:Boolean>
```

```
            <x:Boolean x:Key="IsPinned">False</x:Boolean>
          </scg:Dictionary>
        </sap:WorkflowViewStateService.ViewState>
      </sap2010:ViewStateData>
      <sap2010:ViewStateData Id="Sequence_3"
sap:VirtualizedContainerService.HintSize="436,1998">
        <sap:WorkflowViewStateService.ViewState>
          <scg:Dictionary x:TypeArguments="x:String, x:Object">
            <x:Boolean x:Key="IsExpanded">True</x:Boolean>
          </scg:Dictionary>
        </sap:WorkflowViewStateService.ViewState>
      </sap2010:ViewStateData>
      <sap2010:ViewStateData Id="Main_1"
sap:VirtualizedContainerService.HintSize="476,2078" />
    </sap2010:ViewStateManager>
  </sap2010:WorkflowViewState.ViewStateManager>
</Activity>
```