Opinnäytetyö AMK

Tietojenkäsittely

2022

Miikka Wirtanen

Kielipalvelun työvaiheen automatisointi ohjelmistorobotiikan keinoin

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | tiivistelmä

Turun ammattikorkeakoulu

Tietojenkäsittely

2022| 44 sivua

Miikka Wirtanen

Kielipalvelun työvaiheen automatisointi ohjelmistorobotiikan keinoin

Ohjelmistorobotiikka on teknologia, jonka avulla on mahdollista automatisoida sovelluksia hyödyntämällä samaa käyttöliittymää, jonka kautta ihminenkin työskentelee. Teknologia soveltuu hyvin erilaisten rutiininomaisten työtehtävien hoitoon.

Tässä opinnäytetyössä oli tarkoituksena suunnitella ja toteuttaa kielipalveluihin ja -teknologioihin erikoistuneen toimeksiantajayrityksen työvaiheen automatisointi ohjelmistorobotiikan keinoin. Automaation kohteeksi valikoitui käännöstyökalun projektikohtaisten oletusasetusten säätäminen. Tavoitteena oli työprosessia nopeuttamalla tuottaa yritykselle rahallisia säästöjä sekä arvioida toteutuksen kautta ohjelmistorobotiikan soveltuvuutta kielipalvelun työprosesseihin laajemminkin. Pääasiallisina työkaluina robotin rakentamisessa käytettiin tähän suunnattua automaatioalustaa, koodieditoria sekä käyttöliittymän analyysiin tarkoitettua sovellusta.

Työn lopputuloksena syntyi toimiva robotti, joka kykenee suorittamaan asetusten säädön paljon ihmistä nopeammin. Tämä johtaa pitkällä aikavälillä rahallisiin säästöihin, kun työhön käytettävä aika vähenee. Ohjelmistorobotiikka vaikuttaa toteutuksen perusteella soveltuvan hyvin muihinkin yrityksen työvaiheisiin, kunhan tietyt tekniset sekä osaamiseen, dokumentaatioon ja työn luonteeseen liittyvät edellytykset täyttyvät.

Asiasanat:

Robot Framework, SDL Trados Studio, ohjelmistokehitys, ohjelmistorobotiikka, kielipalvelut

Bachelor´s Thesis | Abstract

Turku University of Applied Sciences

Business Information Technology

2022| 44 pages

Miikka Wirtanen

Automating work stage in language services using robotic process automation

Robotic process automation is a technology that enables automation of applications by utilizing the same interface with which humans also work. The technology is well suited for conducting various routine tasks.

The purpose of this thesis was to design and implement the automation of work stage using robotic process automation for the client company that specializes in language services and technologies. The adjusting of project-specific default settings of the translation tool was selected as the target for automation. The aim was to produce monetary savings for the company by speeding up the work process and evaluating in a broader scope the usability of robotic process automation for work processes of language services. The main tools used in the construction of the robot were an automation platform designed for this purpose, a code editor and an application specifically used for analysis of user interface.

The end result was a working robot that is capable of performing the adjustment of settings much faster than a human. This will lead to monetary savings in the long term as the time spent on the work decreases. Based on the implemented case the robotic process automation seems well suited for other work processes of the company as long as certain technical and other requirements concerning know-how, documentation and nature of work are fulfilled.

Keywords:

Robot Framework, SDL Trados Studio, software development, robotic process automation, language services

Sisältö

Kuvat

Taulukot

Kuviot

Käytetyt lyhenteet ja sanasto

CAT	Tietokoneavusteinen käännöstyökalu (Computer-assisted translation tool) erilaisten dokumenttien kääntämistä varten.
Debuggaus	Virheiden paikallistaminen ja korjaaminen koodista.
Elementti	Tässä kontekstissa jokin sovelluksen käyttöliittymän osa, esimerkiksi ikkuna, painike tai valikko.
IDE	Ohjelmointiympäristö (Integrated development environment), jossa ohjelmoija suunnittelee ja rakentaa ohjelmia.
Refaktorointi	Koodin rakenteen muuttaminen ilman, että sen toiminnallisuus muuttuu. Esimerkiksi koodin luettavuuden selkeyttäminen.
Reformatointi	Koodin ulkoasun uudelleenjärjestely vaatimusten mukaiseksi.
RPA	Englanninkielinen termi ohjelmistorobotiikalle (Robotic process automation).
Skripti	Tiedosto, joka sisältää komentoja ohjelmalle jollakin ohjelmointikielellä.

1 Johdanto

Tämän opinnäytetyön tarkoituksena on suunnitella ja toteuttaa toimeksiantajana olevan yrityksen työvaiheen automatisointi ohjelmistorobotiikan keinoin. Yritys on erikoistunut kieli- ja käännöspalveluihin sekä näitä tukeviin teknologioihin. Kielipalvelun työprosessit sisältävät erilaisia vaiheita, joista osa on luonteeltaan rutiininomaisia ja aikaa vieviä. Näiden prosessien automatisointi säästäisi ajallisia resursseja, kun työntekijät voisivat keskittyä vaativampiin tehtäviin. Tämä puolestaan johtaisi rahallisiin säästöihin työn tehokkuuden lisääntymisen myötä. Yksittäisen työvaiheen automaatiolla on tarkoituksena myös arvioida ohjelmistorobotiikan soveltuvuutta yrityksen kielipalvelun prosesseihin yleisemminkin.

Automatisoinnin kohteeksi opinnäytetyössä on valikoitunut projektikohtaisten asetusten säätäminen SDL Trados Studio -ohjelmistossa, joka on erilaisten dokumenttien käännöstyöhön tarkoitettu työkalu. Ohjelma sisältää runsaasti erilaisia projektikohtaisia asetuksia, joiden säätöihin yrityksellä on olemassa tietyt oletusarvot. Asetusten säätäminen käsin on pitkäveteistä ja tarkkuuttaa vaativaa, joten tämän vaiheen automatisoinnista olisi hyötyä koordinaattorin ja kääntäjän työssä. Osin tämä on jo aiemmin toteutettu AutoHotkey-ohjelmointikielellä kirjoitetun skriptin avulla, mutta uusia työkaluja hyödyntämällä on mahdollista laajentaa läpikäytävien asetusten kirjoa ja toteuttaa myös aiemmin tehty automaatio paremmin.

Opinnäytetyö perustuu valitun tapauksen toteuttamiseen käytettyjen työkalujen avulla ohjelmistorobotiikan teoriaan tukeutuen, joten tutkimusotetta voidaan pitää konstruktiivisena. Analysoimalla toteutettua tapausta on tarkoitus löytää aiheeseen liittyviä yleistyksiä, tyypillisiä piirteitä sekä syy-seuraussuhteita. Lähdeaineistona on käytetty ohjelmistorobotiikkaa käsittelevää kirjallisuutta sekä työkalujen sähköisiä ohjeita ja dokumentaatiota.

Työ etenee siten, että aluksi tutustutaan ohjelmistorobotiikkaan keinona toteuttaa automatisointi: mitä ohjelmistorobotiikka oikeastaan on, miten se eroaa muista automatisoinnin tavoista, mitä hyötyjä sillä voidaan saavuttaa ja mitä haasteita sen toteuttamiseen liittyy? Tämän jälkeen tarkastellaan, mitä menetelmiä ja työkaluja on valikoitu työn toteuttamiseksi ja miksi. Kun nämä on esitelty, on vuorossa robotin suunnitteluvaihe, johon sisältyy Trados Studion esittely, kirjallinen kuvaus robotin toiminnasta sekä mahdollisten ongelmatilanteiden erittely. Sitten on vuorossa itse toteutusvaihe, jossa kerrotaan robotin rakentamisesta, paketoinnista ja käytöstä. Samalla tuodaan esille ne haasteet, joita robotin toteuttamiseen on sisältynyt. Lopuksi on tarkoitus arvioida työn tulosta: oliko lopputulos sellainen kuin toivottiin, mitä asioita voisi kehittää edelleen ja onko käytettyjä työkaluja ja menetelmiä mahdollista hyödyntää laajemmin?

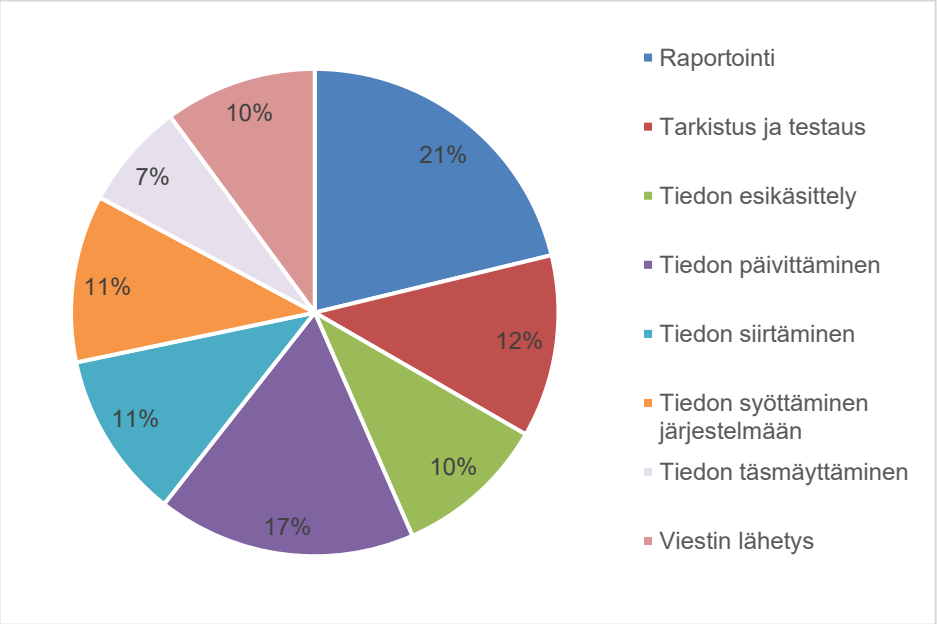
2 Ohjelmistorobotiikka automatisoinnin keinona

2.1 Ohjelmistorobotiikka

Ohjelmistorobotiikka (englanniksi robotic process automation eli RPA) perustuu ihmisen toimintaa jäljitteleviin ohjelmiin, jotka vuorovaikuttamalla muiden ohjelmien kanssa suorittavat tiettyjä sääntöpohjaisia tehtäviä. Nämä tehtävät sisältävät usein näppäilyä, klikkailua ja datan lukemista käytettävästä sovelluksesta. Monista muista automaation keinoista poiketen, joissa ohjelmistojen välinen vuorovaikutus tapahtuu käyttäjältä piilossa olevan rajapinnan kautta, ohjelmistorobotti suorittaa toimintonsa hyödyntämällä samaa käyttöliittymää kuin ihminenkin. Yleisesti ottaen sellaiset tehtävät soveltuvat parhaiten automatisoinnin kohteiksi, joissa on paljon toistuvia, aikaa vieviä vaiheita, joiden suorittaminen perustuu selkeästi määriteltyihin loogisiin sääntöihin. Myös tehtävät, joiden tuottavuus on huono tai jotka sisältävät monia vaiheita useamman ihmisen kesken, ovat hyviä automaation kohteita. (Tripathi 2018, 7-10.)

Valtioneuvoston kanslian vuonna 2018 julkaisema tutkimus (Kääriäinen & al 2018, 9-11) kertoo, minkälaisiin kohteisiin ohjelmistorobotiikkaa käytännössä sovelletaan. Otantaan sisältyi 32 Suomessa toimivaa yritystä yksityiseltä ja julkiselta sektorilta sekä yhteensä 878 käyttötapausta. Näiden perusteella yleisimmät sovelluskohteet liittyivät raporttien ja yhteenvetojen kokoamiseen (21 %), tiedon päivittämiseen (17 %) sekä tarkistuksiin ja testauksiin (12 %). Muita merkittäviä sovelluskohteita olivat tiedon esikäsittely (10 %), siirtely (11 %) ja syöttäminen järjestelmään (11 %), tietolähteiden täsmäyttäminen (7 %) sekä erilaisten viestien lähetys (10 %). Suurin osa soveltamisen kohteista liittyi yrityksen tukitoimintoihin (62 %), jotka jaoteltiin yrityksen tieto-, henkilöstö- ja taloushallintoon, master-datan hallinnointiin, johdon raportointiin sekä muihin tukitoimintoihin. Ydinprosesseihin (38 %) luettiin hankinta, sidosryhmäprosessit sekä myynti-tilaus-toimitusketju, joka oli prosessityypeittäin tarkasteltuna suurin yksittäinen soveltamisen kategoria (29 %). (Kääriäinen & al 2018, 9-11.)

Aineiston pohjalta voidaan siis sanoa, että ohjelmistorobotiikan soveltaminen on keskittynyt viime aikoihin asti suhteellisen yksinkertaisiin, sääntöpohjaisiin tehtäviin. Kuvioon 1 on koottu edellä mainitut tehtävät osuuksineen. Asiat saattavat lähivuosina muuttua, kun robotteihin pyritään integroimaan tekoälyominaisuuksia. Esimerkiksi koneoppiminen on yleistynyt laskentakapasiteetin määrän lisääntyessä sekä parempien laskentamallien kehittyessä. (Kääriäinen & al 2018, 33.) Tekoälyn avulla robotit voivat suorittaa perinteisten, tarkkaan rajattujen prosessien ohella monimutkaisempia tehtäviä, kuten tunnistaa kuvia, prosessoida luonnollista kieltä ja tehdä dataan perustuvia analyyseja ja päätöksiä (Tripathi 2018, 10-11). Ohjelmistorobotiikka tulee myös kehittymään käyttäjäystävällisemmäksi erilaisten graafisten käyttöliittymien myötä, jotka saattavat kaupallisten sovellusten ohella yleistyä myös avoimen lähdekoodin ratkaisuissa (Kääriäinen & al 2018, 33). Odotettavaa onkin, että ohjelmistorobotiikan käyttö tulee helpottumaan ja lisääntymään lähitulevaisuudessa.



Kuvio 1. Ohjelmistorobotiikan käyttökohteet Suomessa vuonna 2018.

2.2 Mahdolliset hyödyt ja haitat

Ohjelmistorobotiikalla on Tripathin (2018, 10-11, 13) mukaan tiettyjä etuja perinteisempiin automatisoinnin keinoihin verrattuna. Ensinnäkin se perustuu samoihin vaiheisiin, joita ihmiset tekevät työskennellessään tietokoneella, kuten hiiren klikkauksiin ja näppäimistön käyttöön. Nämä prosessit on usein jo määritelty ja tunnetaan hyvin, mikä helpottaa niiden automatisointia. Koska robotti toimii saman käyttöliittymän kautta kuin ihminenkin, ei ole tarvetta tehdä muutoksia olemassa oleviin järjestelmiin, mikä luonnollisesti vähentää muutoksiin liittyvien riskien määrää. Perinteisiin menetelmiin verrattuna RPA kykenee myös sopeutumaan poikkeuksellisen dynaamisesti muuttuviin olosuhteisiin, kuten erilaisiin poikkeustilanteisiin, joita ohjelma saattaa kohdata. Lisäksi monien RPA-työkalujen graafisen luonteen vuoksi myös henkilöt, joilta puuttuu ohjelmointikokemusta voivat rakentaa robotteja. (Tripathi 2018, 10-11, 13.)

Ohjelmistorobotiikka parantavaa työn laatua. Robotit eivät poikkea niille annetuista ohjeista, mikä johtaa suurempaan säännönmukaisuuteen tehtävien toimeenpanossa. Robotti myös suoriutuu ihmistä tarkemmin rutiininomaisista töistä, mikä vähentää erilaisten virheiden määrää. (Tripathi 2018, 12-13; Pratt 2021.) Tämä voi johtaa laadukkaampaan dataan, parempaan analytiikkaan ja sitä kautta parempaan päätöksentekoon. Vaikka virheitä tapahtuisi, niitä on helpompi ja nopeampi jäljittää kuin jos työ olisi tehty manuaalisesti, mikä johtuu siitä, että jokainen robotin suorittama vaihe tallennetaan sähköisesti. Virheiden etsinnän ohella robotin tallentamien lokitietojen metadataa ja tageja voidaan hyödyntää liiketoiminnallisten oivallusten löytämiseen ja analytiikassa hyödynnettävän datan keräämiseen. Tällaista dataa voidaan esimerkiksi hyödyntää ennustettaessa tulevaa työmäärää ja robotin kykyä suoriutua tehtävistä ajallaan. (Tripathi 2018, 12-13.)

Manuaalisesti tehtävään työhön verrattuna ohjelmistorobotit myös suoriutuvat tehtävistään erittäin nopeasti (Holmlund 2020; Pratt 2021; Tripathi 2018, 13).

Niitä täytyy joskus tarkoituksella hieman hidastaa, jotta ohjelma, jota ne käyttävät pysyy suoritustahdin mukana. Suuri nopeus johtaa siihen, että suurempi määrä työtehtäviä saadaan tehtyä lyhyemmässä ajassa. Robotit myös sopeutuvat muutoksiin rutiininomaisissa työtehtävissä nopeammin kuin ihmiset, joilta voi kestää muuttaa totuttuja tapoja. Lisäksi robotit ovat paljon kustannustehokkaampia, sillä nopeuden ohella ne voivat työskennellä tauotta kellon ympäri ja suorittaa samanaikaisesti useita tehtäviä. (Tripathi 2018, 12-13.)

Vaikka robotit korvaisivatkin ihmisen joissain töissä, tämä ei kuitenkaan Tripathin (2018, 14) mukaan tee työntekijöistä tarpeettomia, vaan he voivat keskittyä sellaisiin tehtäviin, joita robotit eivät ainakaan vielä osaa tehdä. Tällaisia ovat esimerkiksi työt, jotka vaativat monimutkaista päättelyä ja tunneälykkyyttä, kuten asiakkaiden palvelu. Ohjelmistorobotit voivat jopa lisätä työntekijöiden viihtyvyyttä, kun erilaisten mieltä puuduttavien tehtävien määrä vähenee. Myös asiakastyytyväisyys saattaa parantua, kun tehtävät tulee suoritettua aiempaa nopeammin ja työntekijöillä on enemmän aikaa keskittyä asiakkaiden tarpeisiin. (Tripathi 2018, 14.) Kaikki eivät kuitenkaan näe ohjelmistorobottien vaikutusta työllisyyteen näin positiivisesti. Esimerkiksi Mary K Pratt (2021) on Tripathin kanssa samaa mieltä siitä, että RPA-ratkaisut voivat parhaimmillaan lisätä työntekijöiden ja asiakkaisen tyytyväisyyttä. Hän kuitenkin toteaa, että vaikka robotit eivät aina korvaakaan ihmisiä, on tällainen mahdollisuus olemassa. (Pratt 2021.)

Ohjelmistorobotiikan käyttöönotto ja toteutus ei ole täysin ongelmatonta. Per Holmlundin (2020) mukaan RPA-ratkaisujen kehittäminen ja ylläpito voi olla työlästä ja kallista, sillä ne täytyy aina sovittaa työympäristöönsä. Niinpä pienetkin muutokset järjestelmiin voivat haitata robotin toimintaa. Huonosti suunniteltu ja toteutettu robotti voi myös monistaa tehtyjen virheiden määrän ja pahentaa niiden seuraamuksia, sillä ihmisestä poiketen robotti ei tajua tekevänsä niitä, jos ohjelma ei ota kaikkia tilanteita huomioon. Lisäksi voidaan pohtia, onko ohjelmistorobotiikka aina kestävä ratkaisu pitkällä aikavälillä: se voi

peittää alleen tarpeen tehdä kokonaisvaltaisempia uudistuksia prosessien tehostamiseksi. (Holmlund 2020.)

Haasteita voi myös tuottaa ajan myötä paisuva robottikokoelma, jonka hallinta ja ylläpito saattavat käydä hankalaksi ja kalliiksi. Lisääntyvä monimutkaisuus ilman robottien asianmukaista dokumentaatiota ja hallintaa voi myös tehdä yritystoiminnan kehittämisestä vaikeampaa jatkossa. (Pratt 2021.) Lisäksi ilman tekoälyratkaisuja ohjelmistorobotiikka soveltuu suhteellisen yksinkertaisten työtehtävien automaatioon (Kääriäinen & al 2018, 8), joten paljon vaihtelua ja monimutkaista päättelyä sisältävät prosessit ovat haastavia kohteita roboteille. Taulukkoon 1 on koottu tässä luvussa esille nostettuja hyötyjä ja haittoja, joita ohjelmistorobotiikkaan voi liittyä.

Taulukko 1. Ohjelmistorobotiikan mahdolliset hyödyt ja haitat.

HYÖDYT	HAITAT
Nopeus	Työläs/kallis kehittää ja ylläpitää
Kustannustehokkuus	Mahdollisten virheiden pahentaminen
Sovitettavuus	Kestävyys pitkällä aikavälillä
Laadun paraneminen	Rajallinen soveltuvuus ilman tekoälyä
Säännönmukaisuus	Hallinnalliset haasteet robottien määrän lisääntyessä
Virheiden väheneminen	
Virheiden jäljitettävyys	
Asiakastyytyväisyyden paraneminen	
Työntekijöiden viihtyvyyden lisääntyminen	
Graafiset työkalut	
Automatisoitava prosessi usein tunnetaan	

3 Menetelmät ja työkalut

3.1 Robot Framework automaatioalustana

Erilaisten kaupallisten ja ilmaisten työkalujen joukosta robottien rakentamiseen on valikoitunut Robot Framework. Se on Python-ohjelmointikielen päälle rakennettu avoimen koodin automaatioalusta, jonka avulla on mahdollista toteuttaa testiautomaatioita ja RPA-ratkaisuja. Sillä on selkeästi emokielestään erottuva syntaksi, joka on pyritty tekemään yksinkertaiseksi ja helppolukuiseksi. (Robot Framework 2022b.) Pythonin ohella Robot Framework toimii myös Java- ja .NET -alustoilla hyödyntämällä emokielestä kehitettyjä Jython- ja IronPython-implementaatioita (Robot Framework 2022a). Alustan kehitystä rahoittaa Robot Framework -säätiö, johon kuuluu useita eri yrityksiä ja organisaatioita. Säätiö toimii Suomesta käsin ja sillä onkin merkittäviä suomalaisyrityksiä tukijoina, esimerkiksi OP, Nordea ja Digia. (Robot Framework 2022b.)

Kuten seuraavan sivun kuvasta 1 voi nähdä, on Robot Frameworkin koodi pilkottu toiminnoiltaan erilaisiin osiin, joista yleisimpiin lukeutuvat Settings, Tasks, Variables ja Keywords. Settings voi sisältää muun muuassa robotin dokumentaation (Documentation), käytettävät Python-kirjastot (Library) sekä resurssitiedostot (Resources), jotka ovat Robot Frameworkin syntaksiin perustuvia tukitiedostoja. Variables-nimen alle voi laittaa sellaiset muuttujat, jotka ovat rajaukseltaan globaaleja. Muuttujat merkitään aaltosulkeilla {}. Yleisin niistä on $-alkuinen skalaarimuuttuja, joka voi edustaa mitä tahansa arvoa. Tasks sisältää robotin suorittamat toimenpiteet, joita voi olla yksi tai useampia. Nämä puolestaan voivat koostua yhdestä tai useammasta ehtolauseesta, silmukasta tai avainsanasta.

```
*** Settings ***
Documentation     Template robot main suite.
Library           RPA.Windows
Resource          CustomKeywords.resource

*** Tasks ***
Use Case 1
    Do something

Use Case 2
    Do something else

*** Variables ***
${variable}=      1234
${variable2}=     Some value

*** Keywords ***
Do something
    Log    Hello There!

Do something else
    Log    How are you?
```

Kuva 1. Esimerkki Robot Frameworkin syntaksista.

Avainsanat (Keywords) vastaavat toiminnallisuudeltaan funktioita, mutta ovat syntaksiltaan yksinkertaisempia ja niiden nimeämisessä voidaan käyttää kieltä luonnollisemmin. Funktioiden tavoin avainsanat voivat ottaa vastaan argumentteja ja palauttaa arvoja. Robot Framework ja sen kirjastot sisältävät paljon valmiiksi tehtyjä avainsanoja eri tarkoituksiin, mutta kehittäjän on mahdollista luoda myös omia avainsanojaan. Mikäli nämä perustuvat Robot Frameworkin syntaksiin, ne voidaan laittaa omaan Keywords-osioonsa. Avainsanoja voidaan määritellä suoraan myös Pythonilla. Tämä voi tulla tarpeeseen, jos avainsanoihin halutaan toimintoja, joita ei Robot Frameworkin kirjastoista tavallisesti löydy. Nämä lisätään erilliseen tiedostoon.

Robot Frameworkin etuihin kuuluu sen avoimeen lähdekoodiin perustuva toteutus. Sillä on myös aktiivinen tukiyhteisö, monia kirjastoja kehitetään edelleen. Se sisältää jo valmiiksi lukuisia kirjastoja erilaisiin tarkoituksiin, kuten tiedostojen lukemiseen sekä sovellusten ja selainten käyttöön. Projektin toteutusta ajatellen Robot Framework sisältää esimerkiksi useamman

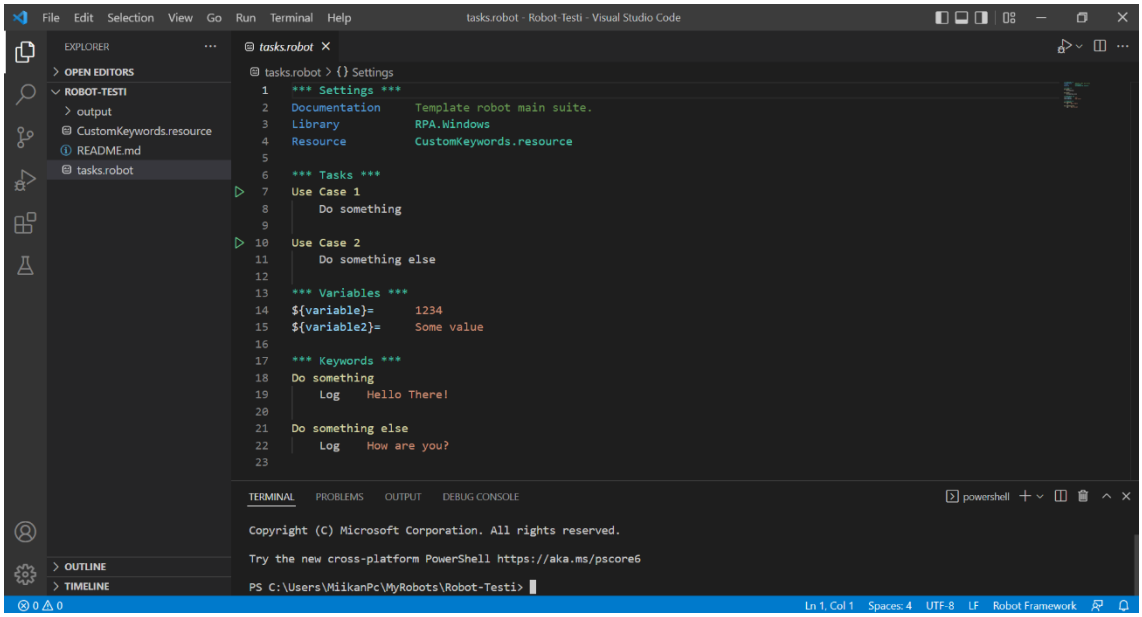
työpöytäsovellusten automaatioon tarkoitetun kirjaston, mistä valita. (Robot Framework 2022b.) Robot Framework on alustasta ja sovelluksesta riippumaton, mikä mahdollistaa laajan kirjon erilaisia sovelluskohteita. Sen avainsanoihin perustuva toteutus on luettavaa ja ymmärrettävää ja mahdollistaa robotin helpon räätälöinnin kehittäjän omiin tarpeisiin. Ajon aikana tulostetut raportit ja lokitiedostot puolestaan helpottavat kehitystyötä, kun viat ovat helposti paikannettavissa ja suoritusnopeus mitattavissa. (Robot Framework 2022c.) Haittapuolena voisi mainita sen, ettei Robot Framework joistakin muista RPA-työkaluista poiketen sisällä graafista kehitystyökalua, mikä rajaa sen kehittäjäkuntaa. Vaikka syntaksi on pyritty tekemään helppolukuiseksi ja ymmärrettäväksi, vaati sen nopea omaksuminen ymmärrystä ohjelmoinnin periaatteista.

Yhtenä etuna AutoHotkey-ohjelmointikieleen nähden voidaan mainita Robot Frameworkin laajempi sovellettavuus ja parempi sopivuus ohjelmistorobotiikan näkökulmasta: erilaisia keinoja kommunikoida käyttöliittymän kanssa on enemmän ja vuorovaikutus näiden välillä on saumattomampaa. Myös kehitystyö on helpompaa tulostettujen lokitiedostojen ansiosta, mikä mahdollistaa nopean virheiden korjaamisen. Jos ohjelmistorobotiikkaa tullaan soveltamaan yrityksen prosesseissa laajemmin tulevaisuudessa, edellä mainitut asiat edesauttavat sen käyttöönottoa huomattavasti.

3.2 Visual Studio Code kehitysympäristönä

Kehitysympäristönä robotin rakentamisessa toimii Visual Studio Code (myöhemmin VS Code). Se on ilmainen koodin muokkaamiseen tarkoitettu tekstieditori tietokoneelle, joka toimii useilla eri alustoilla. VS Code ei sisällä valmiiksi kaikkia ominaisuuksia, joita integroituun kehitysympäristöön (Integrated development environment eli IDE) kuuluu. Sen lukuisat laajennukset mahdollistavat kuitenkin monien ohjelmointikielien käytön sekä kehitysympäristön konfiguroinnin tarpeiden mukaan. (Williams 2021.) Editorin monipuolinen laajennusympäristö onkin yksi tekijä, mikä suosii VS Coden

käyttöä. Se on myös keveytensä ansioista soveltuvampi pieniin kehitysprojekteihin, kuin raskaampi IDE. Kuten kuvasta 2 voi nähdä, on editorin käyttöliittymä varsin suoraviivainen: siihen kuuluu editorinäkymä, jossa koodin muokkaus tapahtuu, alaosan terminaali komentoja ja viestejä varten, sekä vasemman reunan sivupalkki, josta hallitaan erilaisia toimintoja.



Kuva 2. Visual Studio Coden editorinäkymä.

VS Code ei sisällä valmista tukea Pythonille ja Robot Frameworkille, joten muutaman lisäosan asentaminen on tarpeen. Microsoftin Python-laajennukseen kuuluu useita Pythonilla tehtävää kehitystyötä helpottavia ominaisuuksia, kuten koodin tarkistus, debuggaus sekä reformatointi ja refaktorointi. (Microsoft 2022b.) Tuki Robot Frameworkin syntaksille saadaan puolestaan Robocorpin Robot Framework Language Server -laajennuksesta. Se perustuu Language Server -protokollaan, joka sallii valitun ohjelmointikielen tuen asiakkaan (Client) käyttämässä kehitysympäristössä kielipalvelimen kautta. Laajennus tukee muun muuassa koodin täydentämistä, infoapua, validointia sekä syntaksin korostamista. (Robocorp 2021a.)

3.3 Accessibility Insights aputyökaluna

Jotta ohjelmistorobotti voisi vuorovaikuttaa sovelluksen kanssa täytyy olla keino, jonka avulla robotti ymmärtää, mitä sovelluksen osaa tulee kulloinkin käyttää. Kyseessä voi esimerkiksi olla painike, valintaruutu, vetovalikko, valintanappi tai tekstikenttä. Yksi tapa on hyödyntää pikanäppäimiä, mutta näitä ei välttämättä ole kaikille toiminnoille saatavilla. On myös mahdollista tunnistaa kuvia ja tekstiä näytöltä, mutta näiden hyödyntämisessä on tiettyjä rajoitteita, sillä tunnistuksen tarkkuus voi riippua monista tekijöistä, esimerkiksi resoluutiosta. (Palsamäki 2021.)

Windowsin työpöytäsovellusten automatisoinnille keskeinen apu onkin hyödyntää Microsoft UI Automation -saavutettavuusalustaa. Se on ohjelmointirajapinta, jonka avulla on mahdollista saada informaatiota sovelluksen käyttöliittymän elementeistä. UI Automation toimii erilaisten Windowsin käyttöliittymäalustojen kanssa (esimerkiksi Windows Presentation Foundation eli WPF). Alustaa voidaan Microsoftin mukaan käyttää takaamaan saavutettavuus sellaisille ihmisille, joilla on kuuloon, näköön tai tuntoon liittyviä ongelmia sekä ajamaan testiautomaatioon liittyviä skriptejä. Alustan avulla saatavaa informaatiota on mahdollista hyödyntää sellaiseen vuorovaikutukseen käyttöliittymän kanssa, joka ei tapahdu tavallisen syötteen kautta. (Microsoft 2020a.) Niinpä se soveltuu hyvin myös RPA-kehittäjän avuksi, kun etsitään keinoja manipuloida työpöytäsovellusten elementtejä.

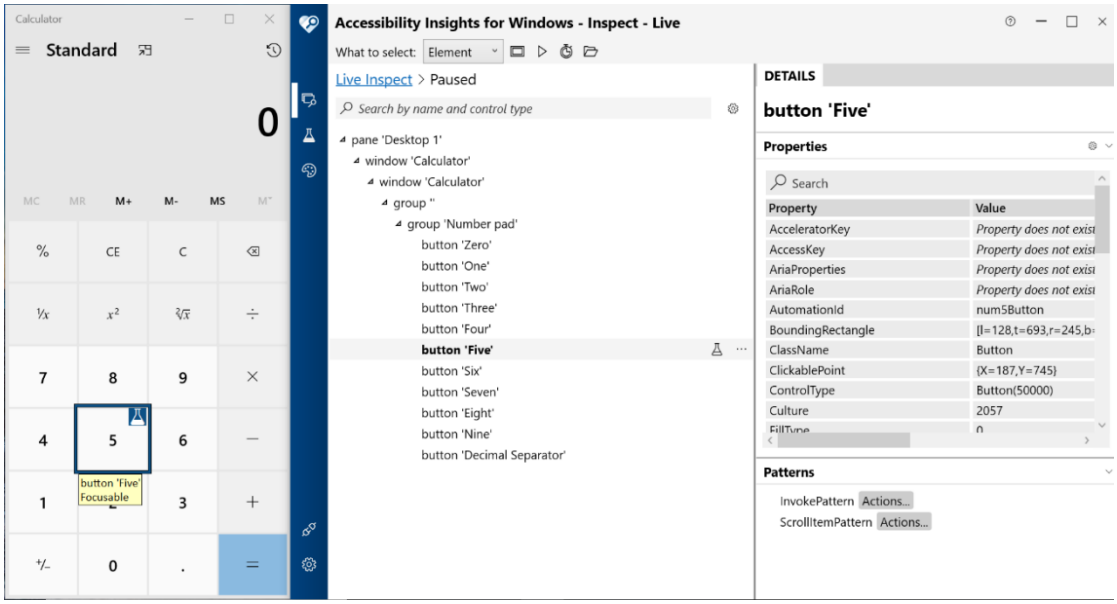
UI Automation esittää käyttöliittymän elementit objekteina, jotka perustuvat IUIAutomationElement-rajapintaan. Elementit ovat puumallisessa hierarkiassa, jossa työpöytä (desktop) on juurena. Jokaisella niistä on kontrollityyppi, joka kertoo, minkälaisesta elementistä on kyse ja asettaa tiettyjä ehtoja niiden toiminnalle. Esimerkiksi painikkeet, valintaruudut ja ikkunat muodostavat omat tyyppinsä. (Microsoft 2021.)

Robotti kykenee tunnistamaan elementtejä ja vuorovaikuttamaan niiden kanssa lukemalla ja tarvittaessa manipuloimalla niiden ominaisuuksien arvoja. Ominaisuuksia on kahdenlaisia: automation element- ja control pattern -ominaisuuksia. Ensin mainitut ovat yleisiä, kontrollityypistä riippumattomia ominaisuuksia, kuten Name, Acceleratorkey ja Classname. Niiden arvot ovat usein staattisia eli muuttumattomia. (Microsoft 2019.) Control pattern on elementin osa, joka kertoo, että millaisia toimintoja kyseiseen elementtiin kuuluu ja mahdollistaa näiden toimintojen manipuloinnin. Esimerkiksi taulukkorakenteella on yleensä grid control pattern, joka kertoo kuinka monta riviä ja saraketta taulukkoon kuuluu ja mahdollistaa taulukon alkioiden haun. Jokainen control pattern edustaa erilaista toimintoa, joten niitä voi kuulua yhteen elementtiin useampi. (Microsoft 2020b.) Jokaisella on myös niille tyypillisiä ominaisuuksia. Grid control pattern toteuttaa esimerkiksi columncount ja rowcount -ominaisuudet. Automation element -ominaisuuksista poiketen control pattern -ominaisuudet ovat luonteeltaan yleensä dynaamisia. (Microsoft 2019.)

Jotta informaation elementeistä saa esille, on käytettävä siihen soveltuvaa työkalua. Tällainen on Microsoftin Acessibility Insights, joka on tarkoitettu kehittäjille korjaamaan elementtien saavutettavuuteen liittyviä asioita. Se sisältää FastPass -toiminnon, jonka avulla pystyy löytämään erilaisia saavutettavuuteen liittyviä ongelmia, kuten pääseekö elementtiin käsiksi näppäimistön avulla, onko tabulointi tehty oikein ja niin edelleen. Sovelluksen Live Inspect -toiminnon avulla on puolestaan mahdollista tarkastella elementtien ominaisuuksia ja asemaa hierarkiassa. (Microsoft 2022a.)

RPA-kehittäjän näkökulmasta sovelluksen hyödyllisin ominaisuus on juurikin Live Inspect -toiminto. Se toimii siten, että käyttäjä osoittaa sovelluksen elementtiä, jota haluaa tarkastella. Elementin ympärille piirtyvät siniset raidat kuvassa 3 kertovat, että elementti on fokuksessa, jolloin informaatio siitä ilmestyy käyttöliittymään. Jos käyttäjä siirtää hiirtään muualle, fokus vaihtuu, ellei tämä ole lukinnut tarkastelukohdetta yläreunan pause-näppäimen avulla.

Live Inspect jakautuu kolmeen osioon: ruudun vasemmanpuoleisessa osiossa näkyy elementin asema hierarkiassa. Properties-osiossa näkyvät automation element -ominaisuudet. Alalaidassa sijaitsevassa Patterns-osiossa puolestaan näkyvät elementin tukemat control pattern -ominaisuudet.



Kuva 3. Laskimen elementti Acessibility Insights -työkalulla tarkisteltuna.

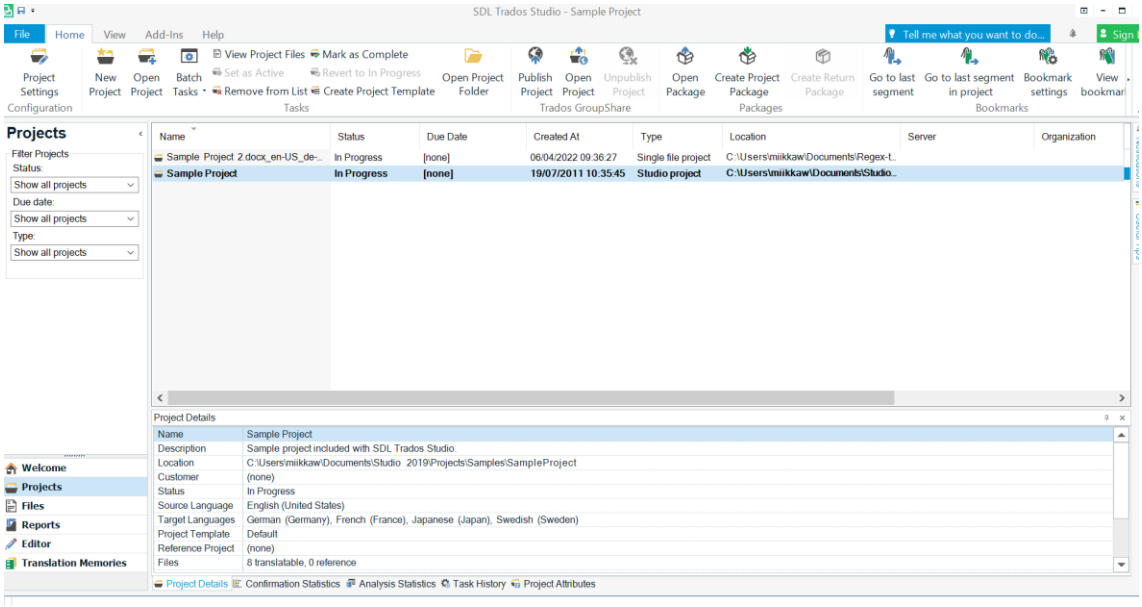
4 Ohjelmistorobotin suunnittelu ja toteutus

4.1 Trados Studio automatisoinnin kohteena

SDL Trados Studio (myöhemmin Trados Studio) on tietokoneavusteinen käännöstyökalu (englanniksi Computer Assisted Translation eli CAT), jonka tarkoituksena on auttaa kääntäjiä työskentelemään nopeammin ja tehokkaammin käännöstöidensä parissa. Trados Studiolla on tuotesivun mukaan noin 270000 käyttäjää maailmassa, mikä tekee siitä suosituimman CAT-työkalun. (RWS Group 2022.) Toimeksiantajalla käytössä olevan, vuoden 2019 version on kehittänyt SDL plc, joka oli brittiläinen käännöspalveluihin ja -ohjelmiin erikoistunut yritys. Nykyään se on tuotteineen osa suurempaa RWS Group -yritystä. (RWS Group 2020.) Koska toimeksianto koskee Trados Studion projektiasetusten automatisointia, on syytä esitellä lyhyesti kyseinen käännöstyökalu tehtävänannon ja ohjelmistorobotiikan näkökulmasta.

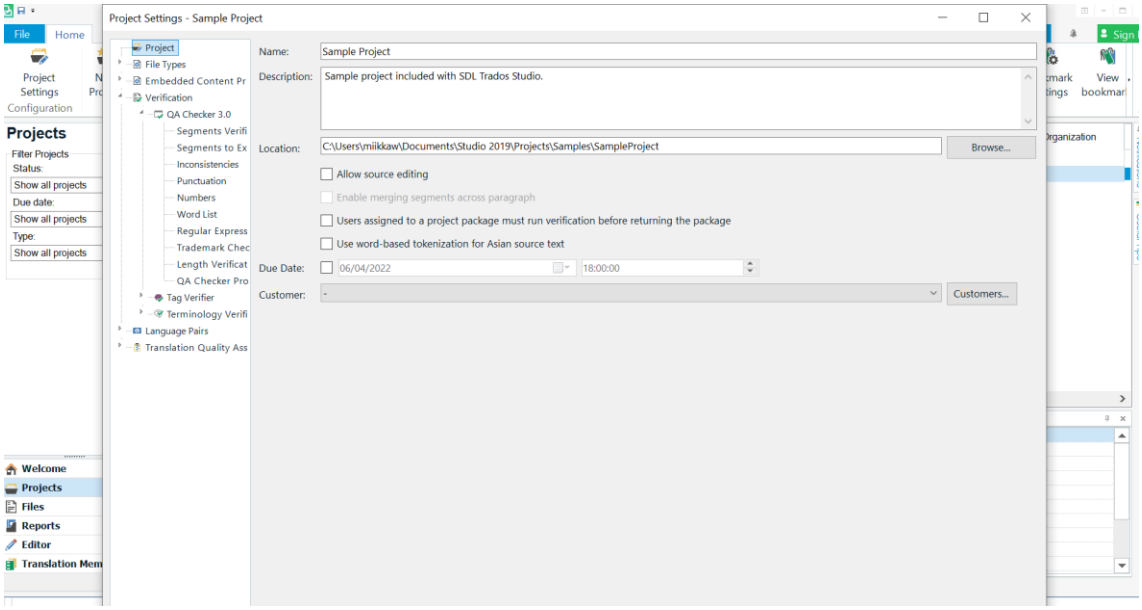
Trados Studio avautuu oletusarvoisesti Home-välilehdeltä Projects-näkymästä, joka sisältää keskellä listan kaikista Trados Studiossa olevista käännösprojekteista, kuten seuraavan sivun kuvasta 4 voi nähdä. Valittu projekti on näkymässä korostettu sinisellä yliviivauksella. Muut näkymät ovat avattavissa vasemmasta alakulmasta. Kohdassa Files näkyvät projektin sisältämät käännöstiedostot sekä muut dokumentit, jotka avautuvat klikattaessa Editor-näkymään, missä itse tiedoston kääntäminen tapahtuu. Reports sisältää erilaisia Trados Studion analyysityökaluilla projektista tuotettuja raportteja, jotka voivat liittyä muun muuassa projektin esikääntämiseen ja tarkistusprosessiin. Translation Memories -osiossa voidaan muokata ja päivittää käännösmuisteihin liittyvää dataa. Käännösmuisti on tietokanta, joka sisältää aiemmin käännettyjä yhden kieliparin (esimerkiksi suomi – englanti) käännösyksiköitä (translation unit eli TU). Käännösyksiköt koostuvat rinnakkaisista lähde- ja kohdekielen segmenteistä eli yksittäisistä sanoista, lauseista tai virkkeistä sekä niihin

liittyvistä metatiedoista, joita voidaan hyödyntää käännettäessä uusia dokumentteja.



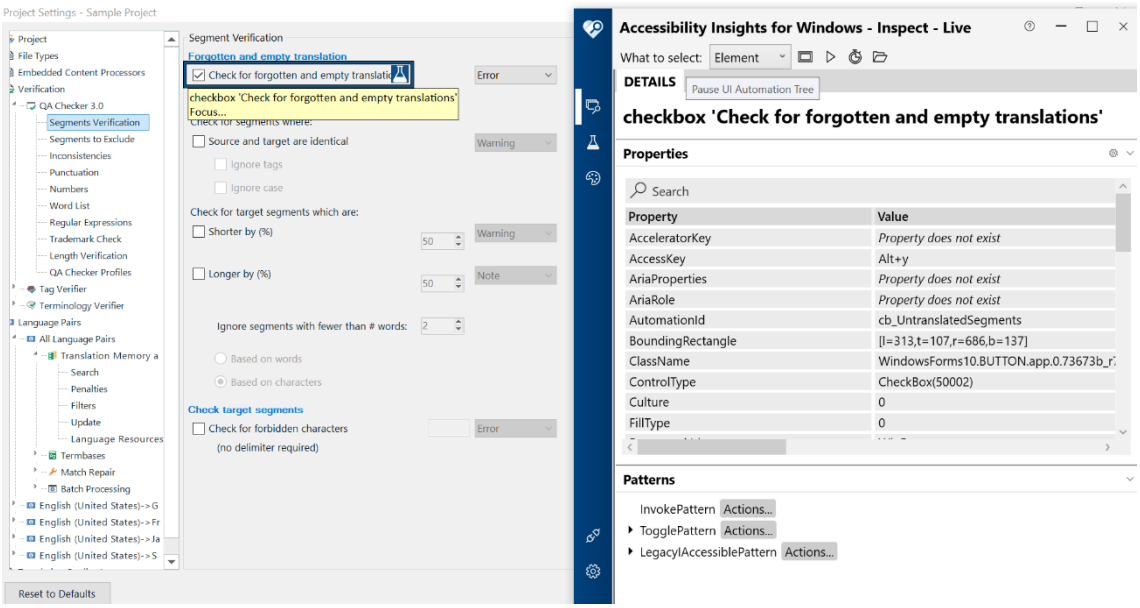
Kuva 4. Projektinäkymä SDL Trados Studiossa.

Projektikohtaisiin asetuksiin pääsee valitsemalla projektinäkymästä oikean projektin ja klikkaamalla vasemmassa yläkulmassa olevaa Project Settings -painiketta. Asetukset jakautuvat kuvassa 5 vasemmalla näkyvään valikkoon ja oikean puolen välilehteen, joka näyttää avatun valikon asetukset. Valikkorakenne on puumallinen: ylätason valikoita painamalla avautuu lisää valikoita, jotka saattavat jakautua edelleen alavalikkoihin. Ylätason valikoista Projects sisältää yleistä tietoa projektista, kuten sen nimen, kuvauksen ja kansiopolun. Projektin tukemista tiedostotyypeistä säädetään File Types -valikon kautta ja Embedded Content Processors sisältää asetuksia, joilla määritetään, miten upotettua sisältöä näytetään ja prosessoidaan. Verification sisältää työkaluja, joiden avulla voidaan tarkistaa, että tekstin kieliasu on määrättyä muotoa. Language Pairs -osiossa ovat projektin sisältämien kieliparien asetukset ja Translation Quality Assessment määrittää ne kriteerit, joilla käännöksen laatua on mahdollista arvioida.



Kuva 5. Projektikohtaisten asetusten valikko Trados Studiossa.

Projektiasetusten säätäminen koostuu pääasiassa erilaisten valintaruutujen ja -painikkeiden painalluksista sekä tekstikenttien ja vetovalikoiden manipuloimisesta. Hyödyntämällä Microsoftin Accessibility Insights -työkalua on mahdollista tarkastella näiden elementtien ominaisuuksia ja löytää sopivia tunnisteita, joiden avulla kertoa ohjelmistorobotille, minkä elementin kanssa tulee vuorovaikuttaa. Kuvan 6 esimerkissä työkalun fokuksessa on asetusten valintaruutu. Insights paljastaa, että ruudulla on uniikki AutomationId-ominaisuus nimeltään cb_UntranslatedSegments. Monet Trados Studion elementeistä sisältävät kyseisen tai vaihtoehtoisen ominaisuuden, jonka avulla ne voi tunnistaa. Tämä mahdollistaa toimintojen automaation ohjelmistorobotiikalla.



Kuva 6. Trados Studion projektiasetusten valintaruudun tarkastelu Accessibility Insights-työkalulla.

4.2 Kuvaus robotin toiminnasta

Edellä mainituista projektiasetuksista automaatio kohdistuu Verification- ja Language Pairs -asetuksiin. Seuraavaksi käydään läpi suunnitellut robotin vaiheet, kun se tarkistaa ja tarvittaessa muuttaa kyseisiä asetuksia. Järjestyksen määrittelyssä on hyödynnetty aiemmin AutoHotkey-skriptin rakentamisessa tehtyä dokumentaatiota. Asetusten tarkemmat säädöt on jätetty pois kuvauksesta ja keskitytty pääpiirteittäin selittämään, mitkä välilehdet kyseisistä asetuksista robotti käy läpi. Ohjelmistorobotin onnistunut suoritus vaatii, että käyttäjä on avannut Trados Studion valmiiksi ja valinnut projektinäkymästä oikean projektin. Tämän jälkeen käyttäjä käynnistää robotin:

1. Robotti avaa Trados Studion ikkunan työpöydällä.
2. Robotti varmistaa, että Home-välilehti on auki ja klikkaa Project Settings -painiketta.
3. Robotti avaa Verification-valikon projektiasetuksista.

4. Robotti avaa alavalikon QA Checker 3.0. Se sisältää erilaisia työkaluja, joiden avulla tekstin kieliasua voidaan tarkkailla. Robotti käy läpi seuraavat valikon alla olevat asetukset:
    4.1. Segments Verification: käännettyjen segmenttien tarkistukseen liittyvät asetukset. Segmentit ovat osia käännettävästä dokumentista, jotka Trados Studio on pilkkonut pienempiin osiin käännöstyön helpottamiseksi.
    4.2. Inconsistencies: tarkistaa tekstin erilaisten epäjohdonmukaisuuksien löytämiseksi, kuten onko sama teksti käännetty usealla eri tavalla.
    4.3. Punctuation: Sisältää erilaisia pilkkusääntöjen tarkistusasetuksia.
    4.4. Numbers: Sisältää numerotarkistuksia liittyen muun muuassa lukuihin, päivämääriin ja mittoihin.
    4.5. Word lists: Voi asettaa sanalistan, joka sisältää sanojen oikean ja väärän muodon. Tarkistaa käännöksen listan mukaisesti.
    4.6. Regular expressions: Sisältää käyttäjän asettamia säännöllisiä lausekkeita (regular expressions), joiden avulla käännetystä tekstistä voidaan löytää vääriä sanoja ja lauserakenteita.
    4.7. Trademark Check: Auttaa tarkistamaan, ovatko tuotemerkkien symbolit siirtyneet halutulla tavalla käännettyyn tekstiin.
    4.8. Length Verification: Tarkistaa onko käännettyjen segmenttien pituus annetuissa rajoissa.
5. Robotti käy läpi Tag Verifier -asetukset, joilla voidaan tarkistaa, ovatko käännettävän tekstin sisältämät tagit siirtyneet onnistuneesti käännettyyn tekstiin.
6. Robotti avaa Language Pairs -asetukset.
7. Robotti avaa All Language Pairs -asetukset, josta voidaan säätää kaikkien projektissa olevien kieliparien asetuksia.
8. Robotti avaa valikon ”Translation Memory and Automated Translation”, josta voi hallita projektin sisältämiä käännösmuisteja.
9. Robotti klikkaa välilehden yläosasta Use-painiketta, mikä avaa pienen valikon. Sieltä robotti valitsee ”Server-based Translation memory.” Avautuu

uusi ikkuna, jossa on listattu toimeksiantajan palvelimella olevia käännösmuisteja.

10. Robotti valitsee ikkunan vasemmassa laidassa olevasta Filter-osiosta projektiin sopivan kirjaston.
11. Robotti klikkaa OK-painiketta, sulkien käännösmuistin valintaikkunan.
12. Robotti käy läpi seuraavat Translation Memory and Automated Translation -valikon alla olevat asetukset:
    12.1. Update: Sisältää käännösmuistin yksiköihin tallentuvia metatietoja.
    12.2. Search: Käännösmuistin hakuun liittyviä asetuksia.
    12.3. Filters: Käännösmuistin käyttöön liittyviä suodattimia.
13. Robotti avaa termikantojen asetukset ja valitsee projektiin oikean termikannan. Ne ovat tietokantoja, jotka sisältävät aihepiiriinsä liittyvää terminologiaa. Niiden avulla voidaan varmistaa, että käännöksessä käytetään asianmukaista termistöä.
14. Robotti avaa Match Repair -asetukset ja tekee tarvittavat muutokset. Asetuksista määritellään, korvataanko käännetyt tekstit lähdetekstiä paremmin vastaavilla käännöksillä.
15. Robotti avaa Batch Process -valikon. ja tekee siellä muutoksia seuraaviin asetuksiin:
    15.1. Analyze Files: Trados Studion analyysityökalun asetuksia.
    15.2. Verify Files: Määrittää näkyvätkö ignored-statuksella merkityt viestit tarkistusraporteissa.
    15.3. Fuzzy Bands: Määrittävät mihin luokkaan käännösmuistista haetut segmenttien käännökset kuuluvat analyysissa.
    15.4. Translation Memory Updates: Määrittää mitkä käännökset lisätään käännösmuistiin ja korvataanko vanhat käännökset uusilla.
16. Robotti avaa pienen ikkunan, missä ilmoittaa käyttäjälle olevansa valmis. Käyttäjä valitsee ikkunasta, haluaako tallentaa asetukset ja painaa Ok.

4.3 Mahdolliset poikkeustilanteet

Ohjelmistorobotti noudattaa täsmällisesti sille koodissa määriteltyjä toimintoja ajon kontekstista riippumatta. Tämä voi aiheuttaa erilaisia poikkeustilanteita, kun robotti ei kykene erilaisista syistä johtuen suorittamaan sille annettua tehtävää kokonaan. Mahdollisten syiden kartoittaminen auttaa robotin rakentamisessa, kun voidaan pohtia keinoja välttää riskit tai ainakin minimoida ne.

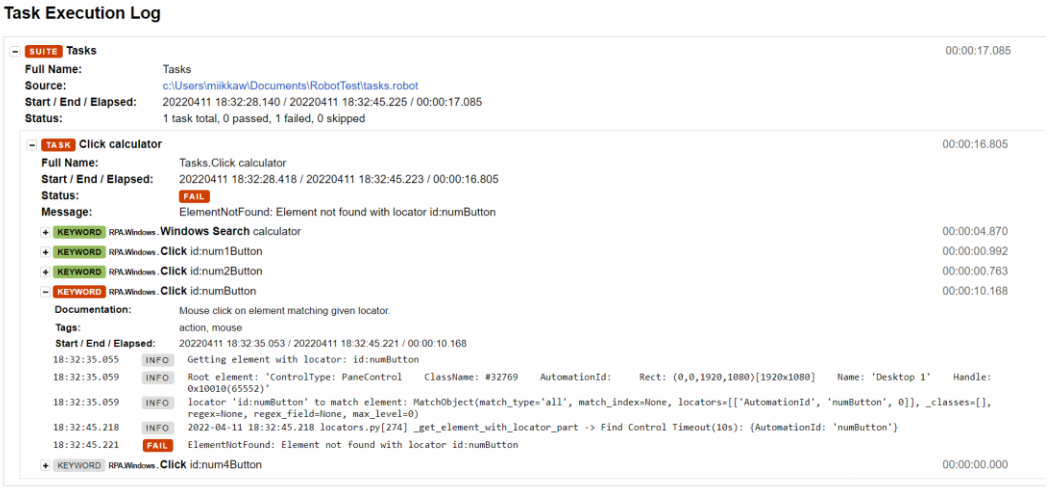
Syy poikkeustilanteisiin voi olla inhimillinen, sillä käyttäjä saattaa toiminnallaan aiheuttaa jotain sellaista, mikä häiritsee robotin suoriutumista. Kuten edellisessä luvussa tuli ilmi, vaatii robotin onnistunut käyttö esivaiheen: Trados Studio on avattava ja oikea projekti valittava, jotta robotti pääsee muokkaamaan asetuksia. Lisäksi on pidettävä huolta, että käyttäjä ei häiritse sen toimintaa kesken ajon. Robotti vuorovaikuttaa sovellusten kanssa saman käyttöliittymän kautta ja osin myös samojen keinojen, kuten hiiren klikkausten avulla. Niinpä käyttäjän samaan aikaan koneella tekemä muu työ voi häiritä pahasti robotin toimintaa. Paras tapa vähentää käyttäjän toiminnasta johtuvia poikkeustilanteita on luoda selkeä ohjeistus siitä, kuinka robottia tulee käyttää ennen ajoa ja sen aikana. Robotin kansioon voidaan luoda tiedosto, joka sisältää ajon vaiheet ja kertoo mitä käyttäjän tulee tehdä niiden onnistumiseksi.

Poikkeustilanteita voivat tietysti aiheuttaa myös erilaiset tekniset ongelmat. Robotti vaatii internet-yhteyden, sillä käännösmuistin valinta vaatii pääsyä yrityksen muistipalvelimelle. Yhteyden hitaus ja varsinkin sen puute aiheuttaa ongelmia, kun valittavaa muistia ei ilmesty sovellukseen. Hitaus voi olla myös ongelma sovelluksen kontekstissa: jos robotin toiminta on nopeampaa kuin sovelluksen, saattaa robotti pyrkiä vuorovaikuttamaan sellaisten elementtien kanssa, jotka eivät ole vielä latautuneet kunnolla käyttöliittymään. Tämä voi johtaa virheellisiin toimintoihin tai estää toiminnan kokonaan, kun kohde-elementtiä ei löydy. Samanlaisiin ongelmiin voi johtaa myös se, mikäli käyttäjällä ja kehittäjällä on eri versio sovelluksesta. Päivitykset voivat nimittäin

tuoda muutoksia käyttöliittymän rakenteeseen, elementtien hierarkiaan ja niiden ominaisuuksien nimistöön.

Teknisiin ongelmiin voidaan vastata eri keinoin. Robotti voidaan määrittää ilmoittamaan käyttäjälle internet-yhteyden puutteesta tai hitaudesta ja keskeyttää ajo, kunnes ongelma on korjattu. Toinen vaihtoehto on jättää tietyn ajan jälkeen käännösmuisti valitsematta ja jatkaa suoritusta. Tällöin muistin valinta jää käyttäjän tehtäväksi. Robotin ja sovelluksen nopeuseroista johtuviin ongelmiin voidaan käyttää hyödynnettävään Robot Framework -kirjastoon sisäänrakennettua mahdollisuutta määrittää avainsanojen suoritukselle aikaraja. Sen avulla on mahdollista säätää, kuinka kauan robotin tulisi yrittää tietyn toiminnan suorittamista, mikä antaa sovellukselle aikaa ladata kaikki elementit näytölle. Eri versioiden yhteensopivuuteen liittyen on ensisijaisesti varmistettava, että käyttäjän ja kehittäjän versiot ovat samat. Mikäli päivitysten myötä sovellukseen tulee muutoksia, on tarkistettava robotin suoritus kyseisessä versiossa ja tehtävä tarvittavat toimenpiteet robotin mukauttamiseksi.

Poikkeustilanteet voivat johtua myös koodin tasolla tehdyistä ratkaisuista. Mikäli virheen takana on pelkkä koodi, on useimmiten helppo paikantaa ongelma. Jos ongelma johtuu koodin ohella erilaisista ajoympäristön piirteistä, voi näiden löytäminen ja korjaaminen olla huomattavasti vaikeampaa. Virheiden etsimisessä auttaa kuitenkin Robot Framework, joka luo jokaisen ajon päätteeksi raportin ja lokitiedoston suorituksesta. Lokitiedostosta voidaan päätellä, missä kohtaa suoritusta virhe on tapahtunut ja mikä sen on aiheuttanut, kuten kuvasta 7 voidaan nähdä.



Kuva 7. Esimerkki Robot Frameworkin luomasta lokitiedostosta.

4.4 Robotin toimintojen koodaaminen

Rakenteeltaan robotin koodi jakautuu päätiedostoon, jossa suoritus tapahtuu, sekä tukitiedostoihin, jotka sisältävät suoritusta tukevia avainsanoja ja muuttujia. Päätiedosto sijaitsee robotin juuressa, kun taas tukitiedostot on laitettu omaan resources-kansioonsa. Koodin hajauttamisen syynä on robotin rakenteen selkeyttäminen. Ilman tukitiedostojen sisältöäkin päätiedostossa on satoja rivejä koodia, joten siirtämällä koodia muihin tiedostoihin voidaan tukea koodin luettavuutta. Robottiin on myös helpompi tehdä muutoksia tulevaisuudessa, kun koodi on rakenteellisesti eritelty. Lisäksi osa avainsanoista on määritelty suoraan Pythonilla, joten nämä pitää joka tapauksessa laittaa erilliseen tiedostoon.

Päätiedosto tasks.robot jakautuu kolmeen osioon. Ensimmäinen näistä on Settings, mihin on dokumentoitu kuvan 8 mukaisesti robotin tarkoitus ja eräät tekniset yksityiskohdat. Sinne on myös lisätty tarpeelliset kirjastot ja resurssitiedostot. Näiden välinen ero on se, että kirjastot on koodattu Pythonilla, kun taas resurssitiedostot sisältävät Robot Frameworkin syntaksia.

```
*** Settings ***
Documentation     ABOUT: This robot checks and corrects settings for SDL Trados Studio Project. These include
...               certain parts of QA checker, tag verifier, translation memory, termbases, match repair and batch settings.
...               Instructions to setup and run robot are found in the readme file in robot folder.
...
...               TECHNICAL: RPA.Windows library has a default locator search depth of 8 from root element.
...               This is to make search more efficient because element tree can be quite big especially in
...               desktop applications. The default search depth can be changed with prefix depth:x. Root element
...               can be set with keywords "Control Window" or "Set Anchor" ("Clear Anchor" sets root back to control
...               window).
Library           Dialogs
Library           RPA.Windows
Library           resources/CustomKeywords.py
Resource          resources/CustomKeywords.resource
Resource          resources/Variables.resource
```

Kuva 8. Robotin Settings osio koostuu dokumentaatiosta, kirjastoista ja resurssitiedostoista.

Tasks-osiossa on itse suoritettava tehtävä. Kuten seuraavan sivun kuvasta 9 voi nähdä, rakentamisessa on päädytty sellaiseen ratkaisuun, että robotti suorittaa vain yhden ison tehtävän, joka on jaettu avainsanojen avulla pienempiin alitehtäviin. Tehtävä on pyritty pilkkomaan siten, että jokainen projektiasetusten erillinen osio vastaisi yhtä avainsanaa. Esimerkiksi avainsana ”Check and correct QA checker punctuation” vastaa QA checkerin asetusten punctuation-osiota, jossa määritellään tarkistuksen pilkkusäännöt. Poikkeuksia tähän säännönmukaisuuteen ovat muun muuassa Trados Studion avaamiseen ja käännösmuistin valintaan liittyvät avainsanat, joissa liikutaan useamman ikkunan läpi.

Avainsanat on myös pyritty tekemään mahdollisimman itsenäisiksi toisistaan. Tämä helpottaa muutoksien tekemistä robotin rakenteeseen, kun alitehtäviä voidaan siirtää, poistaa ja muokata ilman, että täytyy tehdä laajempia toimenpiteitä. Avainsanojen nimeämisessä on pyritty hyödyntämään syntaksin mahdollistamaa selkeyttä, mikä auttaa ymmärtämään niiden merkitystä. Jokaisen avainsanan toiminnon tulisi käydä sen nimestä ilmi. [Teardown]-etupääte voidaan liittää avainsanaan, joka suoritetaan riippumatta siitä, onnistuiko ajo kokonaan vai ei.

```
*** Tasks ***
Checking and selecting correct settings for the Trados Studio project
    Set Wait Time    ${global_wait_time}    #Set global wait time for RPA.Windows action keywords
    Open Studio and QA checker Settings
    Check and correct QA checker segments verification
    Check and correct QA checker inconsistencies
    Check and correct QA checker punctuation
    Check and correct QA checker numbers
    Check and correct QA checker word list
    Check and correct QA checker regular expressions
    Check and correct QA checker trademark check
    Check and correct QA checker length verification
    Check and correct tag verifier
    Select translation memory
    Select translation memory update values
    Check and correct translation memory search
    Check and delete translation memory filters
    Check and correct termbases settings
    Check and correct match repair
    Check and correct batch processing analyze files
    Check and correct batch processing verify files
    Check and correct batch processing fuzzy bands
    Check and correct batch processing translation memory updates
    Show success dialog
    [Teardown]    Run Keyword If    ${success}==False    Show failure dialog    #Shows failure dialog if robot fails
```

Kuva 9. Robotin Tasks osio. Päätehtävä on pilkottu pienemmiksi alitehtäviksi.

Projektiasetusten säätämisessä on paljon toistoa, sillä suurin osa asetuksista koostuu valintaruuduista- ja napeista, vetovalikoista ja arvokentistä. Näille onkin järkevä luoda omat avainsanansa, joita voi sitten kutsua yhä uudelleen. Esimerkiksi kuvan 10 ”Check and correct checkbox” on avainsana, joka tarkistaa ja tarvittaessa korjaa valintaruudun arvon. Se ottaa argumenttina arvon, joka sillä pitäisi olla sekä automation element -ominaisuuden, jonka avulla robotti tunnistaa oikean elementin ruudulta. Samalla periaatteella toimivat myös valintanappien, vetovalikoiden ja arvokenttien vastaavat avainsanat. Tämän rakenteellisen säännönmukaisuuden vuoksi suurin osa päätehtävän alitehtävistä on myös varsin samankaltaisia: erilaisten valikoiden ja nappien tarkistusta ja korjaamista.

```
Check and correct QA checker punctuation
#Opens punctuation tab from QA checker settings and makes corrections if necessary
    RPA.Windows.Click    name:Punctuation depth:10
    Set Anchor    ${QA_checker_anchor}
    Check And Correct Checkbox    ${checkbox_uncheck}    cb_PunctuationDifferences    #Check that source and target end with the same punc
    Check And Correct Checkbox    ${checkbox_uncheck}    cb_SpanishPunctuation    #Check for Spanish punctuation
    Check And Correct Checkbox    ${checkbox_check}    cb_PunctuationSpace    #Check for unintentional spaces before:
    Check And Correct Checkbox    ${checkbox_uncheck}    cb_PunctuationSpacesFrench    #French compliant check
    Check And Correct Checkbox    ${checkbox_check}    cb_MultipleSpaces    #Check for multiple spaces
    Check And Correct Checkbox    ${checkbox_check}    cb_MultipleDotsA    #Check for multiple dots
    Check And Correct Checkbox    ${checkbox_check}    cb_Ignore3Dots    #Ignore ellipsis dots
    Check And Correct Checkbox    ${checkbox_check}    cb_ExtraEndSpace    #Check for extra space at the end of target segment
    Check And Correct Checkbox    ${checkbox_uncheck}    cb_InitialCapsCheck    #Check capitalization of initial letters
    Check And Correct Checkbox    ${checkbox_uncheck}    cb_GlobalCapsCheck    #Check consistency of global capitalization
    Check And Correct Checkbox    ${checkbox_uncheck}    cb_BracketsCheck    #Check brackets
    Check and correct list selection    ${list_warning}    sev_PunctuationSpace    #Check for unintentional spaces before
    Check and correct list selection    ${list_note}    sev_MultipleSpaces    #Check for multiple spaces
    Check and correct list selection    ${list_note}    sev_MultipleDots    #Check for multiple dots
    Check and correct list selection    ${list_warning}    sev_ExtraEndSpace    #Check for extra space at the end of target segment
    Check and correct value    .,:;?!)]    tb_PunctuationSpaceCharacters    #Check for unintentional spaces before
    Clear Anchor
```

Kuva 10. QA Checkerin punctuation asetusten säätäminen koostuu samojen avainsanojen käytöstä yhä uudelleen.

Trados Studion automaatiossa on hyödynnetty työpöytäsovelluksille suunnattua RPA.Windows-kirjastoa, jonka etuna muihin vastaaviin verrattuna on sen nopeus ja monipuoliset keinot päästä käsiksi käyttöliittymän elementteihin. Eräs kirjastolle ominainen piirre on kuvassa 10 näkyvä ankkurin asettaminen (Set Anchor) ja poistaminen (Clear Anchor). Näiden avulla voidaan määrittää, mikä elementti hierarkiassa toimii juurena, josta käsin muita elementtejä etsitään. Normaalisti Windowsin juurena toimii työpöytä (desktop) tai aktiivinen ikkuna. Ankkurin asettamalla varmistetaan, ettei robotin tarvitse käydä läpi kaikkia hierarkioita löytääkseen oikean elementin. Tämä on hyödyllistä, sillä sovelluksissa voi olla useita samannimisiä tai tyyppisiä elementtejä. Rajaamalla määrää varmistetaan, että valinta kohdistuu oikeaan elementtiin. Hierarkiat ovat työpöytäsovelluksissa usein myös syviä, niitä rajoittamalla voidaan löytää elementit nopeammin.

Eräs elementtien etsintään liittyvä piirre RPA.Windows-kirjastossa on se, että oletusarvoinen etsintäsyvyys juurielementistä on 8. Tämä tuottaa hankaluuksia Trados Studion kohdalla, sillä hierarkiat voivat olla paljon tätä syvempiä. Yksi ratkaisu on siirtää ankkurin paikkaa lähemmäs kohde-elementtiä. Toinen tapa on säätää elementtejä kohdistavien avainsanojen depth-ominaisuutta, josta voi muuttaa oletusarvoista etsintäsyvyyttä. Esimerkiksi edellisen kuvan Click-avainsana etsii elementtiä nimeltä ”Punctuation” syvyydeltä 10.

Elementtien aktivointi hoidetaan RPA.Windows-kirjaston avulla käyttämällä hiiren klikkauksia tai näppäimistön painalluksia, kuten kuvasta 11 voi päätellä. ”Click” vie hiiren näytöllä oikean elementin luo ja painaa sitä hiiren vasemmalla painikkeella. Myös oikean puolen painaminen ja tuplaklikkaukset onnistuvat, mutta näille on omat avainsanansa. ”Send keys” puolestaan painaa yhtä tai useampaa sille argumenttina annettua näppäintä. Hiiren etuna on se, että sillä voi painaa mitä vain elementtiä, mutta se on alttiimpi käyttäjän aiheuttamille häiriöille. Läheskään kaikille Trados Studion elementeille ei kuitenkaan ole olemassa pikanäppäimiä, joten robotin suorittamaa klikkailua tarvitaan.

```
RPA.Windows.Click    name:'Language Pairs'
RPA.Windows.Click    name:'All Language Pairs'
Send Keys    keys={DOWN}
Set Anchor    id:tableLayoutPanel3
Send Keys    id:_toolStrip    keys={ALT}(+u)
Send Keys    id:_toolStrip    keys=s
Send keys    keys={ENTER}
```

Kuva 11. Robotti sisältää paljon hiiren klikkauksia ja näppäimistön painalluksia.

Vaikka RPA.Windows-kirjasto onkin monilta ominaisuuksiltaan ketterä, on sen käyttö tuottanut myös haasteita, joista suurin on kirjaston sisältämien toimintojen rajallisuus: kaikille tarpeellisille vuorovaikutuksille Trados Studion kanssa ei ole olemassa avainsanoja. Esimerkkinä mainittakoon kuvassa 12 näkyvä valintaruudun tilan tarkistus, joka antaa robotille oleellista tietoa siitä, tuleeko ruutua painaa vai ei. Ratkaisuna tähän ongelmaan on ollut koodata osa avainsanoista suoraan Pythonilla, johon Robot Frameworkin syntaksi perustuu. Tässä on hyödynnetty uiautomation-kirjastoa, jonka päälle RPA.Windows on rakennettu. Se sisältää laajan kirjon keinoja muokata ja saada informaatiota erilasten kontrollityyppien arvoista. Sieltä löytyvät sopivat metodit valintaruudun tilan tarkistukselle ja muille toiminnoille, joihin RPA.Windows ei pysty.

```
from uiautomation.uiautomation import CheckBoxControl, RadioButtonControl, DataItemControl, ListControl
from uiautomation.uiautomation import SendKeys as Presskeys

#This script contains python-made support keywords for robot. They utilize classes and methods from
#uiautomation library, which is the same library that RPA.Windows uses.

def check_and_correct_checkbox(value, locator):
#Checks if checkbox has correct value based on value parameter. If not, corrects the value
        checkbox = CheckBoxControl(AutomationId=locator)
        checkvalue = checkbox.GetTogglePattern().ToggleState #Gets information about checkbox state
        value= int(value)
        if checkvalue != value:
                checkbox.GetTogglePattern().Toggle(waitTime=0.1) #Toggles the checkbox
```

Kuva 12. Avainsana valintaruudun tarkistamiseksi Pythonilla koodattuna, missä on hyödynnetty uiauto-mation kirjastoa.

Kuten jo poikkeustilanteita käsitellessä tuli ilmi, internet-yhteyden hitaus tai sen puute voi aiheuttaa merkittäviä haasteita käännösmuistia valittaessa. Ongelmana on ensinnäkin se, että jos käännösmuisti ei lataudu ajoissa voi robotti kaatua. Tämän ratkaisu on sinällään varsin suoraviivaista: mikäli muisti ei lataudu määräaikaan mennessä, voi muistin valintaan ja siihen sidoksissa olevat asetukset ohittaa ja jättää ne käyttäjän säädettäväksi. Tässä hyödynnetään kuvan 13 mukaisesti Run keyword and return status -avainsanaa, joka palauttaa arvon sen mukaan, onnistuiko argumenttina ollut avainsana vai ei. Arvoa käytetään ehtolauseessa, joka suorittaa oikean toiminnan sen mukaan. Asiaa kuitenkin hankaloittaa se, että muistin latausruutu jää etualalle pyörimään, mikä estää robottia poistumasta valikosta. Tähän ei toistaiseksi ole keksitty muuta ratkaisua, kuin pidentää avainsanan oletusarvoista aikakatkaisua puoleen minuuttiin ja toivoa, että latausruutu on hävinnyt. On kuitenkin mahdollista, että kestävämpi ratkaisu löytyy tulevaisuudessa.

```
${memory_status}=    Run Keyword And Return Status    Get Element
...    type:Edit name:'Source Language' depth:20
...    timeout=30
Set Global Variable    ${memory_status}
Sleep    500ms
IF    ${memory_status} == True
    Expand Item    Lingsoft    #Expands the item given as argument
    Scroll To Item    ${memory_library}    #Scrolls to item in list given as argument
    RPA.Windows.Click    id:okButton depth:12
    Control Window    id:SettingsDialogForm
    Get Element    type:DataItem name:'False' depth:16
ELSE
    ${warning_status}=    Run Keyword And Return Status    Get Element    name:Warning    timeout=3
    Run Keyword Unless    ${warning_status}==False    RPA.Windows.Click    name:'Warning' > name:'OK'
    RPA.Windows.Click    id:cancelButton
    Control Window    id:SettingsDialogForm
END
```

Kuva 13. Käännösmuistin valinnassa tarkistetaan internet yhteys etsimällä elementtiä ruudulta.

Käyttäjän näkökulmasta on tärkeää saada informaatiota siitä, onko robotin ajo päättynyt ja kuinka se on onnistunut. Tätä varten robottiin on koodattu kaksi erilaista kuvassa 14 näkyvää ikkunavaihtoehtoa käyttämällä Robot Frameworkin Dialogs-kirjastoa, jonka avulla on mahdollista lähettää käyttäjälle viestejä tai pyytää palautetta pienen viesti-ikkunan kautta. Hyödyntämällä [Teardown]-etupäätettä voidaan määritellä, että robotti pyrkii aina ajon päätteeksi näyttämään epäonnistumisdialogin. Siihen on kuitenkin määritelty ehdoksi, että muuttujalla ${success} on oltava tietty arvo, joka on riippuvainen siitä, onko onnistumisdialogi jo näytetty vai ei. Tällä tavoin voidaan ohjata, kumman viestin robotti näyttää.

```
Show success dialog
#Shows a success dialog if robot ran without errors.
    Set Global Variable    ${success}    True    #Set new value for global variable
    ${choice}=    Get Selection From User
    ...    Settings have been successfully modified!${\n}Please select if you want to save settings:
    ...    Yes    No
    IF    "${choice}" == "Yes"
        Send Keys    keys={RETURN}
    ELSE
        Send Keys    keys={ESC}
    END

Show failure dialog
#Shows a failure dialog if robot encountered errors.
    Pause Execution    message=${\n}${SPACE*3}"Oops, something went wrong with the robot!"${SPACE*3}${\n}
```

Kuva 14. Dialogien koodi.

Onnistumisikkuna kertoo ajon onnistuneen ja pyytää käyttäjää valitsemaan, tallentaako asetukset vai ei. Epäonnistumisikkuna näyttää lyhyen viestin siitä, ettei ajo ole onnistunut. Viesti on toistaiseksi sama virheestä riippumatta. On kuitenkin pohtimisen arvoista, josko viestin sisältöä voitaisiin myöhemmin muokata vastaamaan virheen luonnetta. Suurin osa virheistä ei luultavasti ole käyttäjän korjattavissa, mutta ainakin sellaisissa tapauksissa, joissa virheen syynä on internet-yhteys, olisi käyttäjänkin mahdollista vaikuttaa asiaan.

4.5 Paketointi ja käyttö

Robotin käyttöä pohdittaessa on otettava huomioon, ettei kääntäjillä ja koordinaattoreilla todennäköisesti ole paljoa kokemusta koodieditorin tai komentorivin käytöstä. Niinpä näiden työkalujen käyttö robotin ajamiseen ei ole suositeltavaa. Myös asennuksen tulisi olla yksinkertaista. Robotti vaatii kuitenkin toimiakseen erilaisia Pythonilla koodattuja kirjastoja, joten näiden tulisi löytyä käyttäjän koneelta. IT-ammattilainen voisi hoitaa näiden asennuksen, mutta se olisi turhan työlästä.

Robotin käyttäjäystävällisyyden varmistamiseen on olemassa erilaisia ratkaisuja. Eräs mahdollisuus on hyödyntää Robot Frameworkille suunnattuja palveluja robotin ajamisessa. Esimerkiksi automaatioon erikoistunut Robocorp tarjoaa pilvipalveluja ja näihin sidoksissa olevia sovelluksia, joiden avulla on mahdollista käyttää ja hallita robotteja. Palvelujen ilmainen käyttö on kuitenkin rajallista ja vaatisi rekisteröitymistä. (Robocorp 2021.) Koska kyse on toistaiseksi vain yhdestä robotista, ei maksaminen välttämättä kannata. Toinen vaihtoehto on luoda asennuksessa robotin kansioon virtuaaliympäristö, jonne ladattaisiin tarvittavat kirjastot. Näin niitä ei tarvitsi asentaa suoraan koneelle. Robotti voitaisiin ajaa batch-tiedostosta, joka suorittaisi ensimmäisellä kerralla myös virtuaaliympäristön asennuksen. Ratkaisu toimii hyvin, mutta huonona puolena vaatisi Pythonin esiasennusta koneelle.

Paras näköpiirissä oleva ratkaisu on pakata Python kirjastoineen ohjelmatiedostoon (executable) ja ajaa robotti se suorittamalla. Näin käyttäjän ei tarvitse asentaa erikseen Pythonia tai sen kirjastoja koneelle. Robot Frameworkin robot -ja resource-tiedostot voidaan jättää ohjelmatiedoston ulkopuolelle. Niiden sisällyttäminen olisi turhaa ja kasvattaisi tiedostokokoa, mikä hidastaisi tiedoston suorittamista.

Paketoinnissa voidaan käyttää pyinstaller-työkalua, joka on tarkoitettu Pythonin ja sen riippuvuuksien paketointiin. Työkalu on ilmainen ja on ladattavissa komentorivin kautta käskyllä ”pip install pyinstaller.” Se toimii siten, että ensin luodaan Python-skripti, johon on tuotu tarvittavat riippuvuudet, kuten kuvassa 15 näkyy**.** Työkalu asentaa nämä riippuvuudet, kun tiedosto paketoidaan. Koska robotin pääsuoritustiedosto jää ohjelmatiedoston ulkopuolelle, skriptiin on lisätty myös koodirivi, joka käynnistää kyseisen tiedoston suorittamisen ja määrittää ajon tuottamille lokitiedostoille kansion, johon ne tallentuvat

```
import robot
import RPA.core
import RPA.scripts
import RPA.Windows
import uiautomation

#This is the script that runs the robot from StudioSettingsChecker.exe

robot.run("tasks.robot", outputdir="output\\")
```

Kuva 15. Python-skripti, josta luodaan ohjelmatiedosto.

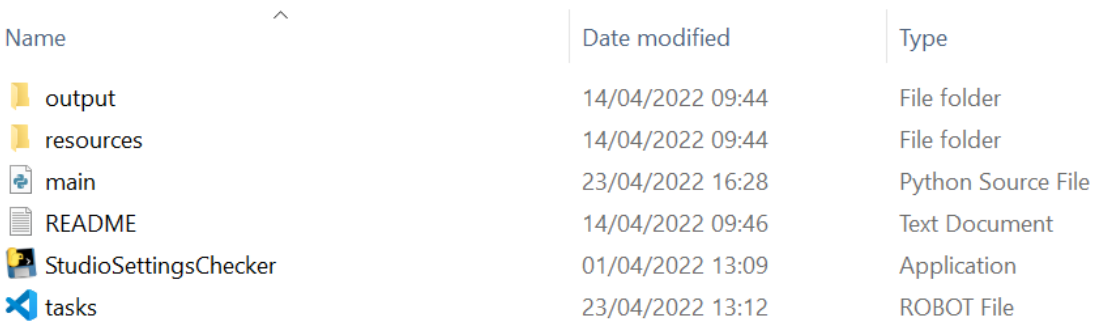
Tämän jälkeen komentoriviltä ajetaan käsky, jossa määritellään pakattava tiedosto. Yksinkertaisimmillaan käsky olisi ”pyinstaller main.py”, eli vain työkalun ja pakattavan ohjelman nimi. Kuten kuvasta 16 saattaa huomata, on paketoinnissa kuitenkin käytetty lisäasetuksia. Näistä ensimmäinen on ”--onefile”, joka määrittää, että kaikki Python-tiedostot paketoidaan yhteen tiedostoon. Toinen on ”--collect-all”, joka kerää kaikki Python-paketin moduulit sekä data- ja binääritiedostot (Cortesi 2022). Tässä tapauksessa jälkimmäinen

kohdistuu robot-pakettiin. Tämä johtuu siitä, että muutoin kaikki sen sisältämät moduulit eivät asennu kunnolla, mikä jättää joitakin robotin tarvitsemia osia saamatta.

```
>pyinstaller --onefile main.py --collect-all robot
```

Kuva 16. Ohjelmatiedoston luonti komentoriviltä pyinstallerin avulla.

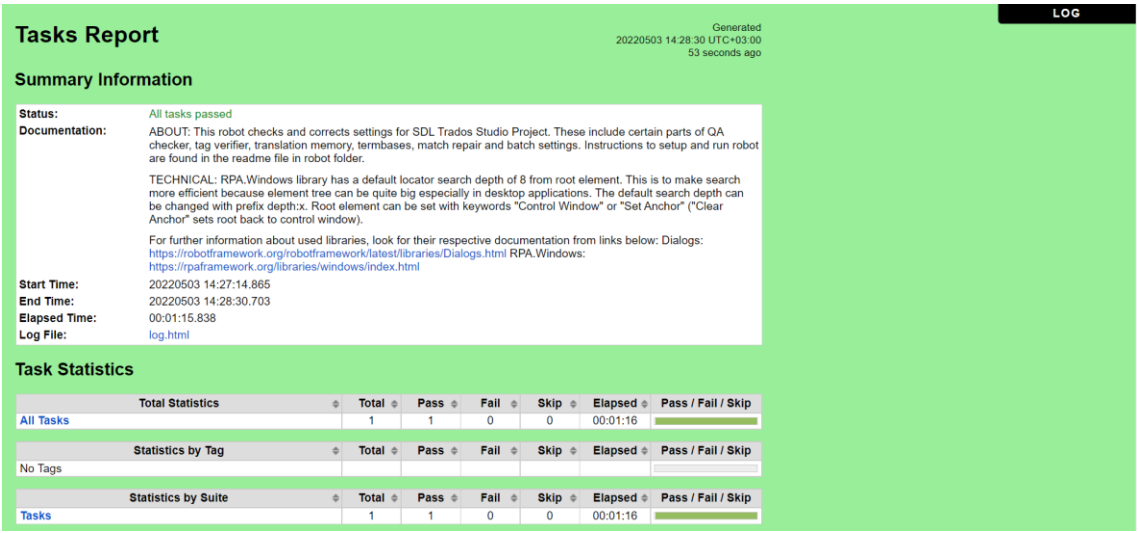
Kun tarvittavat Python-kirjastot on näin paketoitu, on robotti valmis käytettäväksi. Se ei varsinaisesti tarvitse erillistä asennusta, riittää että käyttäjä lataa robotin kansion yrityksen tiedostonjakoresursseista. Kuvassa 17 näkyvän kansion juuresta löytyvät ohjelmatiedosto (StudioSettingsChecker.exe), robotin päätiedosto (tasks.robot), käyttöohjeet (README.txt) sekä paketoinnissa käytetty Python-skripti (main.py). Output-kansioon tulevat robotin ajon aikana tuottamat lokitiedostot ja resources-kansiossa ovat robotin tukitiedostot. Robotin ajamiseen riittää se, että käyttäjä klikkaa ohjelmatiedostoa. Ainoat manuaaliset toimenpiteet ovat Trados Studion avaaminen ja oikean projektin valinta.



Kuva 17. Robotin kansiorakenne.

Kuvassa 18 on robotin tuottama raportti projektiasetusten säätämisestä. Esimerkkiä varten alkuasetukset on säädetty mahdollisimman väärin, koska tämä lisää robotin suoritusaikaa jonkin verran. Suoritusaikaan saattavat

vaikuttaa alkuasetusten ohella internet-yhteys sekä käytössä oleva tietokone, joka on tässä tapauksessa Lenovo Thinkpad 13 G2. Kuten tiedostosta voidaan huomata, on ajo toteutunut onnistuneesti eikä aikaa ole kulunut paljoa minuuttia kauempaa. Robotti siis kykenee suoriutumaan tehtävästään varsin nopeasti, kun ottaa huomioon, että säädettäviä valintaruutuja, vetovalikoita, arvokenttiä ja muita valikoita on yli kahdeksankymmentä. Alle minuutinkin kestäviin ajoihin on mahdollista päästä, mikäli robotin ei tarvitse muuttaa paljoa asetuksia.



Kuva 18. Raportti robotin suorituksesta.

5 Lopuksi

Opinnäytetyön tavoitteena oli suunnitella ja rakentaa ohjelmistorobotti, josta voisi saada kustannussäästöjä säästyneen työajan muodossa yritykselle. Toteutetun tapauksen kautta kertyneen tiedon perusteella voidaan myös arvioida, kuinka hyvin ohjelmistorobotiikka soveltuu yrityksen kieli- ja käännösprosessien automatisointiin ja mitä sen menestyksekäs toteuttaminen vaatii.

Nykyisellään robotti on lopputestausta vaille valmis ja toteuttaa kaikki edeltävän Autohotkey-skriptin toiminnot sekä toiveena olleet lisäominaisuudet. Työtä on edesauttanut se, että Trados Studio soveltuu hyvin ohjelmistorobotiikan kohteeksi: sen elementtirakenne on enimmäkseen saavutettavissa ja elementeistä löytyy paljon ominaisuuksia, joita hyödyntämällä vuorovaikutus robotin kanssa on mahdollista. Tämä ei välttämättä ole mikään itsestäänselvyys Windowsin työpöytäsovelluksista puhuttaessa. Acessibility Insights on ollut korvaamaton apu siinä, että Trados Studion elementeistä ja niiden rakenteesta on saanut tarvittavaa informaatiota. Alkuun ei ollut lainkaan selvää, kuinka tähän tietoon pääsisi käsiksi ja miten sitä voisi hyödyntää. Työssä on auttanut myös aiemmin tehty dokumentaatio, jonka sisältämä informaatio säädettävistä asetuksista sekä näiden arvoista ja säätöjärjestyksestä on paitsi helpottanut robotin rakentamista, myös määrittänyt sen rakennetta.

Robotin nopeus tulee ilmeiseksi, kun ajatellaan, että käyttäjän täytyisi manuaalisesti navigoida kaikkien asetusten läpi ja säätää paitsi kymmenien valikoiden asetukset, myös muistaa näiden asetusten arvot. Sen sijaan robotilta kestää korkeintaan noin 1–2 minuuttia säätää Trados Studion asetukset kohdalleen. Säästämällä aikaa voidaan säästää myös rahaa. Projekteihin käytettävä työmäärä vähenee, mahdollistaen tehdyn työn volyymin kasvattamisen, mikä lisää yrityksen katetta. Vaikka yksittäisen robotin tuottama säästö on vielä kokonaiskuvan kannalta vähäinen, myötävaikuttaa se osaltaan pyrkimyksiin tehostaa työprosessia automatisoinnin avulla.

Kustannussäästöjen lisäksi kyseisestä robotista on myös muita hyötyjä, joista yksi on laadun paraneminen: kun puhutaan kymmenien asetusten säätämisestä, on mahdollisuus inhimillisiin virheisiin ilmeinen. Robotti taas hoitaa tehtävänsä aina samalla tavalla, mikä vähentää virheiden mahdollisuutta. Laatu voi parantua myös epäsuorasti, kun työntekijöillä on enemmän aikaa keskittyä oleellisiin asioihin. Tämä ja töiden valmistumisen nopeutuminen puolestaan johtavat asiakastyytyväisyyden lisääntymiseen. Kun rutiininomaiset, puuduttavat tehtävät vähenevät, paranee myös työntekijöiden viihtyvyys.

Opinnäytetyön perusteella voi sanoa, että ohjelmistorobotiikka Robot Frameworkilla toteutettuna soveltuu hyvin Trados Studion ja mahdollisesti muiden yrityksen käyttämien työpöytäsovellusten automaatioon. Onnistunut automaatio edellyttää kuitenkin muutamia asioita. Teknisenä edellytyksenä on, että sovelluksessa on hyvin toteutettu ja saavutettavissa oleva elementtirakenne, mikä mahdollistaa vuorovaikutuksen sovelluksen ja robotin välillä. Tehtävänannon näkökulmasta on tärkeää, että automatisoitavat toimenpiteet ovat luonteeltaan rutiininomaisia, eivätkä sisällä monimutkaista päättelyä. Myös suorituksessa tarvittavan informaation, kuten asetusten arvon, tulee joko suoraan tai välillisesti olla robotin helposti saavutettavissa. Joistakin muista RPA-alustoista poiketen, jotka käyttävät robotin rakentamisessa visuaalisia työkaluja, edellyttää Robot Framework kehittäjältään myös koodaamisen perusteiden osaamista.

Seuraava vaihe robotin kehityksessä on testeissä löytyvien vikojen korjaaminen sekä parannusehdotusten arviointi ja mahdollinen toteutus. Vikojen korjaamisen haasteena voi olla erilainen käyttöympäristö kuin kehitettäessä ja jatkuva ohjelmistojen ja prosessien päivittäminen. Robot Frameworkin tuottamat lokitiedostot kuitenkin auttavat vikojen tunnistamisessa. Jos ajatellaan jo työn aikana ilmenneitä parannuskohteita, niin internet-yhteyden katkeamiseen tai hitauteen liittyviin ongelmiin olisi hyvä löytää nykyistä kestävämpi ratkaisu. Käyttäjille olisi hyvä antaa edellä mainitussa ja muissa vastaavissa tapauksissa

informaatiota vian luonteesta, koska heidän on mahdollista korjata ongelma itse.

Kun mietitään robotin jatkokehitystä pitkällä tähtäimellä, on yksi mahdollisuus laajentaa käyttökohteita. Tällä hetkellä robotti säätää oletusasetukset vain tietyn tyyppisille projekteille. Tulevaisuudessa voisi olla mahdollista laajentaa asetusten valinta koskemaan myös erilaisia käännösprojekteja. Tämä edellyttää tosin suuria muutoksia robotin rakenteeseen, sillä asetusten arvot luultavasti vaihtelisivat projektityypin mukaan. On myös tutkimisen arvoista, pystyisikö robotin toiminnassa hyödyntämään Trados Studion ohjelmointirajapintaa. Sen avulla voisi mahdollisesti olla helpompi tehdä joitakin toimenpiteitä kuin käyttöliittymän kautta.

Ennen opinnäytetyötä ymmärrykseni ohjelmistorobotiikasta oli lähinnä teoreettista, joten koen robotin suunnittelun ja rakentamisen kehittäneen suuresti käytännön osaamistani aihepiirin osalta. Tiedän nyt, miten ohjelmistorobotti rakennetaan, mitä edellytyksiä robotin toiminta vaatii ja minkälaisia haasteita siihen voi sisältyä. Ymmärrän myös paremmin, mitä ohjelmistorobotiikalla voi tehdä ja mihin kaikkeen sitä on mahdollista soveltaa. Opin samalla paljon uutta Windowsin työpöytäsovellusten rakenteesta ja ominaisuuksista sekä tavoista paketoida ja ajaa ohjelmia. Sain robotin suhteen melko vapaat kädet, ja tekeminen olikin varsin itsenäistä, mutta sain apua aina tarvittaessa. Koen työn edistyneen pääasiassa melko sujuvasti, joskin muutamat haasteet muun muuassa paketointiin ja sopivien kirjastojen puutteeseen liittyen hidastivatkin kehitystä. Luovuus, sinnikkyys ja päättelykyky olivat keskeisessä asemassa näiden ja monien muidenkin ongelmien ratkaisussa.

Lähteet

Cortesi, David 2022. Using PyInstaller. Viitattu 29.4.2022. https://pyinstaller.org/en/stable/usage.html

Kääriäinen & al 2018. Ohjelmistorobotiikka ja tekoäly – soveltamisen askelmerkkejä. Valtioneuvoston selvitys- ja tutkimustoiminnan julkaisusarja, 65/2018.

Microsoft 2019. UI Automation Properties Overview. Viitattu 17.4.2022. https://docs.microsoft.com/en-us/windows/win32/winauto/uiauto-propertiesoverview

Microsoft 2020a. UI Automation. Viitattu 17.4.2022. https://docs.microsoft.com/en-us/windows/win32/winauto/entry-uiauto-win32

Microsoft 2020b. UI Automation Control Patterns Overview. Viitattu 1.5.2022. https://docs.microsoft.com/en-us/windows/win32/winauto/uiauto-controlpatternsoverview

Microsoft 2021. UI Automation Overview. Viitattu 17.4.2022. https://docs.microsoft.com/en-us/windows/win32/winauto/uiauto-uiautomationoverview

Microsoft 2022a. Accessibility Insights for Windows. Viitattu 17.4.2022. https://accessibilityinsights.io/docs/windows/overview/

Microsoft 2022b. Python extension for Visual Studio Code. Viitattu 10.5.2022. https://marketplace.visualstudio.com/items?itemName=ms-python.python

Palsamäki, Jani 2021. Desktop automation strategies and best practices using image locators, keyboard navigation, OCR, and accessibility locators. Viitattu 1.5.2022. https://robocorp.com/blog/desktop-application-automation-strategies-and-best-practices

Pratt, Mary K 2021. What are the advantages and disadvantages of RPA? Viitattu 22.5.2022. https://www.techtarget.com/searchcio/feature/What-are-the-advantages-and-disadvantages-of-RPA

Robocorp 2021a. Language server Protocol for robot framework. Viitattu 1.5.2022. https://robocorp.com/docs/developer-tools/visual-studio-code/lsp-extension

Robocorp 2021b. Robocorp Control Room. Viitattu 1.5.2022. https://robocorp.com/products/control-room

Robot Framework 2022a. Getting Started. Viitattu 13.4.2022. https://robotframework.org/?tab=1#getting-started

Robot Framework 2022b. Robot Framework. Viitattu 14.3.2022 https://robotframework.org

Robot Framework 2022c. Robot Framework User Guide. Why Robot Framework? Viitattu 13.4.2022. https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#why-robot-framework

RWS Group 2020. Acquisition of SDL by RWS enhances capabilities in Language Services and Technology. Viitattu 10.5.2022. https://www.rws.com/about/news/2020/acquisition-of-sdl-by-rws-enhances-capabilities-in-language-services-and-technology/

RWS Group 2022. Trados Studio. Viitattu 17.4.2022. https://www.trados.com/products/trados-studio/

Tripathi, Alok Mani 2018. Learning Robotic Process Automation : Create Software Robots and Automate Business Processes with the Leading RPA Tool – Uipath. Packt Publishing Ltd, Birmingham.

Williams, Ethan 2021. Visual Studio vs. Visual Studio Code. Viitattu 10.5.2022. https://www.codeguru.com/visual-studio/visual-studio-vs-visual-studio-code/