RPA tällä hetkellä ja tulevaisuudessa - RPA-projektien teko Robot Frameworkillä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

25.11.2022

Tiivistelmä

Tekijä:	Niki Tuomiluoto
Otsikko:	RPA tällä hetkellä ja tulevaisuudessa - RPA-projektien teko Robot Frameworkillä
Sivumäärä:	40 sivua
Aika:	25.11.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori Simo Silander

Tämän opinnäytetyön tavoitteena on ollut käydä läpi ohjelmistorobotiikan (RPA) historiaa, tarkoitusta ja tulevaisuuden näkymiä ja selvittää, miten Robot Framework sopii ohjelmistorobotiikan projektien tekoon. Opinnäytetyössä tutustuttiin ohjelmistorobotiikkaan yleisesti sekä selvitettiin sen hyötyjä ja haittoja. Pohdittiin myös ohjelmistorobotiikan etuja ja kannattavuutta erilaisissa projekteissa.

Työssä avattiin avoimeen lähdekoodiin perustuvaa Robot Frameworkiä automaatiotyökaluna ja sen käyttöä RPA-projekteissa. Samalla syvennyttiin sen rakenteeseen ja moduuleihin. Ohjelmistorobotiikkaan sopivia moduuleja käydessä läpi katsottiin tarkemmin myös RPA Framework -kirjastokokoelmaa ja sen käyttöä ohjelmistorobotiikan projekteissa.

Työhön toteutettiin kolme hypoteettista RPA-projektia, jotka toteutettiin Robot Frameworkillä ja sen monia kirjastoja käyttäen. Nämä esimerkkitoteutukset olivat korkean tason toteutuksia, joiden tarkoitus oli näyttää, miten Robot Frameworkillä voidaan ohjelmistorobotiikan projekteja toteuttaa.

Opinnäytetyön viimeiset osiot keskittyivät ohjelmistorobotiikan suosioon, markkinan kasvuun ja vaikutukseen ohjelmistokehitykseen. Työn lopussa summattiin käydyt asiat yhteen ja laadittiin loppupäätelmät ja pohdinnat.

Avainsanat: ohjelmistorobotiikka, ohjelmistorobotti, Robot Framework, automaatio, avoin lähdekoodi

Abstract

Author:	Niki Tuomiluoto
Title:	RPA Now and in the Future – Making RPA Projects with Robot Framework
Number of Pages:	40 pages
Date:	25 November 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Software Engineering
Instructors:	Simo Silander, Senior Lecturer

The purpose of this thesis was to go through the history, purpose and views of the future of robotic process automation (RPA). Robot Framework was at the point of attention for this thesis and how would it fair as tool for RPA-projects. Thesis took an overall look in RPA as well as it's pros and cons and other advantages in different projects.

The thesis covers the open-source automation tool Robot Framework in depth, looking into its structure and modules. While inspecting what modules could be used for RPA, RPA Framework library-collection was inspected as useful tool for RPA-projects.

Three hypothetical RPA-projects were implemented using Robot Framework and its many libraries. These example implementations were high-level implementations, the purpose of which was to show how Robot Framework could be used in RPA-projects.

The last parts of the thesis were focused on popularity of the RPA and how has this popularity shown in the RPA market. The effects of RPA for software development and its processes were considered as well. At the end of the thesis, all previous parts were summed up and final conclusions and thoughts were given.

Keywords: robotic process automation, automation, software robotic, Robot Framework, open-source software

Sisällys

Lyhenteet

RPA: *Robotic Process Automation*. Suomeksi ohjelmistorobotiikka on yleistermi ohjelmistoille, joilla tehdään ohjelmallisesti tehtäviä erilaisissa ympäristöissä ja järjestelmissä, mutta eivät ole kuitenkaan osa alkuperäistä järjestelmää. Tyypillisesti nämä ohjelmat tehdään matkimaan ihmisen toimintaa järjestelmissä.

SOAP: *Simple Object Access Protocol* on viestipohjainen tietoliikenneprotokolla. Sovellusohjelmat lähettää SOAP-kutsuja toisilleen ja yleensä käyttää XML-rakennetta.

XML: *Extensible Markup Language* on merkintäkielien standardi, jota käytetään useasti tiedonvälitys formaattina loogisen rakenteen vuoksi.

REST: *Representational state transfer* on arkkitehtuurityyli ohjelmointirajapintojen toteuttamiseen. REST-arkkitehtuuri ei ota kantaa, millä protokollalla sitä implementoidaan, mutta useimmiten sitä toteutetaan HTTP- tai HTTPS-protokollilla.

BPM: *Business Process Management*. Suomeksi prosessijohtaminen on johtamisoppi, joka käyttää useita tapoja todentaa, mallintaa, analysoida, mitata, parantaa ja optimoida liiketoiminnan prosesseja.

BPMS: *Business Process Management Suite*. Suomeksi liiketoimintaprosessien hallintatyökalukokoelma eli työkalukokoelma, jolla prosessijohtoasiantuntijat voivat suorittaa tavoitteensa.

iBPMS: *intelligent Business Process Management Software*. Suomeksi älykäs liiketoimintaprosessien hallintaohjelmisto. Tämä on lyhyesti tekoälyn ja BPMS:n yhdistelmä.

PyPI: *Python Package Index.* Python-paketti-indeksi on tietovarasto Python-ohjelmistoille ja kirjastoille.

CSV: *Comma-separated values*. Suomeksi pilkuilla erotellut arvot on tiedostomuoto, johon tallennetaan taulukkotietoa.

API: *Application Programming Interface*. Suomeksi ohjelmointirajapinta on rajapinta, johon muut sovellukset pystyvät ohjelmallisesti tekemään kyselyitä.

JVM: *Java virtual machine*. Suomeksi Java-virtuaalikone on abstrakti kone tai virtuaalikone, joka suorittaa käännettyjä Java-ohjelmia.

1 Johdanto

Robotic Process Automation (RPA) eli ohjelmistorobotiikka tuli julkisuuteen 2010 vuosikymmenen alussa ja on ollut kasvussa siitä saakka. Koronaviruspandemia on saanut maailmanlaajuisesti monet yritykset ja tahot huomaamaan ohjelmistorobotiikan ja automaation edut.

Tässä opinnäytetyössä käydään läpi ohjelmistorobotiikan perusideaa, sen hyötyjä ja haittoja sekä miksi ja milloin sitä kannattaisi käyttää. Näitä ominaisuuksia katsotaan muutamien esimerkkien avulla ja koitetaan tarkastella käytännöllisyys edellä. Työssä paneudutaan vahvasti Robot Frameworkiin ja sen soveltuvuuteen ohjelmistorobotiikan ja automaation työkaluna. Robot Framework on avoimenlähdekoodin projekti, ja se on helposti laajennettavissa oleva automaatioalusta. Robot Frameworkillä on monia moduuleita tai kirjastoja ja näistä moduuleista ohjelmistorobotiikalle relevantteja käydään läpi. Näistä yksi on RPA Framework, joka on ylläpidetty kokoelma avoimen lähdekoodin kirjastoja ja työkaluja.

Teorian päätteeksi katsotaan kolmea hypoteettista ohjelmistorobotiikan projektia. Nämä esimerkkitoteutukset toteutetaan Robot Frameworkiä ja RPA Frameworkiä käyttäen. Myös muita kirjastoja käytetään esimerkkitoteutuksissa, esimerkiksi DataDriver-kirjastoa. Esimerkkitoteutuksien idea on näyttää käytännön esimerkkinä robottien rakennetta, kun niitä tehdään Robot Frameworkillä. Toteutukset ovat korkean tason esimerkkejä, joten niissä tehdään paljon oletuksia ja jätetään logiikkaa toteuttamatta.

Ennen loppuyhteenvetoa työssä katsotaan RPA:n tilannetta ohjelmistomarkkinoilla ja sen kasvua lähivuosilta ja myös tulevaisuudessa. Tarkastelun alle tulee myös kysymys, onko RPA:lla vaikutuksia ohjelmistokehitykseen ja sen prosesseihin.

Loppujen lopuksi yhteenvedossa niputetaan aikaisemmat kappaleet yhteen ja annetaan loppupäätelmät ja omat pohdinnat ohjelmistorobotiikasta.

2 Robotic Process Automation eli ohjelmistorobotiikka

Tässä luvussa käydään läpi ohjelmistorobotiikan hyötyjä, haittoja ja sitä, mitä pitää ottaa huomioon RPA-projektin kannattavuudessa. RPA:n haitat -osiossa kuvataan enemmänkin kompastuskiviä, joihin voi projektin varrella törmätä ja miten niitä voi välttää. Lopuksi RPA-projektin kannattavuus -osiossa käydään läpi, mitkä seikat ovat tärkeitä RPA-projektia suunnitellessa. Viimeinen luku toimii yhteenvetona hyötyihin ja haittoihin.

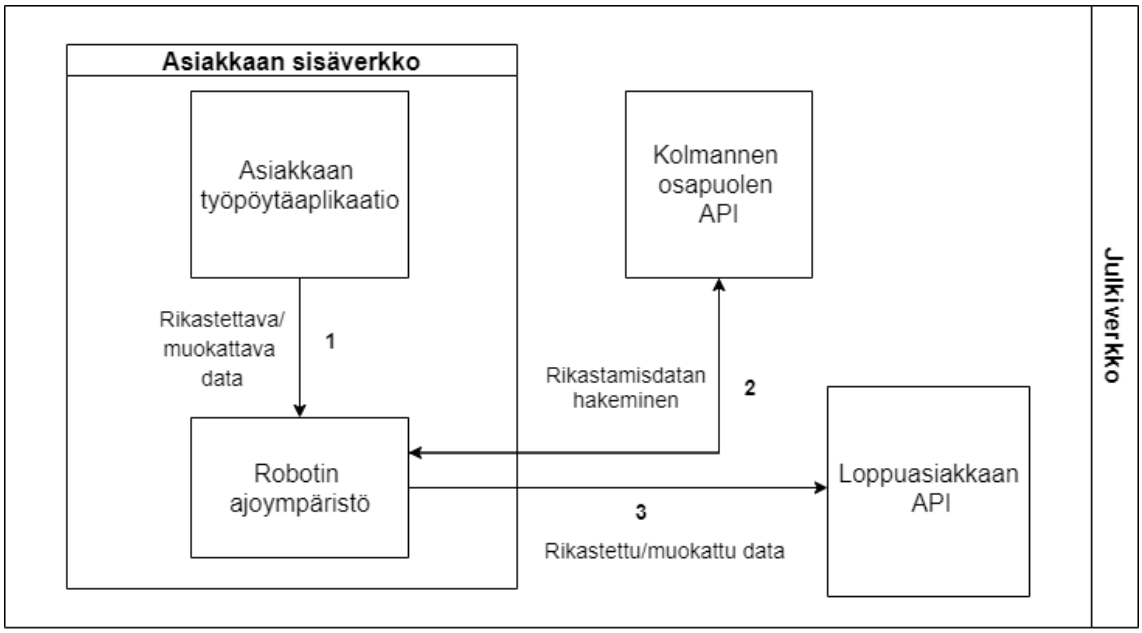
2.1 Mitä on RPA/ohjelmistorobotiikka?

Institute for Robotic Process Automation and Artificial Intelligence (IRPA AI) määrittelee ohjelmistorobotiikan sellaisen teknologian sovellukseksi, jolla yrityksen työntekijät pääsevät konfiguroimaan ohjelmiston tai ”robotin” kaappaamaan ja tulkkaamaan jo olemassa olevia toteutuksia tapahtumien, datan muokkauksen, vastausten käynnistämisen ja muiden digitaalisten järjestelmien kanssa kommunikoinnin käsittelyyn. [1.] Eli lyhyesti, ohjelmistorobotiikka on ohjelmistojärjestelmien tehtävien automatisointia järjestelmistä erillisillä ohjelmistoilla.

Ohjelmistorobotiikkaa käytetään luomaan digitaalista työvoimaa automatisoimalla toistuvia tehtäviä, jotta voidaan maksimoida työntekijöiden tuottavuus antamalla heille mahdollisuus keskittyä arvokkaampiin tehtäviin. [2.] Ohjelmistorobotiikkaa voidaan toteuttaa ulkoisilla ohjelmistoilla, jotka käyttävät muita ohjelmistoja tai päätelaitteita ja käynnistyvät yleensä jostain tapahtumasta kuten saapuvasta sähköpostista tai käyttäjän syötteestä. Tyypillisesti ohjelmistorobotit tehdään matkimaan ihmisen toimintaa järjestelmissä. Lisäksi on mahdollista tehdä ohjelmistorobotteja, jotka käyttävät vain työkaluja (esim. Excel-ohjelma) tai rajapintoja, joihin käyttäjille ei ole käyttöliittymää.

Tästä on hyvä huomioida, että RPA ei ole siis osa käytettäviä sovelluksia tai ohjelmistoja, eikä se ole fyysinen robotti vaan erillinen tietokoneohjelmisto, jolla käytetään tai lisätään toimintoja jo käytettäviin järjestelmiin tai ohjelmistoihin.

Ohjelmistorobotteja on erilaisia riippuen niiden käyttötarkoituksesta. Näitä ovat esimerkiksi käyttöliittymää automatisoivia robotteja, datamuokkaus- ja siirtorobotteja sekä datankeruu- ja analysointirobotteja. Kuvassa 1 on esimerkki, minkälainen datamuokkausrobotin ympäristö voisi olla.



Kuva 1. Esimerkki datamuokkausrobotin rakenteesta.

Kuvan 1 datamuokkausrobotti sisältää kolme vaihetta. Ensimmäisessä vaiheessa robotin ajoympäristöön siirretään rikastettava alkuperäisdata asiakkaan sisäverkossa olevasta työpöytäsovelluksesta. Toisessa vaiheessa haetaan julkiverkossa olevasta kolmannen osapuolenrajapinnasta rikastamisdata. Tämän jälkeen muokataan alkuperäisdata haluttuun formaattiin. Alkuperäisdata voi olla jo valmiiksi muokattavassa virtuaalisessa formaatissa, mutta on mahdollista, että joudutaan käsittelemään dataa haluttuun formaattiin. Samalla kun datamuokkausta tehdään, suoritetaan myös datan rikastus. Datan rikastus voi pitää esimerkiksi sisällään, että rikastusdatasta poimitaan halutut datat ja lisätään ne alkuperäiseen dataan tai molemmat datat yhdistetään uudeksi datakokonaisuudeksi. Kolmannessa vaiheessa siirretään rikastettu tietoaineisto loppuasiakkaalle rajapintakutsulla.

Ohjelmistorobotit yleisesti jaetaan kahteen kategoriaan robotin käynnistämistavan mukaan: *valvottuihin* ja *valvomattomiin* robotteihin.

Valvotulla (eng. attended) robotilla tarkoitetaan ohjelmistorobottitoteutusta, jossa käyttäjä joutuu tekemään manuaalisia muutoksia, jotta robotin ajo saadaan suoritettua loppuun. Ne on tarkoitettu avustamaan käyttäjää tämän tehtävässään nopeuttamalla toistuvia tehtäviä. [2; 3.]

Yhtenä esimerkkinä valvotuista roboteista ovat ohjelmistorobotit, jotka käyttäjä käynnistää manuaalisesti. Toisena esimerkkinä on robotti, joka on syöttänyt dataa järjestelmään ja jää odottamaan käyttäjältä tämän tekemää arvojen tarkastusta, kunnes voi jatkaa tehtävänsä loppuun.

Valvomattomilla (eng. unattended) ohjelmistoroboteilla taas tarkoitetaan, että robotti voi suorittaa tehtävänsä alusta loppuun ilman käyttäjän toimenpiteitä tai syötteitä. Nämä ohjelmistorobotit yleensä käynnistyvät joko ajastetusti tai jostain tapahtumasta. [2; 3.]

Esimerkkinä valvomattomista roboteista ovat robotit, joilla siirretään öisin päivän aikana tehdyt kaupat ulkoiseen rekisteriin, tai robotit, jotka käynnistyvät sähköpostiin tulleesta sähköpostista.

Valvotut ja valvomattomat robotit eivät ole toisiaan poissulkevia. Niillä on omat roolinsa ohjelmistorobottien käyttöönottoprojekteissa. Näiden robottien kombinaatio tuottaa kokonaisen ohjelmistorobotiikkaratkaisun. [3.]

2.2 Ohjelmistorobotiikan hyödyt

Ohjelmistorobotiikan hyödyt ovat lähes samat kuin automatisoinnilla yleisesti eli toistuvien, aikaa vievien tehtävien automatisointi, mikä vapauttaa työntekijät tekemään muita vaativimpia tehtäviä. Tämä automatisointi vähentää virhetilanteita, nopeuttaa prosessia sekä mahdollistaa palvelujen skaalaamista.

Näiden hyötyjen lisäksi on paljon muita hyötyjä, joita ei yleensä tulla ajatelleeksi. Näitä ovat:

- **Yhteensopivuus olemassa olevien järjestelmien kanssa** – Koska RPA-robotit yleisesti matkivat käyttäjien toimintoja, suurin osa palveluista ja järjestelmistä voidaan automatisoida ilman, että niihin tehdään muutoksia. Tämä voi vähentää kehitysaikaa, varsinkin kun kyseessä on vanhempia järjestelmiä. [4.]
- **Paremmat hallinnointimahdollisuudet** – RPA tuo myös työkalut monitorointiin, hallintaan ja organisointiin. RPA mahdollistaa myös näiden tekemisen etänä. Tämän lisäksi datan tarkastus ja analytiikkadatan keruu on mahdollista robotin ajon aikana, kuten myös tallentaminen myöhemmäksi analysointia varten. [4.]
- **Dataintegraatiot ja raportointi** – Koska RPA on ulkoinen ohjelmisto, se ei ole rajoitettu käyttämään samoja dataformaatteja kuin automatisoitavat järjestelmät käyttävät. Tämä mahdollistaa sen, että dataa voidaan hakea eri ympäristöistä ja muokata se haluttuun formaattiin. RPA-robotit tallentavat teoistaan lokin ajoista, jota voidaan jälkikäteen tarkastaa tai käyttää viranomaisraportoinnissa. [4.]
- **RPA mahdollistaa liiketoimintaa** – RPA auttaa myös pieniä ja keskisuuria yrityksiä suorittamaan liiketoimintaansa hyvän hinta-laatusuhteensa takia. Aikaisemmin tehtävät, jotka olisivat vaatineet paljon resursseja suorittaa, hoituvat RPA-robottien avulla kustannustehokkaasti. Tämä rohkaisee uusien liiketoimintaprosessien käyttöönottoa. [5.]
- **RPA parantaa työhyvinvointia** – Harvalle samojen tehtävien teko päivästä ja tunnista toiseen on mieluista ja siksi RPA-robotit parantavat työhyvinvointia vapauttamalla työntekijät toistuvista puuduttavista tehtävistä. Tällöin työntekijät voivat keskittyä haastavampiin tehtäviin ja itsensä kehittämiseen. [6; 7.]

Ohjelmistorobotiikasta on muutakin hyötyä kuin sen suorat vaikutukset. Hyvänä esimerkkinä tästä on se, että RPA-robotti on täysin erillinen ohjelmistototeutus, joten sen toteuttaja ei tarvitse olla sama taho kuin jo käytettyjen palvelujen tarjoaja. Tämä antaa yrityksille pelivaraa toteuttaa järjestelmän päälle toiminnallisuuksia, jotka eivät kuulu sopimukseen. Etuna on myös se, että yritykset eivät ole täysin palvelun tarjoajan armoilla jokaisen muutostarpeen osalta. Esimerkiksi jos yritys on tilannut toimittajalta palvelun, joka ei tue enää uusimpia datalähteitä, esimerkiksi SOAP-rajapinnasta REST-rajapintaan, niin yritys voi kilpailuttaa alkuperäisen palvelun toimittajan muutostyötarjouksen RPA-projektina muiden tarjoajien kanssa ja toteuttaa tarvittavat muutokset RPA-projektina.

2.3 Ohjelmistorobotiikan mahdolliset haitat

Ohjelmistorobotiikka on yleisesti saanut positiivista palautetta, mutta siinäkin piilee sudenkuoppia, joita kannattaa välttää. RPA-projektin haasteet ovat jo esillä alusta asti, suunnittelusta ja kehityksestä ylläpitoon ja omistukseen. Suunnitteluvaiheessa ongelmaksi voi koitua liian suuret odotukset ohjelmistorobottia kohtaan ja unohdetaan ottaa kaikkia riippuvuuksia huomioon, että robotti toimisi moitteettomasti. Näitä riippuvuuksia ovat esimerkiksi datalähteet, järjestelmiin pääsy ja liiketoiminnan prosessin sääntöpohjaisuus. Esimerkkinä tästä on, että kehitysvaiheessa huomataan API-rajapinta, joka toimii datalähteenä, on asiakkaan sisäverkossa eikä julkiverkossa ja robotin suunnitellusta ajoympäristöstä ei ole pääsyä asiakkaan sisäverkkoon. Pitää siis jo projektin alussa kartoittaa hyvin, mitä robotti ratkaisee, mitä se vaatii toimiakseen ja miten sitä ylläpidetään. [8.]

Kehitysvaiheessa ongelmaksi voi tulla esimerkiksi datan hankkimisen ongelmat, järjestelmiin pääsy, järjestelmien muuttuminen ja ajoympäristöjen vakaus tai hitaus. Esimerkit kahdesta jälkimmäisestä ovat yllättävät kuormat ohjelmistorobotin suoritusympäristöstä tai heikosta verkkoyhteydestä. Dataongelmat ja järjestelmiin pääsyongelmat taas yleensä johtuvat heikosta suunnittelusta tai resurssien puutteesta. On myös pidettävä mielessä, että mikäli alla olevat järjestelmät muuttuvat, oli kyseessä sitten käyttöliittymä tai ohjelmistot, robotit harvoin osaavat muokkautua näihin muutoksiin ja tällöin hajoavat. [9.] Ohjelmistorobottien ajoympäristön valinnassa kannattaa ottaa huomioon pääsy käytettäviin järjestelmiin. Mikäli käytettävät järjestelmät ovat eri verkoissa, on otettava huomioon mahdolliset ongelmat verkkopääsyjen kanssa ja tunnistautumisissa.

Vaikka ohjelmistorobotit helpottavatkin työntekijöiden työtaakkaa, on tahoja, jotka ajattelevat, että robotit vievät heiltä työpaikat. On siis hyvin tärkeää ottaa työntekijät mukaan suunnittelu- ja kehitysprosessiin ja kouluttaa heitä asiasta. Tärkeintä on saada työntekijät ymmärtämään, että ohjelmistorobotit ovat heitä varten, heidän avukseen, eivätkä heitä korvaamassa. [8.] Mikäli työntekijät eivät ole kehityksessä mukana, voi robotin suunnittelu helposti olla puutteellinen tai

virheellinen. Tämä voi johtaa siihen, että ohjelmistorobotti tekee tehtäviään väärin, jolloin siitä saatu hyöty on vähäinen tai jopa pahimmassa tapauksessa se haittaa liiketoimintaa. Siksi on tärkeää saada liiketoiminnan henkilöt kertomaan, mitkä ovat oikeat tehtävät auttamaan heitä. [8.]

Kun robottia kehitetään, on pidettävä mielessä, kuka robotin omistaa. Yrityksen eri työntekijät voivat auttaa robotin eri kehitysvaiheissa. Liiketalousprosessin osaavat henkilöt pystyvät antamaan tarkat vaatimukset robotille, he voivat auttaa hyväksymistestauksessa ja antaa arvion projektin onnistumisesta. Tällöin he voivat varmistaa, että ratkaisu on pätevä kokonaisuus. Tietotekniikkatiimi on tarvittu robotin ajoympäristön luontiin, tunnuksiin sekä testidatan luontiin. Vaikka käytettäviä järjestelmiä päivitettäisiinkin harvoin, ohjelmistorobotit ovat silti ohjelmistoja ja voivat tarvita ylläpitoa. Tähän on hyvä varautua RPA-ylläpitotiimillä, varsinkin tilanteissa, joissa robotti suorittaa yritystoiminnalle kriittistä työtä. Siksi on hyvä, että RPA-projektin ja robotin omistajuus koostuu useamman tiimin henkilöistä ja siksi näiden, liiketalous-, tietotekniikka-, RPA-ylläpitotiimien, välinen kommunikaatio on hyvin tärkeää. [4.]

Kun pohditaan ohjelmistorobotiikan käyttöönottoa, on pidettävä mielessä, että sen tarkoitus ei ole korvata ohjelmistojärjestelmiä ja toteuttaa *päästä päähän* (eng. End-to-End) -kokonaisuuksia. [8.] Myös liian laajaa ja nopealla aikataululla tehtyä RPA-toteutusta ei suositella, sillä se voi kaatua organisaation ja työntekijöiden riesaksi käyttöönoton vaikeuksina ja ylläpidon osalta. On siis parempi aloittaa pienellä ja yksinkertaisella RPA-toteutuksella. Kun se todetaan toimivaksi, niin voidaan laajentaa robottien määrää ja kokoa. [10.]

Ohjelmistorobotteja ei ole myöskään suunniteltu korjaamaan käytettävien järjestelmien puutteita tai toiminnallisuuksia pitkäaikaisesti. Jos RPA on toteutettu vain tähän tarkoitukseen, on vaarana, että RPA päätyy loppujen lopuksi osaksi ylläpidettävää vanhaa järjestelmää ja todennäköisesti tulee lisäämään ylläpitoaikaa. Myös on hyvä pitää mielessä, että vaikka RPA ratkaisee paljon ongelmia ja puutteita järjestelmissä, se ei siltikään poista järjestelmien ydinongelmia, mikäli niitä on. [10.]

2.4 Milloin ohjelmistorobotiikkaa kannattaa käyttää?

Ohjelmistorobotiikan kannattavuuteen vaikuttaa monta asiaa. Nämä vaikutukset ovat jokaiselle projektille yksilöllisiä. Siksi kannattavuus kannattaa selvittää kunnolla ennen projektin aloitusta. Näitä vaikuttavia asioita ovat:

- automatisoitavan liiketoimintaprosessin monimutkaisuus
- käytettävän datan lähteet ja monimutkaisuus
- käytettävien järjestelmien muuttuvaisuus
- prosessin odotettu elinikä
- prosessin toistuvuus ja kesto
- tarvittavan ylläpidon määrä
- ajoympäristön rajoitukset ja vaatimukset
- henkilöstön kyky hallita robotteja
- liiketoimintaprosessin raportointitarve
- prosessin suoritusajankohta
- henkilöstön korvaavat työt.

Automatisoitavan liiketoimintaprosessin monimutkaisuus ja datan lähteet vaikuttavat suuresti kannattavuuteen. Mitä monimutkaisempaa prosessia automatisoidaan, sitä enemmän kasvavat riskit muutoksille robotin toimintaan tulevaisuudessa. Näitä riskejä voidaan hallita erilaisilla sääntökoneilla tai valvontajärjestelmillä, kuten prosessijohtamisalustat (BPMS). Prosessijohtamisalustat sisältävät työkalut prosessijohtamisopin (BPM) toteuttamiseen, jossa prosessin valvominen ja analysointi ovat avainasemassa. Ohjelmistorobotiikassa on yleisesti käytetty niin kutsuttua ”viiden sääntöä” (eng. rule of five). Tällä viitataan siihen, että ohjelmistorobotin ei pitäisi tehdä enempää kuin viisi päätöstä, käyttää enempää kuin viittä järjestelmää ja koko prosessin pitäisi olla vähemmän kuin viisi sataa klikkausta. [11.] Datalähteissä optimitilanne on, että kaikki datalähteet ovat standardisoituja ja niiden formaatti pysyy samana.

Järjestelmien jatkuva muuttuminen tai päivittyminen voi aiheuttaa jatkuvaa ylläpitoa roboteille ja siksi se voi olla este RPA-projektin kannattavuudelle. Pitää muistaa, että ohjelmistorobotit ovat pohjimmiltaan vain skriptejä, joten ne eivät pysty reagoimaan dynaamisesti muutoksiin. [12.]

Ohjelmistorobotin elinikää pitää miettiä kahdelta kannalta. Ensinnäkin käytetäänkö robottia tarpeeksi kauan, että siitä saadaan tarpeeksi hyötyä kehityskustannuksiin nähden? Toiseksi, onko robotin elinkaari tarpeeksi lyhyt, että käsiteltäviin järjestelmiin ei tule niin paljon muutoksia, jotta robotin ylläpitokustannukset kasvavat yli tuotekehitysmuutoshinnan? Yksinkertaisia ja stabiileja järjestelmiä tai ohjelmia käyttävä robotti voi helposti pyöriä vuosia. Näitä ovat esimerkiksi datanrikastamis- ja muokkausrobotit. Monimutkaisesta robotista tai muuttuvia järjestelmiä automatisoivasta robotista sen sijaan voi pitkässä juoksussa olla enemmän taakkaa kuin hyötyä.

Kuten kaikissa ohjelmistoratkaisuissa, tietoturva, ylläpito ja hallinnointi ovat isossa osassa koko ratkaisua. Tietoturvaan tarvitaan suojattu ajoympäristö, rajatut oikeudet robotille ja omat tunnukset robotille. Tunnukset pitää tallentaa suojattuun paikkaan, eikä missään nimessä kovakoodattuna skripteihin. [13.] Robotin ylläpitoon ja hallintoon pitää varata resursseja, määrä riippuu siitä, kuinka kriittistä tehtävää robotti tekee liiketoiminnalle. Ohjelmistorobottien hallintaan ja monitorointiin kannattaa sijoittaa resursseja, sillä ne saattavat paljastaa pullonkauloja prosesseissa tai järjestelmissä. Myös monitoroinnilla saadaan nopeasti kiinni virhetilanteita, joita ei välttämättä osattu odottaa. [14.] Toki, on hyvin suositeltavaa, että ennen tuotantoon menoa ohjelmistorobotit testataan huolella.

Ohjelmistorobottien käytön ajankohta on myös syytä selvittää. Yleisesti robottien ajon ajankohta kannattaa sijoittaa toimistoaikojen ulkopuolelle, mikäli se on mahdollista. Tällöin järjestelmissä on vähiten kuormaa, mikä vähentää riskiä järjestelmien hidastumisille tai kaatumisille. Robottien ajaminen toimistoaikojen ulkopuolella vähentää myös riskiä sille, että robotit ja käyttäjät tekisivät päällekkäisiä muutoksia ja tällöin virhetilanteet datan suhteen vähenevät.

Suurin hyöty RPA-projekteissa saadaan irti säästetyistä työntekijöiden työajasta, joka menisi muuten tehdessä toistuvia mekaanisia toimia. Sen sijaan työntekijät voivat keskittyä haastavimpiin ja luovempiin tehtäviin. Siksi tämä

kannattaa selvittää tarkasti, jotta saadaan paras käsitys siitä, kuinka paljon projektista hyödytään. Taulukosta 1 saa pientä osviittaa, miten säästettyä aikaa voi laskea.

Taulukko 1. Kuinka paljon aikahyötyä ohjelmistorobotiikasta saa viidessä vuodessa? Taulukkoa luetaan riviotsikko kertaa sarakeotsikko kertaa viisi vuotta. Tulos kertoo, kuinka paljon aikaa säästyy noin viidessä vuodessa. Eli esimerkiksi toinen rivi (5 sekuntia) ja viides sarake (kuukausittain) on 5 sekuntia kertaa 12 kertaa 5 = 300 sekuntia eli 5 minuuttia.

KUINKA PALJON SÄÄSTÄT AIKAA	KUINKA USEIN TEET TEHTÄVÄÄ					
	50/PÄIVÄ	5/PÄIVÄ	PÄIVITTÄIN	VIIKOTTAIN	KUUKAUSITTAIN	VUOSITTAIN
1 SEKUNTTI	1 PÄIVÄ	2 TUNTIA	30 MINUUTTIA	4 MINUUTTIA	1 MINUUTTI	5 SEKUNTIA
5 SEKUNTIA	5 PÄIVÄÄ	12 TUNTIA	3 TUNTIA	21 MINUUTTIA	5 MINUUTTIA	25 SEKUNTIA
30 SEKUNTIA	4 VIIKKOA	3 PÄIVÄÄ	12 TUNTIA	2 TUNTIA	30 MINUUTTIA	2 MINUUTTIA
1 MINUUTTI	8 VIIKKOA	6 PÄIVÄÄ	1 PÄIVÄ	4 TUNTIA	1 TUNTI	5 MINUUTTIA
5 MINUUTTIA	9 KUUKAUTTA	4 VIIKKOA	6 PÄIVÄÄ	21 TUNTIA	5 TUNTIA	25 MINUUTTIA
30 MINUUTTIA		6 KUUKAUTTA	5 VIIKKOA	5 PÄIVÄÄ	1 PÄIVÄ	2 TUNTIA
1 TUNTI		10 KUUKAUTTA	2 KUUKAUTTA	10 PÄIVÄÄ	2 PÄIVÄÄ	5 TUNTIA
6 TUNTIA				2 KUUKAUTTA	2 VIIKKOA	1 PÄIVÄ
1 PÄIVÄ					8 VIIKKOA	5 PÄIVÄÄ

Vaikka suurin hyöty saadaankin työntekijöiden säästetystä työajasta, kannattaa selvittää ajoissa ennen projektin toteuttamista, mihin tehtäviin työntekijät voivat käyttää vapautuneen aikansa.

Kun kaikki yllä olevat seikat otetaan huomioon, kannattava RPA-projekti sisältää sääntöpohjaisen prosessin, jota automatisoidaan. Sääntöpohjaisella prosessilla tarkoitetaan prosessia, jonka toiminta on rajattu toimimaan tiettyjen sääntöjen mukaan. Yksinkertainen tilakone on hyvä esimerkki sääntöpohjaisesta prosessista. RPA-projektissa sääntöpohjaisen prosessin ja siinä käytettävien järjestelmien on oltava stabiileja eli niihin ei odoteta suuria muutoksia lähiaikoina. Prosessin on myös toimittava sähköisesti ja sen on oltava usein käytetty. Lisäksi automatisoitavan prosessin on hyvä käyttää useampaa järjestelmää, sillä se lisää automatisoinnin kannattavuutta. Mutta useampaa kuin viittä järjestelmää käyttävien prosessien automatisointi on usein liian kompleksista, joten niiden automatisointia ei suositella. On kannattavaa, että prosessia manuaalisesti suorittaville henkilöille on varattu muita hyödyllisempiä ja vaativampia tehtäviä, joihin heidät voidaan siirtää, kun ohjelmistorobotti otetaan käyttöön. Viimeisenä seikkana kannattavan automatisoitavan prosessin elinkaaren on oltava joko rajallinen tai RPA-projektissa on varauduttava ylläpito- ja hallintomenoihin.

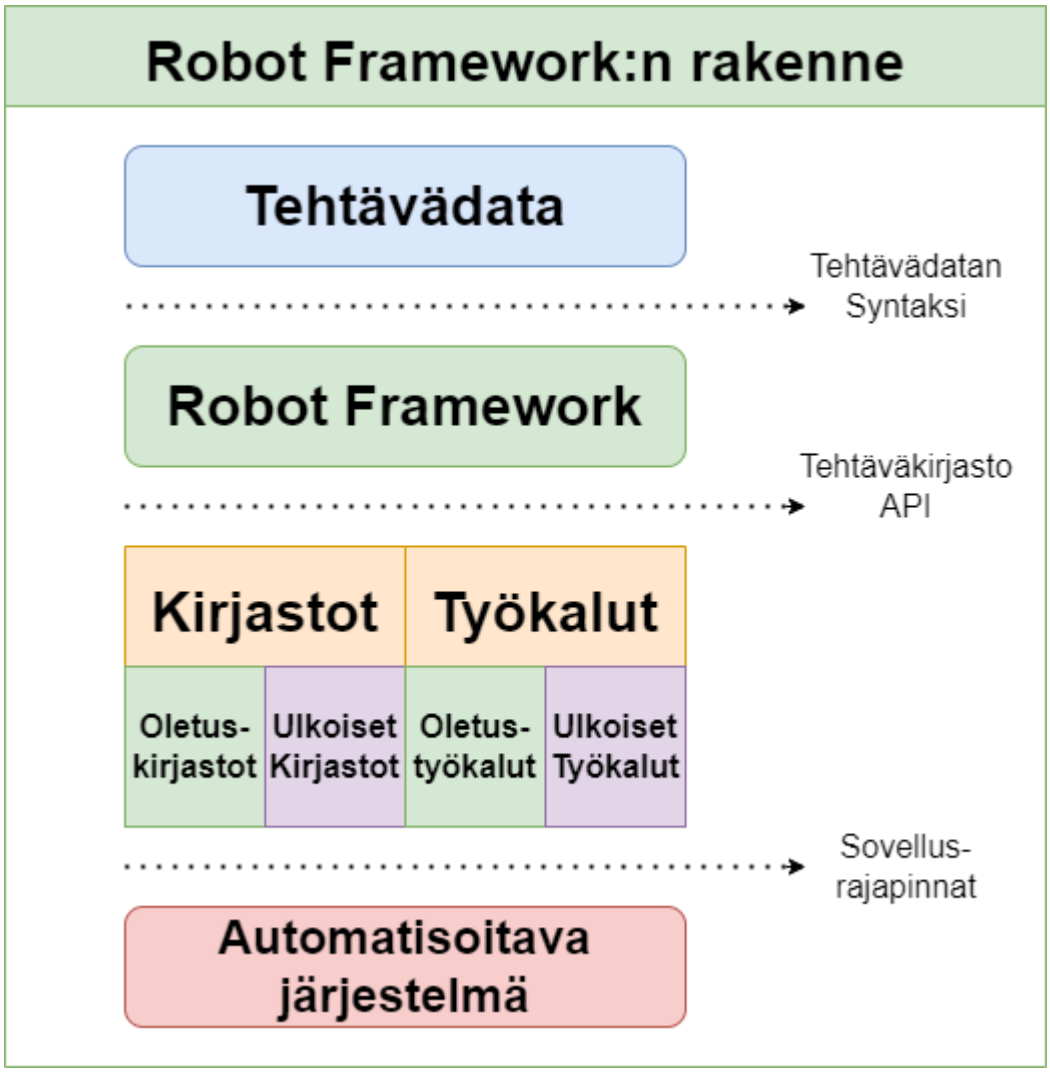
3 Robot Framework

Tässä luvussa käydään läpi, mikä Robot Framework on, miten ja mihin sitä käytetään ja sen hyödyt RPA-projekteissa. Aluksi käydään läpi yleistietoa Robot Frameworkistä ja sen historiasta. Tämän jälkeen pureudutaan tarkemmin Robot Frameworkin moduuleihin, varsinkin niihin, joita yleisesti käytetään ohjelmistorobotiikan toteutuksissa. Lopuksi katsotaan Robocorpin tuoretta RPA Framework -moduulia, joka kasaa yleisimmät RPA-projekteissa käytetyt moduulit ja on kehittänyt niiden käyttöä helpottavia avainsanoja.

3.1 Mikä on Robot Framework?

Robot Framework on geneerinen avoimeen lähdekoodiin perustuva kehys. Sitä voi käyttää testiautomaatioon ja ohjelmistorobotiikkaan. [15.] Robot Frameworkin kehitys alkoi Pekka Klärckin (entinen Laukkanen) diplomityöstä 2005, ja sen ensimmäinen versio kehitettiin Nokia Networksissä samana vuonna. [16; 15.] Vuonna 2008 projekti julkistettiin avoimeksi lähdekoodiksi GitHub:iin versiona 2.0. [15;17.] Robot Frameworkin ylläpitoa ja kehitystä sponsoroi Robot Framework Foundation. [18.]

Robot Framework on modulaarinen kehys, joten se on helposti laajennettavissa omilla kirjastoilla ja siihen voi integroida käytännössä minkä tahansa toisen työkalun. Tämä mahdollistaa tehokkaiden ja joustavien automaatioratkaisuiden kehittämisen. Robot Framework on kirjoitettu Python-kielellä ja sitä voi ajaa myös Jythonilla (JVM) ja IronPythonilla (.NET). [15.] Kuvasta 2 saa yleiskuvaa, miten Robot Frameworkin rakenne voi muodostua ohjelmistorobotiikkaprojekteissa.



Kuva 2. Esimerkki Robot Frameworkin rakenteesta ohjelmistorobotiikkaprojektissa.

Kuva 2 kuvaa aika yleistä rakennetta ohjelmistorobotin rakenteesta. Tästä esimerkkinä on robotti, joka syöttää tietoja verkkolomakkeelle ja hyväksyy lomakkeen. Kuvassa tehtävädata on dataa, jota käytetään robotin toiminnassa tai jonka perusteella robotti tekee automatisoitavan tehtävän. Robot Framework toimii tässä robotin yleiskehyksenä, jota kirjastot ja työkalut laajentavat tehtäväkirjasto API:n kautta. Kirjastot ja työkalut yhdistyvät automatisoitavaan järjestelmään sovellusrajapintojen kautta ja suorittavat tehtävänsä sisään luetun tehtävädatan perusteella.

Robot Frameworkin laajentaminen onnistuu luontevasti Python-moduuleilla ja Python-luokilla, koska Robot Framework on toteutettu Pythonilla. Laajennuksia voi tehdä myös Java-luokilla, jos kehystä ajetaan Jythonilla. [19.] Robot Framework käyttää omaa tiedostotyyppiä ”robot”. Robot-tiedostot sisältävät testi- tai automaatiotapaukset. Laajennukset tuodaan robot-tiedostoihin kirjastoina. Robot-tiedostoihin on myös mahdollista tuoda ulkoisia muuttujatiedostoja, jotka ovat tyypillisesti Python-moduuleita. Omia avainsanakirjastoja voi myös tuoda robot-tiedostoihin. [19.] Näistä enemmän seuraavissa luvuissa ja esimerkeissä.

Robot Frameworkillä on helppo syntaksi, joka perustuu ihmiselle ymmärrettäviin avainsanoihin. Kehyksen syntaksissa on paljon samaa kuin Pythonissa, mutta suurin ero on, että robot-tiedostot sisältävät taulukkotyyppisen syntaksin. Nämä taulukot ovat:

- **Settings**: Tässä osiossa tuodaan kirjastot, resurssitiedostot sekä muuttujatiedostot. Osiossa voidaan myös kuvata metadata testi- ja tehtäväsarjoille ja tapauksille.
- **Variables**: Määritetään muuttujat, joita voidaan käyttää muualla testidatassa.
- **Test Cases**: Luodaan testitapaukset saatavilla olevista avainsanoista.
- **Tasks**: Luodaan tehtäviä saatavilla olevista avainsanoista. Yhdessä robot-tiedostossa voi olla joko testejä tai tehtäviä. Keskitymme tässä työssä tehtäviin, mutta niistä enemmän myöhemmin.
- **Keywords**: Osio omien avainsanojen luontiin.
- **Comments**: Lisäkommentteja tai dataa. Robot Framework jättää nämä huomiomatta.

Seuraavalla sivulla (esimerkkikoodi 1) on hyvin yksinkertainen esimerkki robot-tiedoston rakenteesta:

```
*** Settings ***
Library     String
Resource    keywords/example_keywords.resource
Variables   variables/example_variables.py

*** Variables ***
${EXAMPLE_VAR}      Example text!

# Only Test Cases or Tasks can be in one robot file!
# *** Test Cases ***

*** Tasks ***
Example Task
    Log Example To Console

*** Keywords ***
Log Example To Console
    Log To Console    ${EXAMPLE_VAR}
```

Esimerkkikoodi 1. Yksinkertainen esimerkki Robot Frameworkin syntaksista robot-tiedostossa.

3.2 Robot Frameworkin käyttö RPA-projekteissa

Robot Framework sopii hyvin RPA-projektien tekoon, varsinkin kun toteuttajalla on ohjelmointitaustaa. Robot Frameworkin etuina ovat vähäiset kustannukset, sisäänrakennettu raportointi, kustomoinnin ja laajentamisen helppous ja suuri määrä työkaluja automaatiota varten.

Vähäiset kustannuksen tulevat siitä, että Robot Framework on avoimen lähdekoodin projekti. Tämä tarkoittaa sitä, että sitä voi käyttää kuka tahansa ilman lisenssikustannuksia. Avoimen lähdekoodin ongelmana taas on, että sille ei ole taattua tukea kuten suljetun lähdekoodin tuotteissa yleisesti on. Onneksi Robot Frameworkillä on suuri yhteisö ja Robot Foundation tuki Robot Frameworkin ylläpidolle ja kehitykselle.

Robot Frameworkin sisään rakennettu raportointi toimii hyvin suorituksen seurannassa ja raportoinnissa. Generoiduista loki- ja raporttidatasta voi myös tehdä omia raportteja. Kuten aikaisemmassa luvussa käytiin läpi, Robot Frameworkiä on helppo laajentaa omilla kirjastoilla, mikä antaa hyvin joustavat mahdollisuudet toteuttaa ohjelmistorobotti, kuten tarve vaatii. Yleisemmin tarvittavat työkalut

löytyvät jo laajasta valikoimasta ylläpidettyjä kirjastoja. Mikäli halutusta kirjastosta ei löydy haluttua ominaisuutta, niin halutun toiminnallisuuden voi toteuttaa ja lähettää arvosteltavaksi siten, että se voidaan lisätä avoimen lähdekoodin projektiin hyväksynnän jälkeen.

Robot Framework -robotteja pystyy kehittämään tekstieditorista ohjelmistokehitysalustoihin. Monilla ohjelmistokehitysalustoilla on tuki tai laajennus tukemaan Robot Frameworkiä ja näiden ansiosta robottien kehittäminen on helppoa. Suurimman osan kolmannen osapuolen Robot Framework -kirjastot ovat avointa lähdekoodia ja on saatavilla Pythonin paketti-indeksistä (PyPI). Lisää näistä kirjastoista on seuraavaksi.

3.3 Robot Frameworkin moduulit ja työkalut

Kuten luvussa 3.1 mainittiin, Robot Framework on modulaarinen kehys, eli suurin osa ominaisuuksista on toteutettu kirjastoina moduuleiksi kehyksen päälle. Yleiset kirjastot ja työkalut tulevat kehyksen mukana. Näitä yleisiä kirjastoja kutsutaan **oletuskirjastoiksi**. [20.] Näistä käytännöllisimmät ohjelmistorobotiikalle ovat:

- **Builtin**: Tarjoaa joukon yleisesti tarvittavia geneerisiä avainsanoja. Avainsanat ovat aina saatavilla ilman kirjaston tuontia.
- **Collections**: Tarjoaa joukon avainsanoja Python-listojen ja sanakirjojen hallintaan.
- **DateTime**: Kirjasto päivämäärien ja ajan muunnoksiin.
- **Dialogs**: Tarjoaa avainsanat suorituksen tauottamiseen ja käyttäjäsyötteen hankintaan.
- **OperatingSystem**: Kirjasto, joka mahdollistaa monenlaisten käyttöjärjestelmään liittyvien tehtävien suorituksen siinä järjestelmässä, jossa Robot Framework on käynnissä.
- **Process**: Kirjasto prosessien ajamiseen järjestelmässä.
- **Screenshot**: Tarjoaa avainsanat kuvakaappauksien ottamiseen ja tallentamiseen.
- **String**: Kirjasto merkkijonojen muokkaamiseen ja sisällön varmentamiseen.
- **XML**: Kirjasto XML-dokumenttien käsittelyyn.

Oletuskirjastojen mukana tulee myös kirjastot **Remote** ja **Telnet**, mutta näitä harvemmin käytetään. Remote-kirjasto tarjoaa rajapinnan käyttää kirjastoja, jotka ovat muualla, esimerkiksi toisella koneella. Nämä kirjastot voidaan kirjoittaa kielillä, jotka tukevat XML-RPC-protokollaa. [21.] Yleisesti käytettävät kirjastot on hyvä olla ladattuna ajoympäristössä, mutta mikäli tämä ei ole mahdollista, Remote-kirjasto mahdollistaa etäkäytön. Telnet-kirjasto mahdollistaa yhteyden avaamisen Telnet-palvelimiin ja suorittaa komentoja avatulla yhteydellä. [21.] Telnet-palvelimia on vielä paljon käytössä, mutta on suositeltavaa käyttää Secure Shell (SSH) -yhteyksiä tietoturvan takia, mikäli mahdollista.

Kehyksen mukana tulee neljä yleistä työkalua, jotka ovat erillisiä ohjelmia suoritusohjelman rinnalla. [20.]

- **Rebot**: Työkalu lokien ja raporttien generointiin ja yhdistämiseen suorituksista syntyvistä XML-tulosteista.
- **Libdoc**: Työkalu avainsanadokumentaation generointiin kirjastoille ja resurssitiedostoille.
- **Testdoc**: Työkalu, jolla voi generoida korkeantason dokumentaation HTML-muodossa Robot Framework testitapauksista.
- **Tidy**: Robot Framework-tiedostojen formaatin putsaamisen ja päivittämisen tekevä työkalu.

Robot Framework tukee myös ulkoisia, kolmannen osapuolen tekemiä kirjastoja ja laajennuksia. Suurin osa ulkoisista kirjastoista ovat tehty testausta varten, mutta yleisesti niitä voi käyttää myös ohjelmistorobotiikkaan. Näistä oleellisempia ohjelmistorobotiikalle ovat:

- **DataDriver**: DataDriver on datavetoinen laajennus, joka mahdollistaa datan lukemisen erilaisista tietolähteistä. Laajennus alustavasti tukee datan lukemisen .csv-, .xls- ja .xlsx-tiedostoista, mutta mahdollistaa myös muiden tietolähteiden lukemisen kustomoiduilla Python lukijaluokilla. Laajennus otetaan käyttöön kuten kirjastot. [22.]
- **SeleniumLibrary**: Verkkotestauskirjasto, joka käyttää Selenium-kehystä verkkosivustojen automatisointiin. [23.]
- **Browser Library**: Verkkotestauskirjasto, joka käyttää Playwright-kehystä verkkosivujen automatisointiin. [24.]
- **RequestsLibrary**:Kirjasto, joka tarjoaa toiminnallisuuksia HTTP-ohjelmointirajapintatestaukseen käärimällä Python:in Requests-kirjaston. [25].

- **AppiumLibrary**: Kirjasto, joka tarjoaa toiminnallisuudet kommunikoida Android- ja iOS-applikaatioiden kanssa käyttämällä Appium:ia. [26.]
- **Database Library**: Tietokantatyökaluja tarjoava kirjasto. Toimii tietokantojen kanssa, jotka toteuttavat *Python Database API Specification v2.0*. [27.]
- **SSHLibrary**: Testikirjasto SSH:n ja SFTP:n käyttöön. [28.]
- **Zoomba Library**: Kirjastokokoelma käyttöliittymä, REST-rajapinta ja SOAP-rajapinta automatisointiin. [29.]
- **Robotframework-FlaUI**: Avainsanapohjainen käyttöliittymäautomaatiokirjasto Windows sovelluksille. Pohjaantuu FlaUI-kirjastoon. [30.]
- **ArchiveLibrary**: Kirjasto zip- ja tar-arkistojen käsittelyyn. [31.]
- **SoapLibrary**: Kirjasto suunniteltu käyttäjille, jotka haluavat toimia verkkopalvelujen automatisoinnissa kuin he käyttäisivät SoapUI:ta eli lähettää kutsuja käyttämällä XML-tiedostoja ja saada vastaukseksi XML-tiedoston. [31.]
- **FTP Library**: FTP-asiakasohjelman toiminnallisuuksia tarjoava kirjasto. FTP-viestintää hoitaa *ftplib*. [32.]

Robot Frameworkille on tehty paljon ulkoisia kirjastoja erilaisiin tarpeisiin, ja ne ovat yleisesti avointa lähdekoodia. Robot Frameworkille on myös tehty sen toimintaa tukevia ja laajentavia ulkoisia työkaluja. Ohjelmistorobotiikalle hyödyllisimpiä ovat:

- **Pabot**: Robot Framework testien- ja tehtävien-rinnakkaissuorituksen mahdollistava työkalu. [31.]
- **Robocop**: Staattinen koodianalyysityökalu Robot Frameworkille. Käyttää viimeisintä robot-ohjelmointirajapintaa ja monia sisäänrakennettuja sääntöjä, joita voi helposti konfiguroida tai asettaa pois päältä. [31.]
- **Jenkins plugin**: Liitännäinen, jolla kerätään ja julkaistaan Robot Frameworkn suorituksen tulokset Jenkins:ssä.
- **DbBot**: Työkalu sarjallistaa Robot Frameworkn suoritusten tulokset SQLite-tietokantaan. Tämän tarkoitus on tarjota hyvä pohja omien raportointi- ja analyysityökalujen kehittämiselle. [31.]
- **RobotMK**: Työkalu, joka integroi Robot Frameworkn tulokset Checkmk-monitorointijärjestelmään. [31.]
- **Fixml**: Työkalu Robot Frameworkn rikkinäisten tulostiedostojen korjaamiseen. [31.]

Kuten aikaisemmin on mainittu, Robot Frameworkille on tehty paljon kirjastoja ja työkaluja erilaisiin tarpeisiin, mutta mikäli tietylle käyttötarkoitukselle löydy sopivaa valmista kirjastoa, on mahdollisuus tehdä omia kustomoituja kirjastoja. [33.] Näitä kirjastoja voi tehdä kielillä, jotka mainittiin luvussa 3.1.

Robot Frameworkissä on kolme erilaista testikirjasto-ohjelmointirajapintaa. Nämä ovat staattiset ja dynaamiset ohjelmointirajapinnat sekä hybridiohjelmointirajapinnat. Staattinen kirjasto on yksinkertaisimmillaan Python-moduuli tai Python- tai Java-luokka, jonka metodien nimet on määritelty avainsanoiksi. Staattisessa kirjastossa avainsanat ottavat vastaan samat argumentit kuin niitä implementoivat metodit moduuleissa tai luokissa. Avainsanat raportoivat epäonnistumisen virheillä, lokitus suoritetaan kirjoittamalla oletustulosteeseen ja avainsana voi palauttaa arvoja käyttämällä palautuslauseketta. [34.] Staattisesta kirjastosta ja sen käytöstä on yksinkertainen esimerkki esimerkkikoodissa 2.

```
# example_static_library.py
def print_name(first_name, last_name):
    name = "{} {}".format(first_name, last_name)
    print("Name is {}.".format(name))
    return name

# Robot file
*** Settings ***
Library     example_static_library.py    # Import the module

*** Tasks ***
Example Task
    ${NAME} =   Print Name    Test  Tester
    Log    ${NAME}
```

Esimerkkikoodi 2. Yksinkertainen esimerkki staattisesta kirjastosta ja sen käytöstä Robot Frameworkissä. Tässä tapauksessa staattinen kirjasto on Python-moduuli.

Dynaaminen kirjasto on luokka, joka implementoi metodin, jolla haetaan niiden avainsanojen nimet, joita luokka implementoi, ja toisen metodin, joka suorittaa nimetyn avainsanan annetuilla argumenteilla. Implementoitavien avainsanojen nimet sekä suoritustavat voidaan määrittää dynaamisesti suoritusaikana, mutta niiden tilan raportointi, lokitus ja arvojen palautus tehdään samanlailla kuin staattisessa kirjastossa. [34.]

Yksinkertainen esimerkki dynaamisen kirjaston rakenteesta on esimerkkikoodissa 3.

```
# Dynamic library example
from robot.api.deco import keyword

class DynamicLibraryExample:

    def get_keyword_names(self):
        # Get all the library's attributes and their values.
        attributes = [(name, getattr(self, name)) for name in
                      dir(self)]

        # Remove attributes that don't have 'robot_name' set.
        keywords = [(name, value) for name, value in attributes
                    if hasattr(value, 'robot_name')]

        # Return 'robot_name', if it's given, or the original 'name'.
        return [value.robot_name or name for name, value in keywords]

    def run_keyword(self, name, args, kwargs):
        print("Keyword '%s' with positional arguments %s and named ar-
               guments %s is running." % (name, args, kwargs))

    @keyword
    def keyword_example_method(self):
        # This is keyword used in robot.
```

Esimerkkikoodi 3. Esimerkki dynaamisen kirjaston rakenteesta. Kirjasto toteuttaa metodin, joka palauttaa avainsanojen nimet (get_keyword_names) ja metodin, joka suorittaa annetun avainsanan annetuilla argumenteilla (run_keyword). Annotaatio *keyword* tulee kirjastosta *robot.api.deco*, joka asettaa metodille attribuutin *robot_name*. Tällä attribuutilla esimerkissä tunnistetaan avainsanametodit. Esimerkki on rakennettu Robot Frameworkin käyttöoppaan esimerkeistä. [35.]

Hybridikirjasto on nimensä mukaan hybridi staattisen ja dynaamisen kirjaston välillä. Kirjasto on luokka, joka sisältää metodin, joka kertoo, mitä avainsanoja luokka implementoi, mutta nämä avainsanat täytyy myös olla käytettävissä suoraan. Kaikki muu on samaa kuin staattisessa kirjastossa paitsi tapa, millä löydetään, mitkä avainsanat on implementoitu. Tästä esimerkki esimerkkikoodissa 4.

```
from external_library import example_keyword

class HybridLibraryExample:

    def get_keyword_names(self):
        return ['my_example_keyword', 'example_keyword']

    def my_example_keyword(self, arg):
        print("My Example Keyword was called with '%s'" % arg)

    def __getattr__(self, name):
        if name == 'example_keyword':
            return example_keyword
        raise AttributeError("Didn’t find attribute '%s'" % name)
```

Esimerkkikoodi 4. Yksinkertainen esimerkki hybridikirjastosta. Esimerkki on muokattu versio Robot Frameworkin käyttöoppaan esimerkistä. [36.]

On myös olemassa kirjastoja, jotka kasaavat erilaisia kirjastoja yhdeksi kokonaisuudeksi ja jopa muokkaavat niitä yleisen käyttötarkoituksen mukaan. Ohjelmistorobotiikalle on tehty vastaava kirjastokokonaisuus, jota käydään läpi tarkemmin seuraavassa alaluvussa.

3.4 RPA Framework

RPA Framework on kokoelma avoimen lähdekoodin kirjastoja ja työkaluja ohjelmistorobotiikkaan liittyen. Se on suunniteltu käytettäväksi Robot Frameworkillä ja Pythonilla. Projektin tavoite on tarjota hyvin dokumentoituja ja aktiivisesti ylläpidettyjä ydinkirjastoja ohjelmistorobotiikkakehittäjille. Projektia sponsoroi Robocorp.

Tämä kokoelma sisältää kirjastoja erilaisten käyttöliittymien automatisointiin, datamanipulointiin, eri tiedostotyyppien, tietokantojen ja pilvipalvelujen hallintaan. Kokoelmassa on muitakin kirjastoja, mutta pääasiana on, että kokoelman kirjastoilla voi tehdä suurimman osan ohjelmistoroboteista. Kokoelma myös sisältää käärekirjastoja, jotka laajentavat yleisiä kirjastoja, jotta niiden käyttö palvelisi paremmin ohjelmistorobotteja. Tarkemmat ja päiväntasaiset tiedot kirjastoista löytyvät RPA Frameworkin dokumentaatiosivuilta. [37.]

4 Mahdollisia RPA-toteutuksia

Tässä luvussa käydään läpi kolme hypoteettista esimerkkitapausta, joissa ohjelmistorobotiikkaa voidaan käyttää. Esimerkeissä näytetään vain robotiikkaan liittyvät koodit ja tarkan logiikan tai datan keruun oletetaan tapahtuvan muissa tiedostoissa tai kirjastoissa. Esimerkeillä pyritään näyttämään mahdollisia toteutuksia RPA-projekteista ja robottien rakenteesta Robot Frameworkiä käyttäen.

4.1 Tiedostojen rikastaminen kahden järjestelmän välissä

Yritys käyttää kahta erillistä sovellusohjelmaa yrityksen toimintaan. Sovellusohjelmat ovat vanhoja, eikä niillä ole yhteyttä julkiverkkoon tai toisiinsa. Yritys on kehittämässä täysin uutta nykyaikaista sovellusta korvaamaan vanhat sovellukset. Uusi järjestelmä käyttää uutta tietorakennetta, joka koostuu vanhojen järjestelmien tietorakenteista.

Tarkoitus oli tehdä kertaluontainen datamigraatio, kun uusi sovellus olisi valmis. Viivästysten takia kaikki tarvittavat toiminnallisuudet eivät olleet valmiit, kun uusi sovellus otettiin käyttöön. Tämä aiheutti ongelman päivittäisen datamigraation kanssa, kun osa henkilöstön tehtävistä tehtiin vielä vanhoilla sovelluksilla.

Yritys päätti toteuttaa datamigraation valvotulla ohjelmistorobotilla helpottaakseen henkilöstön päivittäistä toimintaa. Robotin toiminta määriteltiin näin:

- Vanhoista sovelluksista viedään edellisen päivän tapahtumat CSV-tiedostoihin virtuaaliympäristöön syöttöhakemistoon.
- Robottia säilytetään ja ajetaan samassa virtuaaliympäristössä, johon CSV-tiedostot siirretään. Robotti käynnistetään manuaalisesti.
- Robotti hakee nämä tiedostot ja muokkaa näistä yhdistetyn rikastetun CSV-tiedoston.
- Muokkauksen jälkeen robotti jää odottamaan käyttäjän varmistusta, että rikastaminen on tehty oikein.
- Kun varmistus on annettu, robotti siirtää tiedoston määritettyyn paikkaan.

```
*** Settings ***
Library    RPA.Tables
Library    RPA.Dialogs
Library    Collections
Library    OperatingSystem

*** Variables ***
${INPUT_1_PATH}    path/to/input/data1.csv
${INPUT_2_PATH}    path/to/input/data2.csv
${OUTPUT_PATH}     path/to/output/combined_data.csv
# List of the output file headers
@{OUTPUT_HEADERS}  field1    field2   field3   field4
# The variable used to identify rows to be combined
${ID_FIELD}        field1

*** Tasks ***
Data Migration
    Read Input Files And Combine Them
    Wait For Confirmation
    Move Combined File To Output Folder

*** Keywords ***
Read Input Files And Combine Them
    ${DATA1}=    Read table from CSV    ${INPUT_1_PATH}
    ${DATA2}=    Read table from CSV    ${INPUT_2_PATH}
    # Merge data sets.
    ${MERGED_DATA}=    Merge Tables    ${DATA1}    ${DATA2}    in-
dex=${ID_FIELD}
    ${RESULT_DATA}=    Create Table    ${EMPTY}    ${False}    ${OUT-
PUT_HEADERS}
    # Clear memory if dealing with large files
    ${DATA1}=  Set Variable    ${None}
    ${DATA2}=  Set Variable    ${None}

    # Go through merged table rows and
    # add them to a new table with correct structure.
    FOR    ${row_data}   IN    @{MERGED_DATA}
       Add Table Row   ${RESULT_DATA}    ${row_data}
    END
    Write table to CSV    ${RESULT_DATA}    result.csv

Wait For Confirmation
    ${DIALOG_TEXT} =  catenate  SEPARATOR=
    ...  File merge was successful!
    ...   Please confirm that data is correct.
    ...  If there were changes to the merged file,
    ...   please upload modified file bofore proceeding.
    Add Icon      Success
    Add Heading   ${DIALOG_TEXT}
    Add File      ${CURDIR}/result.csv   Merged file
    Add File Input  Modified file  Modified file  ${CURDIR}  ${CURDIR}
    Add Submit Buttons    buttons=Yes    default=Yes
    Run dialog

Move Combined File To Output Folder
    Move File   ${CURDIR}/result.csv    ${OUTPUT_PATH}
```

Esimerkkikoodi 5. Esimerkki mahdollisesti toteutuksesta luvun 4.1 ohjelmistorobotiikkaprojektiin.

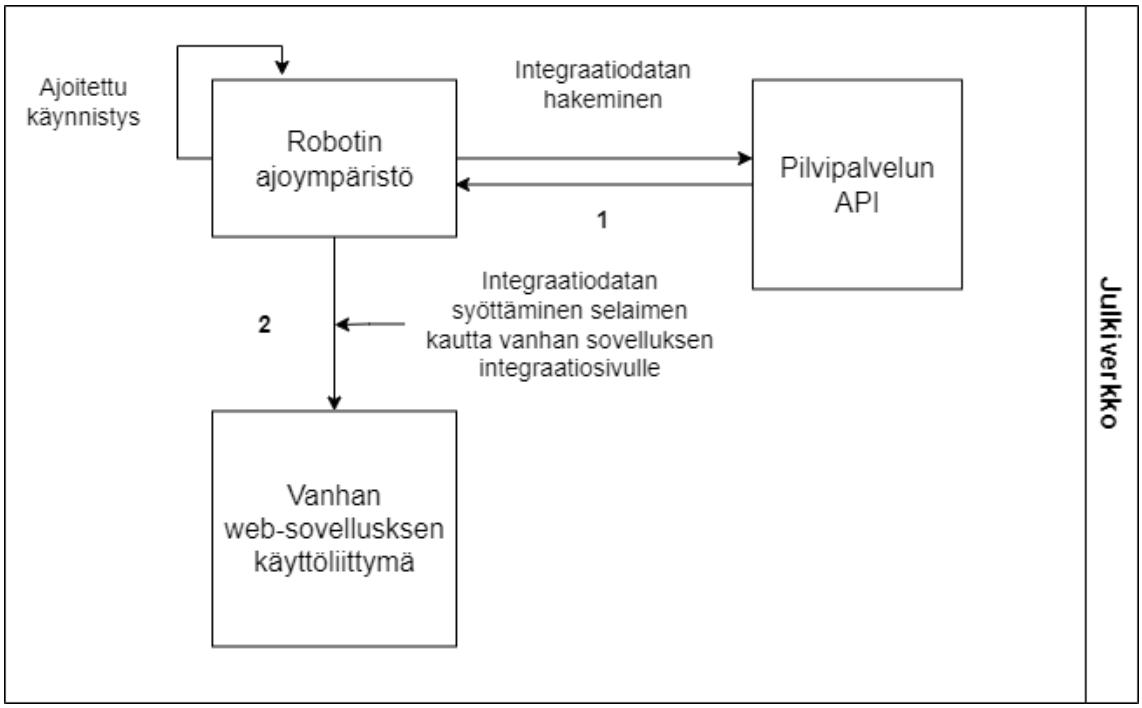
Aikaisemman sivun esimerkissä (esimerkkikoodi 5) *Variables*-osiossa asetetaan robotin muuttujat, joita käytetään avainsanoissa. Robotin suoritus alkaa lukemalla ja yhdistämällä vanhoista järjestelmistä viedyt CSV-tiedostot. Yhdistämisen jälkeen robotti varmistaa, että taulukkomuotoisen datan sarakkeet ovat halutussa järjestyksessä. Tämän jälkeen robotti näyttää käyttäjälle dialogi-ikkunan ja jää odottamaan käyttäjän manuaalista validointia. Käyttäjä voi ladata yhdistetyn datan itselleen ja tehdä muokkauksia dataan ja asettaa muokatun datan samaan dialogiin. Kun validointi on suoritettu, robotti siirtää tiedoston määriteltyyn hakemistoon. Esimerkissä oletetaan, että data on eheää eikä käyttäjä syötä muuta tiedostoa kuin CSV-tiedoston, jos muutoksia on tehty. Toteutus toteuttaa määritetyt toiminnallisuudet, kun tarvittavat muuttujat annetaan robotille. Kuten esimerkistä huomataan, suurin osa logiikasta tapahtuu jo valmiina olevissa kirjastoissa, mikä tekee robottien tekemisestä helpompaa.

4.2 Tietojen hakeminen rajapinnasta ja niiden syöttö järjestelmään, jossa on vain käyttöliittymärajapintana

Yritys oli tilannut vuosia aikaisemmin web-sovellustoteutuksen ohjelmistotalolta hallitakseen asiakastietoja ja tilauksia. Yritys on saanut tarjouksen uudesta pilvipalvelusta, joka on moderni ja on toiminnallisuuksiltaan lähes vastaava aikaisempaan palveluun. Sen ylläpitokustannukset ovat huomattavasti pienemmät kuin nykyisen ratkaisun.

Aikaisempi ohjelmistotalo on antanut tarjouksen nykyisen järjestelmän uudistamisesta, mutta se on hyvin kallis ja arvioitu toteutusaika on pitkä. Pilvipalvelusta uupuu kriittinen integraatio, jonka palveluntarjoaja on ilmoittanut toteuttavansa kalenterivuoden sisään. Pilvipalvelu tarjoaa ohjelmointirajapinnat, joista voi hakea tai syöttää asiakastietoja. Vastaavia rajapintoja ei ole nykyisessä ratkaisussa.

Yritys päättää siirtyä käyttämään pilvipalvelua ja toteuttaa ohjelmistorobotin siksi aikaa, kunnes kriittinen integraatio on implementoitu. Yritys laatii ohjelmistorobotille rakennekaavan (kuva 3), jossa kuvataan robotin toiminta ylätasolla.



Kuva 3. Tämän luvun ohjelmistorobottiprojektin rakennekuvaus.

Robotti hakee rakennuskuvauksen (kuva 3) mukaan tietyn aikajakson asiakastiedot pilvipalvelun ohjelmointirajapinnasta ja syöttää ne aikaisemman palvelun integraatioon selaimen kautta, koska integroidulla palvelulla ei ole ulkoisia rajapintoja. Ohjelmistorobotin toiminta määritellään näin:

- Robotti toimii valvomattomasti, ja se ajastetaan käynnistyvän tietyn aikavälein.
- Robotille annetaan rajatut oikeudet rajapintoihin ja rajattu pääsy vanhaan järjestelmään.
- Robotti hakee pilvipalvelusta rajapintakutsulla tarvittavat tiedot.
- Robotin syöttää kaikki haetut tiedot vanhaan järjestelmään selaimen kautta.
- Virhetilanteissa robotti siirtyy syöttämään seuraavaa datasettiä.
- Raportit säilötään konfiguroituun hakemistoon. Raporttien pitää olla aikaleimattuja.
- Virhetilalanteissa robotti lähettää ilmoitussähköpostin konfiguroituun sähköpostiin.

Esimerkkitoteutus koostuu useammasta tiedostosta, itse robottitiedostosta, erillisestä muuttujatiedostosta, Robot Framework-DataDriver -kirjaston lukijasta ja käynnistysskriptistä. Alla DataDriver-kirjaston lukija (esimerkkikoodi 6):

```
from DataDriver.AbstractReaderClass import AbstractReaderClass
from DataDriver.ReaderConfig import TestCaseData
import requests


class api_data_parser(AbstractReaderClass):

    def get_data_from_source(self):
        task_data = []
        # First authenticate and get the token to get the actual data
        token = self.handle_auth(self.kwargs['api_auth_url'],
                                 self.kwargs['api_user'],
                                 self.kwargs['api_pwd'])
        # Get the task data
        json = self.get_json_data(token, self.kwargs['api_url'])
        for data in json["data"]:
            task_data.append(TestCaseData('Input values for {}'
                                          .format(data["id"]), data))
        return task_data

    def handle_auth(self, auth_url, user, pwd):
        # Add headers and such if required.
        response = requests.post(auth_url, data={'username': user,
                                                 'password': pwd})
        json = response.json()
        return json["access_token"]

    def get_json_data(self, token, api_url):
        # Add to headers correct content type.
        headers = {"authorization": "Bearer {}".format(token)}
        response = requests.get(api_url, headers=headers)
        return response.json()
```

Esimerkkikoodi 6. Tämän luvun Esimerkkitoteutuksen **Robot Framework–DataDriver** -kirjaston kustomoitu lukija *api_data_parser.py*. Lukijassa haetaan todennustunniste ohjelmistorajapintoja varten ja sen jälkeen haetaan syötettävät tiedot. Nämä tiedot käydään läpi ja jaetaan osiin. Tällä mahdollistetaan kaikkien tietojen syöttämisen yrittäminen, mikäli virhetilanteita tulee vastaan. Jako myös helpottaa raportointia.

```
*** Settings ***
Resource    api_robot_variables.resource
Library    DataDriver    reader_class=${CURDIR}/api_data_parser.py
...                      file_search_strategy=None
...                      api_url=${API_URL}
...                      api_auth_url=${API_AUTH_URL}
...                      api_user=${API_USER}
...                      api_pwd=${API_PWD}
Library    RPA.Email.Exchange
Library    RPA.Browser.Playwright
Task Template    Handle Migration
Suite Teardown   Run Keyword If Any Tests Failed   Send Email Notifi-
cation

*** Tasks ***
Fetch Data From API And Insert It To Old System

*** Keywords ***
Handle Migration
    [Arguments]    ${data}
    Navigate To Old System And Input Info

Navigate To Old System And Input Info
    [Arguments]    ${data}
    New Browser    chromium    headless=false
    New Context    viewport={'width': 1920, 'height': 1080}
    New Page    ${UI_URL}
    Handle Login
    Navigate To Integration View
    Fill Info And Save    ${data}

Handle Login
    # TODO
    Fill Text     usernameField   ${UI_USER}
    Fill Secret   passwordField   ${UI_PWD}
    Click  login

Navigate To Integration View
    # TODO

Fill Info And Save
    [Arguments]    ${data}
    # TODO

Send Email Notification
    Authorize     username=${ACCOUNT}  password=${PASSWORD}
    Send Message  recipients=${RECIPIENT_ADDRESS}
    ...           subject=Failure notification from <RPA Robot name>
    ...           body=<p>There were failures when running the inte-
gration robot!</p>
    ...           save=${TRUE}
    ...           html=${TRUE}
```

Esimerkkikoodi 7. Tämän luvun esimerkkitoteutuksen robottitiedosto *api_robot.robot*. Robotti ajaa tehtäviä, jossa kirjaudutaan verkkopalveluun ja tallennetaan sinne annetut tiedot. Mikäli suoritusvirheitä on tullut, robotti lähettää ilmoitussähköpostin.

Robottitiedossa robotti konfiguroi Robot Framework-DataDriver -kirjaston, joka palauttaa ajettavat tehtävät syötettävien tietojen kanssa. Jokainen tehtävä käynnistää **Task Template**en määritellyn avainsanan, joka avaa uuden selaimen ja navigoi vanhan palvelun verkkosivulle. Verkkosivuille kirjaudutaan ja navigoidaan integraation sivuille. Tiedot syötetään ja tallennetaan integraation sivuille. Selaimen automatisointi tehdään **RPA.Browser.Playwright**-kirjaston avulla. Kun kaikki tehtävät on ajettu, robotti tarkastaa, onko suoritusvirheitä tapahtunut. Tämän määritys tapahtuu **Suite Teardown** -kohdassa. Jos virheitä on tapahtunut, robotti lähettää sähköpostin ilmoittaakseen asiasta **RPA.Email.Exchange**-kirjaston avulla.

Kuten robottitiedostosta huomataan, siinä ei ole *Variables*-osiota, mutta silti käyttää muuttujia. Nämä muuttujat ovat määritelty omaan tiedostoon ja siihen viitataan **Resource**-komennolla *Settings*-osiossa.

```
*** Variables ***
${API_URL}          https://cloudapp.com/api/data
${API_AUTH_URL}     https://cloudapp.com/api/token
${API_USER}         ROBOT_API_USER
${API_PWD}          SHOULD_BE_SOMEWHERE_ELSE

${UI_URL}           https://oldsystemreplaceme.org/
${UI_USER}          ROBOT_UI_USER
${UI_PWD}           SHOULD_BE_SOMEWHERE_ELSE

${EMAIL_ACCOUNT}              ACCOUNT_NAME
${EMAIL_PWD}                  ACCOUNT_PWD
${RECIPIENT_EMAIL_ADDRESS}    RECIPIENT
```

Esimerkkikoodi 8. Tämän luvun Esimerkkitoteutuksen muuttujatiedosto *api_robot_variables.resource*. Tiedosto sisältää muuttujat pilvipalvelun ohjelmistorajapintaan kirjautumiseen, vanhan järjestelmän käyttöliittymään kirjautumiseen ja virhesähköpostin lähettämiseen.

Näillä tiedostoilla saadaan tehtyä suurin osa määrityksistä, mutta vielä uupuu ajastettu käynnistys ja raportin siirtäminen konfiguroitavaan raporttihakemistoon. Nämä tehdään lyhyellä käynnistysskriptillä seuraavalla sivulla (esimerkkikoodi 9):

```
#!/bin/bash
robot -T api_robot.robot
mv ./*.html  /path/to/report/dir/
```

Esimerkkikoodi 9. Robotin käynnistysskripti *run_api_robot.sh*. Skripti käynnistää robotin, asettaa sille aikaleima parametrin, joka luo raporttitiedostojen nimeen aikaleiman. Robotin ajon jälkeen raporttitiedosto siirretään haluttuun hakemistoon.

Käynnistysskripti voidaan ajastaa **Task Scheduler**illä Windows-käyttöjärjestelmässä ja **Cron**illa Unix-ympäristöillä. Tällä saadaan kaikki määritykset toteutettua.

4.3 Tietokannassa olevan datan migraatio sovellusrajapintaan

Organisaatio käyttää useaa sisäistä palvelua, joista muutamat ovat hyvin vanhoja. Organisaatio on päivittämässä palvelujaan ja tähän siirtymävaiheeseen on suunniteltu datamigraatiota helpottavaa robottia.

Ideana on hakea vanhojen palveluiden tietokannoista datat kyselyillä, muokata ne haluttuun rakenteeseen ja lähettää data uusien palvelujen ohjelmointirajapintoihin. Robotin määritellään toimivan näin:

- Robotti käynnistetään manuaalisesti.
- Robotti lukee konfiguraatiotiedostosta tietokantaan ja ohjelmointirajapintaan liittyvät tiedot.
- Robotti suorittaa määritetyn tietokantakyselyn hakeakseen siirrettävän datan.
- Robotti muokkaa tietokantakyselystä saadun datan ohjelmointirajapinnan hyväksymään muotoon.
- Robotti lähettää muokatun datan määriteltyyn rajapintaan.

Käyttäjän siis odotetaan käynnistävän robotin manuaalisesti ja asettavan tarvittavat tiedot robotille. Käyttäjän odotetaan myös käsittelevän mahdolliset virhetilanteet. Esimerkkitoteutus on korkean tason esimerkki eikä välttämättä toimi kaikissa tilanteissa. Esimerkissä ei oteta kantaa tietokantakyselyyn, mihin muotoon data pitää muokata tai miten muokkaus tehdään. Tämä esimerkkitoteutus on seuraavalla sivulla (esimerkkikoodi 10).

```
*** Settings ***
Resource    db_to_api_variables.resource
Library     RPA.Database
Library     RPA.HTTP

*** Variables ***
# Database variables
${DB_TYPE}          pymysql
${DB_NAME}          db_name
${DB_HOST}          127.0.0.1
${DB_PORT}          3306
${DB_USER}          username@hostname
${DB_PWD}           super_secret_pwd

${QUERY}            SELECT * FROM customers

# API variables
${API_URL}          https://corpapp.com/api/data
${API_AUTH_URL}     https://corpapp.com/api/token
${API_USER}         ROBOT_API_USER
${API_PWD}          super_secret_api_pwd

# Api endpoint structure as dictonary mapping
# the field structures (e.g. name -> customer_name)
&{ENDPOINT_FORMAT}  name=customer_name   address=post_address

*** Tasks ***
Fetch Data From DB And Send To API
    Connect To Database   ${DB_TYPE}  ${DB_NAME}  ${DB_USER}
    ...                   ${DB_PWD}   ${DB_HOST}  ${DB_PORT}
    @{RESULTS} =    Query    ${QUERY}
    ${MODIFIED_DATA} =  Modify Query Data To API Format
    ...                 &{ENDPOINT_FORMAT}
    Send Data To API    ${MODIFIED_DATA}

*** Keywords ***
Modify Query Data To API Format
    [Arguments]    ${FORMAT}
    # TODO: do the format change
    &{MODIFIED} =   Create Dictionary
    [Return]   ${MODIFIED}

Send Data To API
    [Arguments]    ${JSON}
    &{AUTH_DATA} =  Create Dictionary  username=${API_USER}
    ...     password=${API_PWD}
    ${AUTH_RESPONSE} =  POST  ${API_AUTH_URL}     data=${AUTH_DATA}

    &{HEADERS} =  Create Dictionary
    ...    authorization=Bearer ${AUTH_RESPONSE.json()}[access_token]
    ${DATA_RESPONSE} =  POST  ${API_URL}  json=${JSON}
    ...    headers=${HEADERS}
```

Esimerkkikoodi 10. Yksinkertainen esimerkkitoteutus 4.3 tilanteeseen. Robotti yhdistää tietokantaan ja suorittaa kyselyn. Kyselyn tulokset muokataan formaattiin, jonka ohjelmointirajapinta hyväksyy. Sen jälkeen robotti lähettää muokatun datan rajapintaan.

5 RPA nyt ja tulevaisuudessa

Tämä luku keskittyy katsomaan ohjelmistorobotiikan nykytilannetta ja mahdollisia tulevaisuuden kehittymistä. Luvussa käydään läpi myös ohjelmistorobotiikan vaikutuksia ohjelmistokehitysprojekteihin ja itse ohjelmistokehityksen prosesseihin.

5.1 RPA:n tilanne tällä hetkellä

Ohjelmistorobotiikka on ollut pitkään olemassa, mutta alkoi saamaan tuulta alleen 2017 aikaan. Tällöin sille ennustettiin 2,9 miljardin Yhdysvaltain dollarin (USD) markkina-arvoa vuodelle 2021. Vuotta aikaisemmin ohjelmistorobotiikan markkina-arvo oli 250–271 miljoonaa dollaria, joten hyvin suurta kasvua ennustettiin. [38; 39.]

Ennusteet eivät aivan osuneet oikeaan, sillä globaali ohjelmistorobotiikkamarkkina kasvoi vielä enemmän. Vuoden 2021 markkina-arvo arvioitiin olevan 7,11 miljardia USD, ja nykyiset ennusteet vuodelle 2022 ovat 10,01 miljardia dollaria. [40.]

Tähän suuren kasvuun on vaikuttanut useita tekijöitä, esimerkkinä chattibottien ja muiden digitaalisten avustajien suosio, mutta maailmanlaajuinen koronaepidemia kasvatti kiinnostusta automaatioon entisestään. Terveydenhoitoalalla ohjelmistorobotiikkaa on käytetty henkilöstön perehdytykseen, ympärivuorokautiseen ajanvaraukseen, asiakashallintaan ja lääkekehityksen nopeuttamiseen. [40.]

Ohjelmistorobotiikan alan kasvuun ovat myös vaikuttaneet kehittyvät mahdollisuudet suorittaa komplekseja tehtäviä ja jopa päästä-päähän-automatisointeja. Kun aluksi automatisoitiin yksinkertaisia staattisesti toistuvia prosesseja ja tehtäviä, niin nykypäivänä sama halutaan tehdä näille vaativille tehtäville. Näiden

suorittaminen onnistuu jo nykypäivänä kehitetyillä teknologioilla kuten tekstintunnistuksella (OCR), koneoppimisella ja analytiikalla ohjelmistorobotiikan alan alla. [41.]

Tämän hetken ohjelmistorobotiikan trendit pyörivät juuri tekoälyn ja ohjelmistorobotiikan yhdistämisen ympärillä. Tätä teknologiaa kutsutaan älykkääksi automaatioksi. Älykkään automaation odotetaan poistavan enemmän kuin 40 % palvelupisteasioinnista vuoteen 2025 mennessä. [41.]

5.2 RPA:n tulevaisuudennäkymät

Kuten aikaisemmasta luvusta huomataan, ohjelmistorobotiikan suosio on kasvanut merkittävää tahtia, ja näin odotetaan tapahtuvan myös tulevaisuudessa. Kasvun odotetaan kasvavan jopa 43,52 miljardiin dollariin vuoteen 2029 mennessä. [40.]

Tätä kasvua ajaa eteenpäin älykkään automaation kasvava tarve, kun hyvin kilpailutetulla markkinalla tarvitaan yhä monimutkaisempien prosessien yksinkertaistamista automaatiolla. Automaation ja tekoälyn kombinaatiolla yritykset pyrkivät lisäämään tuottavuutta ja tehokkuutta sekä vähentämään inhimillisiä virheitä täten, mikä lisää asiakastyytyväisyyttä. [41.]

Kasvavana ohjelmistorobotiikan laajennuksena on hyperautomaatio, joka on ohjelmistorobotiikan, älykäs liiketoimintaprosessien hallintaohjelmiston ja tekoälyn yhdistelmänä. Hyperautomaation on kuvattu olemaan enemmän kuin automaatiota, sillä se tuo huomattavia muutoksia yrityksiin ja toimintoihin. Muutosten alla ovat yrityksen osat IT-infrastruktuurista liiketoimintojen suunnitteluun ja päätöksiin. Hyperautomaation nähdäänkin olevan kokonaisvaltaisempi lähestymistapa automaatioon. [42.]

Hyperautomaation ennustetaan vähentävän yritysten operatiivisia kustannuksia 30 % yhdistämällä hyperautomaatioteknologioita uudelleensuunnitetuille opera-

tiivisille prosesseille vuoteen 2024 mennessä. [43.] Hyperautomaatio vaatii huomattavaa teknillistä osaamista ylläpitää ja kehittää. Tähän alalla on puututtu kehittämällä vähäisen koodin tai koodittomia alustoja, mikä mahdollistaa henkilöstön, joilla ei ole ohjelmointitaustaa, tekemään robotteja. Mutta pelkästään kyky tehdä robotteja ja laajentaa niiden määrää ei aina riitä. Robotit tarvitsevat prosesseja, joita automatisoida. On myös hyvä selvittää, miten parantaa jo olemassa olevia prosesseja ennen kuin niitä automatisoidaan.

Tähän edesauttavat teknologiat, joilla tunnistetaan prosesseja. Näitä ovat prosessilouhinta (eng. process mining) sekä prosessien löytäminen (eng. process discovery). Näiden avulla yritykset löytävät helpommin uusia prosesseja ja mahdollisuuksia parantaa nykyisiä prosesseja ohjelmistorobotiikkaa sekä hyperautomaatiota varten. [44.]

Selvästi on paljon syitä, miksi ohjelmistorobotiikan suosio tulee kasvamaan tulevaisuudessa, mutta on myös huolen aiheita, jotka on otettava huomioon. Näitä kasvua hidastavia asioita ovat vastahakoisuus yrityksissä siirtyä manuaalisesta automaatioon. Tätä vastahakoisuutta voi selittää se, että ohjelmistorobotiikkateknologioita ei yleisesti ymmärretä yrityksissä. Tähän vähäisen koodin alustat auttavat yksinkertaistamalla ohjelmointia tai poistamalla sen kokonaan ohjelmistorobottien kehityksessä.

Myös ongelmaksi ohjelmistorobotiikassa on, että pelkkä automatisointi ei paranna prosesseja. Mikäli prosessi on viallinen, niin sen automatisointi voi vain kasvattaa virheitä ja pahimmassa tapauksessa piilottaa niitä enemmän. Tähänkin on alalla puututtu jo edellä mainituilla prosessilouhinnalla, prosessien löytämisellä ja muilla analytiikalla.

5.3 RPA:n vaikutukset ohjelmistokehittämiseen sekä sen prosesseihin

Ohjelmistorobotiikalla on vähäinen vaikutus itse ohjelmointiin ohjelmistokehityksessä, mutta sillä voidaan yksinkertaistaa ohjelmistokehityksen prosesseja. Sitä

voidaan käyttää esimerkiksi jatkuvan kehityksen putkien hallintaan, kehitystikettien analysointiin ja massamuokkauksiin, testidatan generointiin, asiakaspalautteiden tai bugi-ilmoitusten käsittelyyn tai datan varmuuskopiointiin. [45.]

Ohjelmistorobotiikka voi myös auttaa ohjelmistokehittäjiä keskittymään hankalempiin ja uusiin järjestelmiin, sillä ohjelmistorobotiikkaa voidaan käyttää paikkaamaan vanhempia järjestelmiä toimimaan uudella tavalla. Silloin ohjelmoijien aika voidaan käyttää pelkästään uuden toteutuksen kehittämiseen sen sijaan, että uutta ja vanhaa toteutusta kehitettäisiin ja ylläpidettäisiin samaan aikaan. Robotit ovat myös helposti skaalattavissa tarpeen tullen. Yleisesti samaa ei voi sanoa vanhoista järjestelmistä.

Vähäisen koodin alustat kasvattavat henkilöstön määrää, jotka voivat tehdä robotteja. Näitä henkilöitä voivat olla esimerkiksi projektitason henkilöt, jotka ovat lähimmässä asiakaskontaktissa. Tämä auttaa siihen, että ohjelmoijille jää enemmän aikaa keskittyä haastavampiin muihin tehtäviin.

Ohjelmistorobotit voivat myös säästää rahallisia resursseja ohjelmistokehityksestä, sillä yleisesti robottien tekeminen on nopeampaa ja yksinkertaisempaa. Mikäli robottien tekemiseen käytetään avoimen lähdekoodin ratkaisua, esimerkiksi Robot Frameworkiä, niin taloudelliset kulut pysyvät pieninä (ks. 3.2).

6 Yhteenveto ja päätelmät

Ohjelmistorobotiikan suosio on tasaisessa kasvussa ja näin odotetaan jatkuvan vielä monen vuoden ajan. Tätä kehitystä auttaa tekoälyn ja vähäisen koodin alustojen jatkuva kehitys. Mitä helpommaksi näiden käyttöönotto ja yhdistäminen ohjelmistorobotiikkaan tulee, sitä enemmän ohjelmistorobotiikka laajenee.

RPA:ta tukevien teknologien kehitys myös laajentaa projektien määrää missä ohjelmistorobotiikkaa kannattaa käyttää. RPA-projektien yleistyminen myös vähentää yleisempien ongelmakohtien ilmaantumista, kun kokemusta niiden teke-

misestä kasvaa. Myös automatisoitavien prosessien analysointiteknologiat ehkäisevät tilanteita, joissa automaatio kasvattaa virhetilanteiden määrää toistamalla viallista prosessia. Yhteenvetona vaikuttaa siltä, että mitä pidemmälle mennään teknologian ja kehityksen kanssa, RPA:n rooli kasvaa samalla, kun sen heikkouksia viilataan pois.

Miten sitten Robot Framework automaatioalustana muiden alustojen rinnalla? Robot Framework pärjää hyvin automaatioalustana joustavuutensa ja vähäisten kustannustensa ansiosta. Mutta sen heikkous on, että se vaatii ainakin yhden henkilön, jolla on ohjelmointi- tai skriptaustaustaa. RPA Framework auttaa Robot Frameworkiä tässä suhteessa paljon, mutta ei poista tätä ongelmaa.

Mitä enemmän vähäisen koodin alustat yleistyvät ja ei-tekninen henkilöstö innostuu tekemään robotteja, sitä vähemmän Robot Framework todennäköisesti on sopiva työkalu yrityksille, ainakin toistaiseksi. Toki tämä ei poista sitä, että Robot Framework tulee olemaan äärimmäisen käytetty työkalu konsulteilla ja tekniikanalan yrityksissä. Tämä tarkoittaa sitä, että Robot Frameworkin käyttö RPA-markkinoilla kasvaa, mutta sen kasvu on todennäköistä maltillista.

Omat näkemykseni ohjelmistorobotiikasta yhtyy päätelmien kanssa eli ovat hyvin positiiviset. En ihmettele, miksi sen suosio kasvaa niin hurjaa vauhtia. Tekoälyn kehittyminen ja sen lisääminen automaatioon vapauttaa enemmän ihmisiä tekemään luovia ja haastavampia tehtäviä, mikä on mielestäni suuri mahdollisuus nähdä, miten työtehtävien kehitys muuttuu alalla kuin alalla.

Robot Framework on jäänyt itselleni hyvin lähelle sydäntä. Mielestäni se on mahtava automaatioalustana: tekee sitten ohjelmistorobotiikkaa tai automaatiotestausta. Mahdollisuudet tehdä helposti omia kirjastoja Pythonilla tai laajentaa valmiita kirjastoja omien tarpeiden mukaan on mahtava ominaisuus. Myös Robot Frameworkin sisäänrakennettu raportointi helpottaa sekä ohjelmistorobottien että testiautomaatien tekemistä merkittävästi.

Loppujen lopuksi opinnäytetyön tekeminen tähän aiheeseen on ollut mukava, vaikkakin hyvin pitkä kokemus. Tunnen, että tästä kokemuksesta jäi jotain konkreettistakin käteen. Jään innolla odottamaan mitä tulevaisuus tuo tullessaan ohjelmistorobotiikan ja Robot Frameworkin osalta.

Lähteet

1 Definition and Benefits. Verkkoaineisto. <https://irpaai.com/definition-and-benefits/>. Luettu 30.10.2021.

2 Leibowitz, Stuart; Kakhandiki, Abhijit. 2018. What's the difference between "attended" and "unattended" RPA bots? Verkkoaineisto. <https://www.ibm.com/blogs/cloud-computing/2018/11/19/attended-unattended-rpa-bots/>. 19.11.2018. Luettu 30.10.2021.

3 Break down the differences between attended and unattended automation, and which best addresses your business needs. Verkkoaineisto. <https://www.automationanywhere.com/rpa/attended-vs-unattended-rpa>. Luettu 30.10.2021.

4 Hankiewicz, Kamila. 2018. The Benefits And The Challenges Of RPA Implementation. Verkkoaineisto. <https://kamila.medium.com/the-benefits-and-the-challenges-of-rpa-implementation-56044316dfec>. 27.3.2018. Luettu 30.10.2021.

5 Barlow, David. 2020. How Robotic Process Automation is solving the problems of Small to Medium sized Business: Looking beyond the hype of RPA. Verkkoaineisto. <https://www.sherpaworks.com.au/how-robotic-process-automation-is-solving-the-problems-of-small-to-medium-sized-business-looking-beyond-the-hype-of-rpa/>. 30.6.2020. Luettu 1.11.2021.

6 Ariwala, Pinakin. 2021. 12 Popular Benefits of RPA in Business. Verkkoaineisto. <https://marutitech.com/benefits-of-rpa-in-business/>. Päivitetty 14.10.2022. Luettu 1.11.2021.

7 Rushworth, Darren. 2017. Don't fear the robots: RPA benefits everyone. Verkkoaineisto. <https://disruptive.asia/rpa-automation-benefits-everyone/>. 18.8.2017. Luettu 1.11.2021.

8 7 Challenges to Implementing Robotic Process Automation (RPA) & How to Overcome Them. 2020. Verkkoaineisto. <https://www.cigen.com.au/cigenblog/7-challenges-implementing-rpa-how-overcome-them>. 20.1.2020. Luettu 2.11.2021.

9 Bloomberg, Jason. 2018. Why You Should Think Twice About Robotic Process Automation. Verkkoaineisto. <https://www.forbes.com/sites/jasonbloomberg/2018/11/06/why-you-should-think-twice-about-robotic-process-automation/?sh=6d15c47c5fe1>. 6.11.2018. Luettu 2.11.2021.

10 DeBrusk, Chris. 2017. Five Robotic Process Automation Risks to Avoid. Verkkoaineisto. <https://sloanreview.mit.edu/article/five-robotic-process-automation-risks-to-avoid/>. 24.10.2017. Luettu 8.11.2021.

11 Laskowski, Nicole. 2018. In a hot RPA market, the 'rule of five' keeps CIOs focused on use cases. Verkkoaineisto. <https://searchcio.techtarget.com/blog/TotalCIO/In-a-hot-RPA-market-the-rule-of-five-keeps-CIOs-focused-on-use-cases>. 31.7.2018. Luettu 8.11.2021.

12 McHugh, Brian. 2021. How to Avoid RPA's Disadvantages, According to Gartner. Verkkoaineisto. <https://www.advsyscon.com/blog/rpa-disadvantages-risks/>. Päivitetty 8.11.2022. Luettu 23.11.2021.

13 Dilmegani, Cem. 2021. 21 RPA Pitfalls & Audit Checklist to Tackle Them in 2022. Verkkoaineisto. <https://research.aimultiple.com/rpa-pitfalls/>. 26.10.2017. Luettu 23.11.2021.

14 Tomek, Michal. 2019. Robotic Process Automation: Monitoring Bots As a Necessary Step to Success. Verkkoaineisto. <https://www.minit.io/blog/robotic-process-automation-monitoring-bots-as-a-necessary-step-to-success>. 1.10.2019. Luettu 23.11.2021

15 INTRODUCTION. Verkkoaineisto. <https://robotframework.org/#introduction>. Luettu 23.11.2021.

16 Laukkanen, Pekka. 2006. Data-Driven and Keyword-Driven Test Automation Frameworks. Diplomityö/Verkkoaineisto. <http://eliga.fi/Thesis-Pekka-Laukkanen.pdf>. 24.2.2006.

17 tool images are now copied correctly. 2008. Verkkoaineisto. <https://github.com/robotframework/robotframework/releases/tag/2.0>. 24.6.2008.

18 RBTFRMWRK/FOUNDATION. Verkkoaineisto. <https://robotframework.org/foundation/>. Luettu 23.11.2021.

19 Creating test libraries. Verkkoaineisto. <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#creating-test-libraries>. Luettu 23.11.2021.

20 Standard libraries. Verkkoaineisto. <https://robotframework.org/robotframework/#standard-libraries>. Luettu 23.11.2021.

21 RESOURCES. Verkkoaineisto. <https://robotframework.org/?tab=builtin#resources>. Luettu 23.11.2021.

22 DataDriver for Robot Framework®. Verkkoaineisto. < https://github.com/Snooz82/robotframework-datadriver#datadriver-for-robot-framework>. Luettu 28.2.2022.

23 Introduction. Verkkoaineisto. <https://github.com/robotframework/SeleniumLibrary/#introduction>. Luettu 28.2.2022.

24 robotframework-browser. Verkkoaineisto. <https://github.com/MarketSquare/robotframework-browser#robotframework-browser>. Luettu 28.2.2022.

25 readme. Verkkoaineisto. <https://github.com/MarketSquare/robotframework-requests#readme>. Luettu 28.2.2022.

26 Introduction. Verkkoaineisto. <https://github.com/serhatbolsu/robotframework-appiumlibrary#introduction>. Luettu 28.2.2022.

27 Robotframework-Database-Library. Verkkoaineisto. <https://github.com/franz-see/Robotframework-Database-Library#robotframework-database-library>. Luettu 28.2.2022.

28 Introduction. Verkkoaineisto. <https://github.com/robotframework/SSHLibrary#introduction>. Luettu 28.2.2022.

29 What is RobotFramework-Zoomba? Verkkoaineisto. <https://github.com/Accruent/robotframework-zoomba#what-is-robotframework-zoomba>. Luettu 28.2.2022.

30 Introduction. Verkkoaineisto. <https://github.com/GDATASoftwareAG/robotframework-flaui#introduction>. Luettu 25.11.2022.

31 RESOURCES. Verkkoaineisto. <https://robotframework.org/?tab=libraries#resources>. Luettu 28.2.2022.

32 Robot Framework FTP Library. Verkkoaineisto. <https://github.com/kowalpy/Robot-Framework-FTP-Library#robot-framework-ftp-library>. Luettu 28.2.2022.

33 Introduction. Verkkoaineisto. <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#introduction-2>. Luettu 4.11.2022.

34 Different test library APIs. Verkkoaineisto. <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#different-test-library-apis>. Luettu 4.11.2022.

35 Dynamic library API. Verkkoaineisto. <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#dynamic-library-api>. Luettu 4.11.2022.

36 Hybrid library API. Verkkoaineisto. <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#hybrid-library-api>. Luettu 4.11.2022.

37 Libraries. Verkkoaineisto. <https://rpaframework.org/#libraries>. Luettu 7.11.2022.

38 Tomek, Michal. 2019. The Current State of Robotic Process Automation. Verkkoaineisto. <https://www.minit.io/blog/the-current-state-of-robotic-process-automation>. 24.9.2019. Luettu 8.11.2022.

39 Fersht, Phil; Snowdon, Jamie. 2017. The Robotic Process Automation market will reach $443 million this year. Verkkoaineisto. <https://www.horsesforsources.com/rpa-marketsize-hfs_061017/>. 10.6.2017. Luettu 8.11.2022.

40 Robotic Process Automation Market Size [2022-2029] Worth USD 43.52 Billion | Exhibiting a CAGR of 23.4%. 2022. Verkkoaineisto. <https://www.globenewswire.com/en/news-release/2022/09/27/2523189/0/en/Robotic-Process-Automation-Market-Size-2022-2029-Worth-USD-43-52-Billion-Exhibiting-a-CAGR-of-23-4.html>. 27.9.2022. Luettu 8.11.2022.

41 Robotic Process Automation Market Size, Share & Trends Analysis Report By Type, By Service, By Application, By Deployment, By Organization, By Region, And Segment Forecasts, 2022 – 2030. Verkkoaineisto/Raportti yleiskatsaus. <https://www.grandviewresearch.com/industry-analysis/robotic-process-automation-rpa-market>. Luettu 21.11.2022.

42 Dilmegani, Cem. 2022. Guide to Hyperautomation in 2022: Technologies, Pros & Cons. Verkkoaineisto. <https://research.aimultiple.com/hyperautomation/>. 10.10.2022. Luettu 21.11.2022.

43 Gartner Forecasts Worldwide Hyperautomation-Enabling Software Market to Reach Nearly $600 Billion by 2022. 2021. Verkkoaineisto. <https://www.gartner.com/en/newsroom/press-releases/2021-04-28-gartner-forecasts-worldwide-hyperautomation-enabling-software-market-to-reach-nearly-600-billion-by-2022>. 28.4.2021. Luettu 21.11.2022.

44 Differences between Process Mining vs. Process Discovery vs. Task Mining. Verkkoaineisto. <https://www.workfellow.ai/differences-between-process-mining-vs-process-discovery-vs-task-mining>. Luettu 22.11.2022.

45 6 Ways To Leverage RPA in IT Operations. 2021. Verkkoaineisto. <https://www.botreetechnologies.com/blog/6-ways-to-leverage-rpa-in-it-operations>. 24.11.2021. Luettu 25.11.2022.