HAMK
Hämeen ammatti-
korkeakoulu

Ohjelmistotestauksen automatisointi

Case: Laskutusprosessi

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus

Syksy 2022

Christa Bergström-Nieminen

HAMK
Hämeen ammatti-
korkeakoulu

Tiivistelmä

Koulutus	Tietojenkäsittelyn koulutus	
Tekijä	Christa Bergström-Nieminen	Vuosi 2025
Työn nimi	Ohjelmistotestauksen automatisointi	
Ohjaaja	Lasse Seppänen	

Opinnäytetyön tarkoituksena oli selvittää, minkälainen testausstrategia ja -automaatiot soveltuvat aloittavalle startup-yritykselle. Tutkimuksen avulla pyritään varmistamaan alan parhaiden käytänteiden omaksuminen testausstrategiaa ja automatisointeja luodessa. Opinnäytetyön toimeksiantajana oli RoboTech, joka on perusteilla oleva yritys.

Opinnäytetyön tietopohja koostuu strategisesta tasosta, jossa esitellään Agile-kehitysmenetelmä sekä syvennytään testausstrategiaan liittyviin teorioihin. Tämän jälkeen käsitellään järjestelmätestausta operatiivisella tasolla syventymällä testauksen tyyppeihin, rakenteeseen ja vaiheisiin, jonka jälkeen siirrytään testauksen automatisoinnin teoriaan. Opinnäytetyö on toiminnallinen. Tutkimus pohjautui teorian, käytännön ja valittujen ohjelmistojen analysointiin, jonka perusteella pystyttiin luomaan kaksi automatisoitavaa testitapausta. Analysoinnissa käytettiin teorioiden suosittelemia menetelmiä, kuten Pareto.

Tutkimuksissa havaittiin, että hypoteesista poiketen, startup-yrityksille ei suositella testausstrategian luontia, sillä sen ylläpito sitoo turhaan resursseja. Pienissä yrityksissä testaustavoista sopiminen onnistuu usein ilman kattavaa strategiaa. Toinen havainto oli, että testauksia automatisoidessa määrällisesti eniten tulisi olla lyhyitä yksikkötestauksia, toiseksi eniten integraatiotestauksia ja vähiten käyttöliittymätestauksia. RoboTechille kuitenkin luotiin kaksi käyttöliittymätestausta UiPath-ohjelmistolla, sillä tämä oli ennakkoon määritetty automatisoinnissa käytettäväksi ohjelmistoksi. Kolmanneksi havaittiin, että automatisoitavien kohteiden valinnassa tulisi hyödyntää Pareto-periaatetta, jolla pyritään löytämään 80 % ongelmista automatisoimalla 20 % tapauksista.

Avainsanat	Agile-kehittäminen, testauksen automatisointi, UiPath ja testausstrategia
Sivut	37 sivua ja liitteitä 6 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Abstract

DP	Degree Programme in Business Information Technology
Author	Christa Bergström-Nieminen
Subject	Software testing automation
Supervisors	Lasse Seppänen

Year 2025

The purpose of the thesis was to find out what kind of testing strategy and automations are suitable for a startup company. The thesis aimed to confirm the implementation of the best practices of the field in creation of test strategy and test automations. Work was conducted to RoboTech, which is currently to be founded.

The knowledge base of the thesis is based on the strategic level in which Agile development methodology is presented and deep dive into the theories related to testing strategy. After that, the thesis presents theories related to software testing on an operative level to divided into different types of testing, testing structure and phases, following testing automation theories. The thesis is a functional study. Study was based to analysing theory, practical and selected software. Based on the analysis two test scenarios were automated. Analyses were conducted by methods recommended in theories, like the Pareto-method.

During the study, it was noticed that contrary to hypothesis, testing strategy is not recommended for startup companies, because it ties unnecessarily resources. In small companies the best ways and practices to conduct tests are easily agreed and communicated without full testing strategy. Second finding was, that when automating test scenarios, majority of cases should be short unit tests followed by integration tests, and the least amount should be user interface tests. However, the automated test scenarios created for Robotech were user interface tests due to the preliminarily selected software UiPath, with which the user interface tests were more easily created. Third observation of the study was to utilize Pareto-method in selecting automated test cases. In the theory, target is to find 80% of the bugs by automating 20% of the cases.

Keywords	Agile development, testing automation, UiPath and test strategy
Pages	37 pages and 6 pages

Sisällys

Sanasto

VBA	Visual Basic for Applications, Microsoftin luoma ohjelmointikieli.
RPA	Robotic Process Automation, ohjelmistorobotti, joka suorittaa sääntöihin perustuvia työtehtäviä.

Kuvat

Ohjelmakoodit

Liitteet – esimerkki

1 Johdanto

Järjestelmät ja niiden kehitys on oleellinen osa liiketoimintaa, sillä sen avulla pyritään tarjoamaan asiakkaille parempia palveluita kustannustehokkaasti sekä parantamaan sisäisten prosessien läpimenoaikaa, jotta yritys pystyy keskittymään tuottavaan työhön. Riippumatta järjestelmäkehityksen viitekehyksestä, kaikissa yksi oleellinen osa on kehitystöiden testaus. Järjestelmätestauksen tavoitteena on varmistaa, ettei kehitys ole rikkonut olemassa olevia toiminteita ja vahvistaa, että se vastaa kehitykselle määritettyjä vaatimuksia.

Opinnäytetyössä määritellään startup-yritys RoboTechin liiketoiminnan kannalta soveltuva testausstrategia ja automatisoidaan kaksi kriittisintä testitapausta. Tavoitteena on vahvistaa järjestelmän toimivuus kriittisten toimintojen osalta ja pystyä käyttämään esimerkkeinä automatisointeja myyntityössä esimerkkeinä. Automatisointi tapahtuu UiPath prosessiautomaatio järjestelmällä, joka on yrityksen liiketoiminnan kannalta oleellinen työkalu ja henkilöstöllä löytyy kokemusta automatisoinneista UiPathin avulla.

Opinnäytetyöhön ei oteta mukaan tilaus-laskutus-prosessien ulkopuolella olevia liiketoimintaprosesseja ja liiketoiminnan kriittisten toiminnallisuuksien osalta luodaan automaatiot vain soveltuvimmalle kahdelle prosessille. Opinnäytetyö keskittyy loppukäyttäjätestaukseen johtuen järjestelmän kevyestä koodauksesta sekä vähäisistä relaatioista muihin järjestelmiin tai tietokantoihin. Opinnäytetyössä ei myöskään käsitellä rasitustestausta, sillä yrityksen järjestelmään ei liity suurta kuormitusta. Työn tulee vastata kysymyksiin:

- Mitä suositellaan testausstrategiaksi startup-yritykselle?
- Millaista testausautomaatiota suositellaan yritykselle?
- Mitkä ovat liiketoimintakriittiset automatisoitavat testitapaukset yrityksen tilaus- ja laskutusjärjestelmästä?

2 Agile-kehitys ja testausstrategia

Luvussa käsitellään yritysten strategista tasoa ja linjauksia liittyen yrityksen sisäiseen järjestelmäkehitykseen. Luku alkaa agile-viitekehityksen käsittelyllä, joka määrittelee järjestelmäkehityksen tavan. Toisessa alaluvussa käsitellään testausstrategiaa, sillä testaaminen on osa agile-kehitystä ja testausstrategiaan liittyvän teorian avulla pystytään vastaamaan kysymykseen minkälaista testausstrategiaa, suositellaan startup-yritykselle.

2.1 Agile-kehitys

Agile-kehityksen pääperiaatteena on ratkaista ongelmat tiiminä ja projektin parissa työskentelevien yhteisenä tavoitteena on parantaa tuotetta ja tuottaa mahdollisimman paljon arvoa asiakkaalle. Kasvanut kilpailu on synnyttänyt tarpeen ketterämmälle kehitykselle ja nopeammille kehitysprojekteille, sillä yritysten on tarjottava erilaisia palveluita asiakkaille yhä nopeammin. Agile-testauksen näkökulmasta tavoitteena on auttaa tuottamaan laadullisesti mahdollisimman hyvä tuote. (Crispin & Gregory, 2008; Myers ym., 2011)

Kun puhutaan agile-kehityksen roolituksista, keskiössä on testaajien taidot ja mitä taitoja tarvitaan, jotta testaustuote pystytään toimittamaan. Tämän vuoksi taidot jaetaan kahteen tiimiin, joita ovat asiakas- ja kehitystiimi. (Crispin & Gregory, 2008)

Asiakastiimiin kuuluvat henkilöt, joilla on liiketoimintatuntemusta ja he pystyvät määrittelemään liiketoimintanäkökulman testaukseen. Asiakastiimin tehtävänä on varmistaa, että testitapauksissa otetaan huomioon liiketoiminnan tarpeet, jotta testattava tuote täyttää sille asetetut vaatimukset. Asiakastiimi on jokaisessa vaiheessa mukana kehityksessä varmistamassa, että tekninen tiimi ymmärtää kehitykselle asetetut vaatimukset. (Crispin & Gregory, 2008)

Kehitystiimiin kuuluvat muun muassa kehittäjät, järjestelmien pääkäyttäjät, tietoturva-asiantuntijat sekä arkkitehdit eli kaikki, jotka ovat tuottamassa koodia tai varmistamassa teknistä laatua. Tavoitteena on, että kehitystiimi voi vapaasti valita mitä kehitystä he tuottavat ja taitoja pyritään saavuttamaan usealta osa-alueelta. Eri kehityksissä tarvitaan kuitenkin eri taitoja ja tiimin tulee huomioida se, kenellä on parhaat kyvykkyydet kehityksen

suorittamiseen. Testausten suorittajat ovat osa kehitystiimiä, sillä he ovat keskeinen osa agile-kehitystä ja heidän roolinsa on toimia kehittäjien tukena ja auttaa heitä varmistamaan kehityksen laadukkuus. (Crispin & Gregory, 2008)

Agile-kehityksessä optimaalisessa tilanteessa asiakas- ja kehitystiimit toimivat yhtenä yksikkönä kohti yhteisiä tavoitteita. Agile-kehitys etenee tavallisimmin yhdestä neljään viikkoon kestävissä kehityssykleissä eli sprinteissä. Asiakastiimi priorisoi kehityspyynnöt kehitystiimille, jotka määrittelevät kuinka monta kehitystä pystytään ottamaan työnalle. Tämän jälkeen tiimit työskentelevät yhdessä, jotta koodi saadaan luotua, testattua ja parannettua, kunnes kehitys voidaan viedä suunnitellun aikataulun mukaisesti tuotantoon. (Crispin & Gregory, 2008)

Vaikka asiakas- tai kehitystiimissä ei ole välttämättä yhtään henkilöä, joka mieltää itsensä testaajaksi, testaus on oleellinen osa agile-kehitystä. Jotta kehittäjälle pystytään kertomaan, mitä hänen tulee muuttaa koodissa, tarvitaan testausta. Kaikki henkilöt kehityksen ympärillä ovat vastuussa testauksesta ja laadun varmistuksessa. Yksi agile-testauksen metodeista on oppia tuotteesta testausten aikana ja antaa testausten tulosten johdattaa testauksia, joten henkilö ei välttämättä miellä itseänsä perinteiseksi testaajaksi, jolla on ennakkoon määritelty tarkka kuvaus testattavista skenaarioista. Tämä on linjassa agile-kehityksen kanssa, sillä tavoitteena on vastata kehitykseen ja pystyä tuottamaan arvoa nopeasti. (Crispin & Gregory, 2008)

Kun verrataan Agile-testausta ja perinteistä testausta, selkeä ero on siinä missä vaiheessa testaus tapahtuu. Perinteisessä testauksessa testaus tapahtuu kehityksen lopussa, juuri ennen tuotantoon vientiä. Tämä aiheuttaa usein sen, että aikataulut venyvät, sillä testaukselle ei varata tarpeeksi aikaa tai tarvittavia korjauksia ei ole aikataulutettu. (Crispin & Gregory, 2008)

Agile-testauksessa testaus tapahtuu aina kun koodi on saatu valmiiksi, jolloin kehittäjä saa palautteen ja pystyy tekemään tarvittavia muutoksia. Muutosten jälkeen koodi testataan taas. Jos koodi läpäisee testauksen, kehittäjä siirtyy seuraavaan vaiheeseen. (Crispin & Gregory, 2008)

Toinen selkeä ero perinteisen ja agile-kehityksen välillä on roolituksiin liittyvät erot. Perinteisessä kehityksessä testaajat ovat viimeinen osa kehitystä ja he antavat palautteen

aikaisemmille osille, kun taas agile-kehityksessä koko kehitystiimi työskentelee yhdessä kaikissa vaiheissa. (Crispin & Gregory, 2008)

2.2 Testausstrategia

Testausstrategialla määritellään yrityksen testausperusteet ja -mallit kokonaisuudessaan, jotta yrityksen työntekijöillä on yhtenäinen kehys, johon testaukset perustuvat. Tällä pyritään varmistamaan, ettei yrityksen käyttämissä tai myymissä sovelluksissa ole virheitä sekä liiketoiminnan laadukas toiminta. Yrityksen kasvaessa strategian vaikuttavuus kasvaa, sillä sen avulla pystytään viestittämään ja varmistamaan yhtenäiset tavat suorittaa testauksia yrityksen sisällä ja yrityksen toimialan mukaan mahdollisesti myös asiakkaiden kanssa. (Yushkevich, 2024)

Testausstrategian tulee pitää sisällään useita näkökulmia testaukseen liittyen, kuten valittu viitekehys, riskit, roolit, työkalut, automaatiot ja dokumentaation kattavuus. Strategian laatiminen tulee tehdä huolellisesti ja strategian luojalla tulisi olla syvä ymmärrys organisaation luonteesta, järjestelmistä sekä kehitysprosesseista, jotta pystytään varmistamaan strategian soveltuvuus ja jalkautus yrityksen operatiiviselle tasolle. (Yushkevich, 2024)

Testausstrategiaa luodessa sitä ei saa sekoittaa testaussuunnitelmaan, joka määrittelee testauksen skenaariot kehitykseen tai projektiin liittyen. Testaussuunnitelma tulee tehdä jokaiselle projektille tai kehitykselle erikseen ja suunnitelmaa voidaan täydentää ja päivittää kehitysprojektin edetessä. Strategiaa muutetaan harvoin ja sen sisältö tulee huomioida testaussuunnitelmaa luodessa. (AltexSoft, 2024; Yushkevich, 2024)

2.3 Testausstrategian hyödyt ja sisältö

Ensisijaisena tarkoituksena testausstrategialla on varmistaa, että yrityksen työntekijät ymmärtävät miten järjestelmien testaus vaikuttaa yrityksen sisäisten ja ulkoisten järjestelmien laatuun. Strategian avulla henkilöstö tietää mitä toimia tulee toteuttaa ja milloin. Tämä auttaa myös asiakkaiden odotusten hallinnassa, sillä testaus vaikuttaa suoraan kehitystöiden aikatauluihin. Testausstrategian avulla varmistetaan myös kehitystyön laatu ja se vähentää yllätyksiä kustannuksissa. (Tiukova, 2024)

Testausstrategian hyödyt ja sisältö riippuvat strategian yleisöstä, joita voivat olla esimerkiksi sidosryhmät, testaajat tai koko yritys. Jos dokumentin tarkoituksena on pystyä vakuuttamaan yrityksen johto siitä, että testaukselle tulee varata budjettia, on keskityttävä testauksesta saataviin hyötyihin ja perusteltava esimerkiksi yrityksen valitsemien ohjelmistojen kustannukset, mutta jättää tarkat tekniset yksityiskohdat dokumentaatiosta. (AltexSoft, 2024)

Toinen tarkoitus testausdokumentaatiolle on testaajien ohjaaminen, jolloin dokumentaation tulisi korostaa käytänteitä ja miten strategiaa voidaan hyödyntää työtehtävissä. Esimerkiksi valitut suunnittelutekniikat, testaukseen liittyvät järjestelmät ja dokumentaatiovaatimukset ovat kyseiselle ryhmälle olennaisempia kuin tarkat perustelut, miksi näihin on päädytty tai niiden tuomat hyödyt. (AltexSoft, 2024)

Kolmantena yleisönä voi olla koko yritys, jolloin sisällön tulisi olla selkeä ja testausstrategia tulisi olla kuvattuna hyvin yleisellä tasolla. Erilaisten lukijaryhmien takia testausstrategiasta kannattaa luoda useampi kohderyhmän mukaan kirjoitettu strategia, joka pohjautuu yleiseen ylätason strategiaan. Tällä varmistetaan, että strategiasta on luotu tarpeeksi kattava dokumentaatio, joka vastaa kaikkien lukijoiden tarpeisiin. (AltexSoft, 2024)

Kaikille yrityksille ei suositella testausstrategian luontia, sillä sen luonti ja ylläpito vaatii henkilöstöä ja resursseja. Esimerkiksi pienet startup-yritykset saattavat pärjätä ilman testausstrategiaa, sillä yritykset tyypillisesti ratkaisevat ongelmat lennosta. Henkilöstöä on myös vähän, joten tarvittaville henkilöille on helppo kertoa sanallisesti testausten kattavuudesta ja niiden suoritustyylistä. Tieto voidaan esimerkiksi lisätä muutamalla lauseella testaussuunnitelmaan. (AltexSoft, 2024)

2.4 Testausstrategioiden tyypit

Testausstrategioille on määritelty yhdeksän eri tyyppiä, ja yritys pystyy hyödyntämään yhtä tai useampaa strategiaa. Usein eri tilanteissa eri strategia saattaa soveltua paremmin. Esimerkiksi tietoturvaan liittyviä testauksia tulee tehdä eri strategian perusteella kuin agile-viitekehyksen mukaista järjestelmätestausta. Jokainen strategiatyyppi pyrkii varmistamaan kehityksen laadun erilaisten vaatimusten ja standardien täyttämiseksi. (Yushkevich, 2024)

Ensimmäinen testausstrategian tyyppi on metodinen strategia. Strategiaa noudatettaessa ennen testejä tulee määritellä säännöt ja standardit, jotka tulee olla täytettynä ja tätä käytetään usein tietoturvatestauksessa. Metodinen strategia soveltuu myös hyvin pankki- ja vakuutusaloille, joissa toimintaa sääntelee useat lait ja säännöt. (Yushkevich, 2024)

Toinen tyyppi on ketterä strategia, joka edistää yhteistyötä ja sopeutumiskykyä. Tässä korostetaan yhteistyön merkitystä ja kehitystiimin jäsenet, joita ovat muun muassa kehittäjä, asiakas ja testaaja työskentelevät lähellä toisiaan koko kehityskaaren ajan. Tällä pyritään varmistamaan nopea reagointi ongelmien ilmentyessä, jotta saadaan kehitysaika pidettyä mahdollisimman lyhyenä. (Yushkevich, 2024)

Kolmantena on analyyttinen strategia, jossa kehittäjät käyvät läpi projektin tekniset vaatimukset etukäteen ja varmistavat että heidän huomionsa lisätään kehityskohtaiseen testausstrategiaan. Tämä testausstrategian tyyppi soveltuu hyvin projekteihin, jotka sisältävät todennäköisempiä riskejä, jotta nämä tulevat varmasti testatuksi kattavalla tasolla. (Yushkevich, 2024)

Neljäs tyyppi on reaktiivinen strategia, jossa testaus aloitetaan ohjelmiston tai kehityksen julkaisun jälkeen. Reaktiivinen strategia perustuu joustavuuteen ja vikoihin reagoidaan niiden ilmaantuessa. (Yushkevich, 2024)

Viidentenä on mallipohjainen strategia, jossa pyritään simuloimaan todellisia olosuhteita. Testaajat ovat vastuussa tämän mallin luomisesta ja olosuhteiden huomioimisesta. Usein tämä strategia sisältää myös mallinnuskielien käyttöä. Strategia soveltuu yrityksille tai kehityksiin, joissa arkkitehtuuri on monimutkainen ja vuorovaikutussuhteiden ymmärtäminen voi olla haastavaa. (Yushkevich, 2024)

Kuudentena on ohjattu/neuvoa-antava strategia, jossa luotetaan esimerkiksi asiakkaiden asiantuntijuuteen priorisoimalla heidän ajatuksiaan ja ehdotuksiaan testauksiin liittyen. Tällä pyritään varmistamaan, että kehitys vastaa asiakkaan toiveita. Strategia soveltuu muun muassa käyttöliittymätestaukseen ja esimerkiksi yritykset, jotka luovat sivustoja asiakkaille voisivat hyötyä tästä strategiasta. (Yushkevich, 2024)

Seitsemäntenä on regression vastainen strategia, jonka pääajatuksena on voimakas automatisointi ja regressioriskien minimointi. Strategia soveltuu yrityksille, joiden tuotteisiin

tai ohjelmistoihin tulee usein päivityksiä. Regression vastaisessa strategiassa pyritään varmistamaan, ettei tuote taannu uudistusten vuoksi. (Yushkevich, 2024)

Kahdeksantena on prosessin/standardin mukainen strategia, jonka tarkoituksena on testauksessa seurata valittua standardia ja varmistaa sen pätevyys kehityksessä. Esimerkkinä käytetystä standardista on esimerkiksi ISO:n kaltaiset kansainväliset standardit. Tämä testausstrategia soveltuu erityisesti yrityksille, jotka toimivat tarkkojen sääntelyiden alla. (Yushkevich, 2024)

Viimeisenä eli yhdeksäntenä testausstrategiana on sekoitettu strategia. Tämä strategia pyrkii yhdistelemään parhaita käytänteitä, joilla optimoidaan testauskäytänteitä, jotta varmistetaan parhaat tulokset. Usein yrityksillä on erilaisia tarpeita ja tilanteita, joten sekoitettu strategia soveltuu useimmille yrityksille. (Yushkevich, 2024)

3 Järjestelmätestaus

Järjestelmätestauksen osalta syvennytään testauksen operatiivisen puolen teorioihin, kuten agile-testauksen parhaisiin käytänteisiin, testauksen rakenteeseen ja periaatteisiin sekä testauksen vaiheisiin. Tämän jälkeen käydään läpi testitapausten automatisointiin liittyviä teorioita. Luku tuo teorianäkökulman tutkimuskysymyksiin ”Millaista testausautomaatiota suositellaan yritykselle?” ja ”Mitkä ovat liiketoimintakriittiset automatisoitavat testitapaukset yrityksen tilaus- ja laskutusjärjestelmästä?”.

3.1 Regressiotestaus agile-kehitysympäristössä

Regressiotestauksesta puhuttaessa tarkoitetaan ohjelmistotestauskäytäntöä, jonka tavoitteena on varmistaa olemassa olevien toiminteiden virheetön toiminta uusien toiminteiden kehityksen yhteydessä. Tämä on oleellista, sillä järjestelmien kehitystyössä usein toiminteiden välillä on usein relaatioita ja muutos yhdessä toiminteessa saattaa sitä kautta vaikuttaa myös toiseen toiminteeseen. Regressiotestauksessa ajetaan aikaisempien kehitysten testiskenaarioita läpi ja varmistetaan, ettei tuloksiin ole tullut muutoksia. (Solutions, 2025)

Testaus kuuluu tärkeänä osana järjestelmäkehitykseen. Toisin kuin perinteisessä vesiputousmallissa, jossa testaus tehdään projektin lopussa, agile-kehityksessä testaaminen suoritetaan iteratiivisesti useissa sykleissä. Syklien aikana regressiotestauksen tarve korostuu, sillä seuraavan syklin kehitykset saattavat vaikuttaa joko suoraan tai epäsuorasti edellisen syklin kehityksiin. Tällöin edellisessä syklissä suoritettuja testauksia olisi kannattavaa testata uudelleen. (Crispin & Gregory, 2008; Solutions, 2025)

Agile-projekteissa suurempi kokonaisuus jaotellaan pienempiin hallittaviin osiin ja aina osan tullessa valmiiksi se testataan ja viedään tuotantoon seuraavassa julkaisussa. Tällä pyritään varmistamaan projektien nopeampi läpimenoaika. Nopeuden vuoksi ohjelmistojen toimivuus tulisi testata säännöllisesti, jotta pystytään välttämään regressio-ongelmia. Agile-projektien aikana testitapauksia tulisi luoda siten, että regressiotestauksia pystyisi tarpeen mukaan automatisoimaan, jolloin toiminteiden laatu pystytään todentamaan myös projektin päättymisen jälkeen. (Crispin & Gregory, 2008; Solutions, 2025)

3.2 Testauksen rakenne ja periaatteet

Kaikissa testauksissa pätee viisi dimensiota, jotka tulee pitää mielessä, vaikka testausten määrittelyssä otettaisiin huomioon vain muutama tekijä ja muut jätetään avoimeksi. Päätös jättää tekijöitä avoimeksi voi antaa testaajalle vapauden toimia haluamallaan tavalla, mikä saattaa olla tavoitteen mukaista esimerkiksi käyttöliittymätestauksen yhteydessä, kun halutaan antaa tilaa testaajien luovuudelle. (Kaner ym., 2011)

Ensimmäisenä dimensiona ovat **testaajat**, jotka tyypillisesti ovat henkilöitä organisaation sisältä, jotka käyttävät tuotetta (käyttöliittymätestaus) tai potentiaaliset asiakkaat (beta-testaus). Toisena dimensiona on **testausten kattavuus**, jossa määritellään, mikä on tarvittava laajuus testaukselle, jotta saadaan tarvittavat hyödyt, esimerkiksi katetaanko yksikkötestauksessa kaikki tehdyt yksiköt. Kolmantena dimensiona ovat **mahdolliset ongelmat**. Tällä pyritään vastaamaan kysymyksiin ”Miksi testausta suoritetaan?”, ”Mitä riskejä testauksella pyritään vähentämään?” Neljäntenä dimensiona ovat **aktiviteetit**, jonka avulla pyritään määrittelemään, miten testataan esimerkiksi tutkivalla testausmenetelmällä. Viidentenä dimensiona on **arviointi**, joka määrittää onnistuiko testitapaus vai ei ja mistä tiedetään hyvä tulos. (Kaner ym., 2011; Myers ym., 2011)

Testaukseen liittyvät dimensiot tuovat esiin testauksissa esiintyvän rakenteen ja näihin liittyen pystytään määrittämään kymmenen perusperiaatetta testaukselle, jotta voidaan varmistaa kattava ja luotettava testaus. (Kaner ym., 2011; Myers ym., 2011)

Testaukseen liittyvät kymmenen perusperiaatetta ovat:

1. Oleellinen osa testitapausten suunnittelua on määritellä odotettu lopputulos.
2. Kehittäjän tulisi välttää oman ohjelmiston testaamista.
3. Ohjelmoivan organisaation ei tulisi testata omia ohjelmistojaan.
4. Kaikkien testausprosessien tulisi noudattaa kattavaa analysointia testauksen tuloksista jokaisessa testauksessa.
5. Testitapauksissa tulisi olla kirjallisesti kuvattuna ehdot, jotka eivät päde testitapaukseen, kuten myös siinä pätevät ja odotetut ehdot.
6. Ohjelman testaaminen sen osalta, että se tekee kuten halutaan, on vasta puolet testauksesta. Toinen puoli on testata sitä, mitä ohjelman ei pitäisi tehdä.
7. Kertakäyttöisiä testitapauksia tulee välttää, jos ohjelmaa testataan useammin kuin kerran.
8. Testitapauksia ei saa luoda siinä oletuksessa, ettei virheitä löydy.
9. Tuotannossa olevien virheiden määrä on verrannollinen testeissä löydettyjen virheiden määrään.
10. Testaaminen on äärimmäisen luovaa ja älyä haastava tehtävä. (Myers ym., 2011)

Ensimmäisen periaatteen mukaan testitapausten odotettu lopputulos tulee määritellä etukäteen, jotta vältytään tilanteilta, joissa järjestelmän virheellinen käyttäytyminen hyväksytään positiivisena testituloksena, sillä ihmisen peruspsykologian mukaan useimmiten henkilö pyrkii näkemään sen mitä haluaa nähdä. Järjestelmätestauksessa usein haluttu lopputulos on virheetön toiminnallisuus. Kun haluttu lopputulos on kirjallisesti määritetty ennen testauksen suorittamista, vertailu onnistuneesta tuloksesta on objektiivisempi ja tulkinnanvaraa on vähän. (Myers ym., 2011)

Toinen ja kolmas periaate perustuvat samaan ajatukseen eli kehittäjän tai kehitysorganisaation tulisi välttää oman kehityksen testausta. Tämä perustuu siihen, että oman tuotoksen näkee usein positiivisesta näkökulmasta ja virheiden löytämistä pyritään välttämään. Testauksissa näkökulman tulisi kuitenkin olla virheiden löytämisessä, minkä vuoksi oman kehityksen testaus aiheuttaa eturistiriidan. Toinen selkeä riski aiheutuu siitä, että kehittäjä tai kehitysorganisaatio on ymmärtänyt tarpeet väärin, sillä silloin heidän

näkökulmastaan kehitys voi olla virheetön, mutta asiakkaan näkökulmasta kehitys ei toimi odotetusti. Tämän vuoksi testaajana ei tule olla vain yhtä henkilöä. (Myers ym., 2011)

Neljännen periaatteen mukaan jokaisen testin tulokset on aina käytävä läpi. Tämä periaate on selkein, mutta usein se sivuutetaan. Tämän vuoksi kehityksen virheellinen toiminta saattaa jäädä huomaamatta testausten jatkuessa, vaikka yksinkertaisimmissa testitapauksissa virhe olisi ollut selkeästi huomattavissa ja virheellinen toiminta saattaa päätyä julkaistavaan tuotteeseen. (Myers ym., 2011)

Viides ja kuudes periaate sisältävät samoja komponentteja, sillä molemmissa korostetaan testitapausten luontia odotetulle toiminnallisuudelle sekä virheelliselle toiminnalle. Useimmiten testitapauksissa korostuu oletettu toiminta, mutta yhtäläisesti tulisi pitää mielessä virheellisen toiminnan mahdollisuus todellisessa elämässä. Tämän vuoksi testauksissa olisi syytä nähdä myös, miten järjestelmä toimii virheellisten toiminteiden kanssa. Tällöin voidaan nostaa todennäköisyyttä siihen, että kehitys toimii odotetusti myös virheellisen toiminnan osalta. (Myers ym., 2011)

Seitsemännen periaatteen mukaan tulisi välttää testitapauksia, jotka ovat kertakäyttöisiä. Tätä tapahtuu usein silloin kun keksitään testitapauksia lennosta kehityksen yhteydessä. Kuitenkin, jos kehitykseen kohdistuu myöhemmin päivitys, testauksen laadusta yleensä tingitään ja testauksen kattavuus ei ole samalla tasolla kuin käyttöönotossa. Jos käyttöönoton yhteydessä testitapaukset luodaan kestävästi, niitä pystytään käyttämään päivityksen yhteydessä ja testausten kattavuus pysyy samalla tasolla. (Myers ym., 2011)

Kahdeksannessa ja yhdeksännessä periaatteessa perusajatuksena on, ettei kehitys ole ikinä virheetön. Tämän vuoksi testitapauksia ei saa suunnitella sillä ajatuksena, ettei virheitä ole. Tämä rajoittaa testitapausten laatua, sillä kun virheitä ei haluta löytää, tehdään tapauksista usein suppeampia. Kehityksiin liittyy aina virheitä ja virheiden määrä korreloi siihen, kuinka monta virhettä on vielä löytämättä. Kun tämä hyväksytään, pystytään kattavampaa testausta ja resursseja investoimaan kehityksiin, joista on löytynyt eniten virheitä. (Myers ym., 2011)

Kymmenennen periaatteen mukaan testaus on luovuutta ja älykkyyttä vaativa tehtävä. Useimmiten testausten epäonnistuminen johtuu liian yksipuolisesta ajattelumallista ja, jos testitapaukset ovat liian samanlaisia keskenään, tuotteen käyttöönotto epäonnistuu, vaikka

testejä olisi suoritettu kymmenkertainen määrä. Testitapausten määrä ei korvaa tapausten laatua ja vaihtelevuutta. (Kaner ym., 2011; Myers ym., 2011)

3.3 Testauksen vaiheet

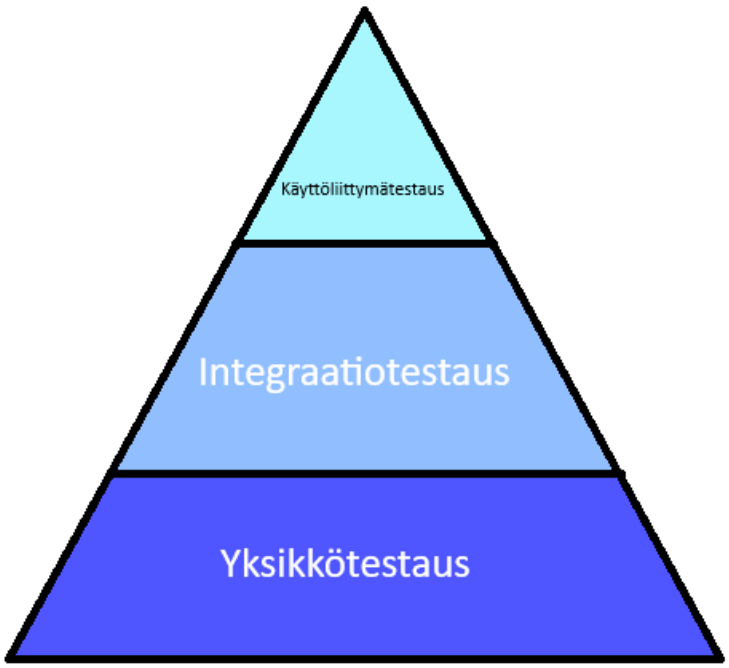
Testauksen voi jaotella kolmeen eri vaiheeseen, joita ovat käyttöliittymätestaus, integraatiotestaus ja yksikkötestaus. Yksikkötestauksen tarkoituksena on olla äärimmäisen nopea ja se on hyvä tapa saada nopea informaatio siitä, toimiiko koodi vai ei. Yksikkötestausten heikkoutena on, että ne voivat jättää asioita ulkopuolelle eivätkä ne ota kantaa integraatioiden toimivuuteen. (Rasmusson, 2016)

Integraatiotestausten tarkoituksena on nimensä mukaisesti varmistaa, että kehitykseen liittyvät yhteydet esimerkiksi toisista järjestelmistä ja verkkosivuista toimivat yhteen kehityksen kanssa. Integraatiotestausten heikkoutena on se, etteivät ne anna kovin tarkkaa palautetta mahdollisista virhekohdista. Vaikka virhe havaitaan, aiheuttajan löytäminen vaatii usein tarkempaa tutkimusta. (Rasmusson, 2016)

Käyttöliittymätestaus varmistaa, että kehitys toimii kokonaisuutena ja testauksen avulla voidaan varmistaa, että kehitys toimii kaikilla arkkitehtuurisilla tasoilla. Käyttöliittymätestauksen huonona puolena on sen hitaus ja testausten epävarmuus, jos niitä ei suorita ihminen. (Rasmusson, 2016)

Testauksen tasoja voi ajatella pyramidina, jossa pohjan muodostavat yksikkötestaukset, joita tulee olla määrällisesti eniten, tämän jälkeen tulee integraatiotestaukset, joita on lukumäärällisesti seuraavaksi eniten. Kestonsa vuoksi vähiten tulisi olla käyttöliittymätestauksia. Niitä ei voi kuitenkaan jättää kokonaan pois, sillä ne tarjoavat arvokasta tietoa siitä, että kehitys toimii alusta loppuun. Kuva 1 on testauspyramidi havainnollistettuna. (Rasmusson, 2016)

Kuva 1. Testauspyramidi



3.4 Testauksen automatisointi

Automatisoidulla testauksella tarkoitetaan kehityksissä tarvittavan ohjelmistotestauksen automatisointia. Tämä voi tarkoittaa yksikkö-, integraatio-, tai käyttöliittymätestauksen automatisointia. Toistuvaa testausta kannattaa automatisoida, jotta saadaan vähennettyä testauksen kustannuksia ja nopeutettua virheiden löytämistä sekä vähennetään inhimillisiä virheitä. Automatisoitujen testausten avulla yritys pystyy testaamaan huomattavasti enemmän kuin pelkän manuaalisen testauksen avulla, sillä automatisoitu testaus vapauttaa ihmisten aikaa testitapauksiin, jotka vaativat enemmän huomiota. (Garousi & Mäntylä, 2016; Rasmusson, 2016; Smartbear, ei pvm.)

Onnistuneeseen testausten automatisointiin ei kuitenkaan päästä automatisoimalla kaikkia testejä, joita ihmiset suorittivat aikaisemmin, vaan testauksen automatisointi tulee suunnitella huolella ja tehdä tarkoituksenmukaisesti. Väärin tehtynä automatisointi voi viedä turhia resursseja ja antaa jopa virheellistä palautetta järjestelmän toimivuudesta. (Garousi & Mäntylä, 2016; Rasmusson, 2016)

Kun lähdetään miettimään testauksen automatisointia, on hyvä tuntea ja tunnustaa manuaalisen testauksen vahvuudet verrattuna automatisoituun ja päinvastoin. Ihminen pystyy havainnoimaan konetta paremmin ja ymmärtämään mistä erilaiset virheet aiheutuvat, kun taas koneen vahvuutena on väsymätön työskentely samojen testausten parissa, joissa ihminen väsyisi nopeasti. Vahvuuksien tunnistaminen auttaa määrittelemään soveltuuko testitapaus automatisoitavaksi vai ei. (Dames, 2017)

Automatisoitavien testitapausten suunnittelussa kannattaa käyttää apuna olemassa olevia käytänteitä ja tarkistuslistoja, joiden avulla voi määritellä kannattaako testitapausta automatisoida vai ei. Esimerkiksi automatisoitavia testitapauksia voisi olla:

- Toistuvat ja paljon aikaa vievät testit.
- Tapaukset, joissa tulee yleisimmin inhimillisiä virheitä.
- Useita tietueita ja henkilöitä sisältävät testit.
- Liiketoimintakriittinen toiminne.
- Testaukset, joita ei pysty suorittamaan manuaalisesti. (Lowenthal, 2020)

Vaikka testitapaus kuuluisi johonkin yllä olevaan kategoriaan, se ei suoraan tarkoita, että testi olisi kannattava automatisoida. Esimerkiksi toistuva ja aikaa vievä testaus saattaa silti olla parempi suorittaa manuaalisesti, jos siihen kohdistuu paljon muutoksia ja vaatisi siten jatkuvaa automaation päivittämistä. Tällöin automatisointiin käytetyt investoinnit ylittäisivät saatavat hyödyt. (Lowenthal, 2020)

Testien automatisoinnissa tulisi pysyä kriittisenä, sillä mitä enemmän testitapauksia automatisoidaan, sitä enemmän kustannuksia todennäköisesti aiheutuu testiohjelmistojen ylläpidosta ja lisensoinneista. Automatisoidut testitapaukset vaativat myös henkilön, joka ylläpitää niitä päivitysten yhteydessä ja varmistaa, että testit ajautuvat suunnitelmien mukaisesti. Vaikka tietyn prosessiosan testaus on automatisoitu, voi toisinaan olla syytä ajaa myös manuaalisia testejä, sillä automatisoitu tapaus ajautuu aina samalla tavalla, kun taas ihminen voi muuttaa toimintatapojaan ja sitä kautta löytää uusia virheellisiä toiminteita. (Dames, 2017)

Yksi automatisointistrategia voisi olla käyttää Pareto-periaatetta, jonka mukaan 80 % vaikutuksista johtuu 20 % syistä. Tällöin kun 20 % testitapauksista automatisoidaan, saadaan 80 % kriittisistä toiminteista testattua. Strategian onnistumisen kannalta on

tärkeitä löytää oikeat 20 % automatisoitavista kohteista. Alussa kannattaa jutella kehittäjien kanssa, jotka pystyvät usein kertomaan mitä koodia käytetään useimmiten eri toiminnallisuuksissa tai mihin toiminteeseen viitataan useimmiten. Tämän jälkeen on selvitettävä, mitkä ovat liiketoiminnan näkökulmasta kaikista kriittisimpiä toiminnallisuuksia ja aiheuttavat siten suurimman liiketoiminnallisen riskin. Näiden identifioimiseen tarvitaan liiketoimintatuntemusta ja liiketoiminnan edustajien tulisi osallistua kriittisten toiminteiden määrittelyyn. (Dames, 2017)

Automatisoinneille tulee olla määritettynä elinkaari ja tarvittaessa tarpeettomia automatisointeja tulee poistaa tai muokata. Näin voi tapahtua silloin, kun prosessi muuttuu, jolloin arvioidaan uudestaan prosessin kriittisyys ja tarvittaessa muokataan automaatiota tai poistetaan se. Myös toistuvat testitulokset voi olla herätteitä tarkastella automatisoinnin tarpeellisuutta. Esimerkiksi, jos testitapaus tulee poikkeuksetta onnistuneena takaisin ja poistaminen ei sisällä liiketoimintariskejä, voi sen poistaminen olla suositeltavaa. Vastaavasti, jos tulos on toistuvasti epäonnistunut, tulee epäonnistumisen syitä tarkastella kehityksen lisäksi myös automatisoidun testitapauksen osalta. Onko testin mahdollista onnistua määritellyllä tavalla vai tuleeko testitapausta korjata. Jos testitapausta ei pysty korjaamaan, on se syytä poistaa käytöstä. (Dames, 2017)

Toinen testauksen automatisoinnin strategia voi olla pyramidin muodostaminen. Pyramidissa ajatuksena on, että määrällisesti eniten automatisoituina ovat testitapaukset, jotka ovat helppoja ja edullisia. Nämä ovat useimmiten yksikkötestauksia, joiden tuottaminen on nopeaa ja kehittäjä saa tällöin myös nopeasti palautteen toimiiko koodi vai ei. Pyramidin huipulla on testitapaukset, jotka ovat kalleimpia tuottaa ja ylläpitää, jolloin niiden automatisoinnin tulisi olla mahdollisimman vähäistä. Nämä pitävät sisällään käyttöliittymätestaukset. Pyramidin keskellä on integraatiotestaukset, jotka ovat hieman raskaampia tuottaa ja ylläpitää kuin yksikkötestaukset, mutta ovat kevyempiä kuin käyttöliittymätestaukset. (Rasmusson, 2016)

Testauksia automatisoidessa tulisi miettiä pystyykö joitain toiminteita vahvistamaan ainoastaan yksikkö- ja integraatiotestausten avulla. Pyramidi-strategian perusperiaatteena on suorittaa testaus aina mahdollisimman matalassa vaiheessa ja välttää päällekkäisiä testejä. Päällekkäiset toiminteiden testaukset ovat kuitenkin väistämättömiä ja ne tulee hyväksyä, käyttöliittymätestauksia tehdessä testataan myös useita toiminteita, jotka on katettu myös yksikkötestauksilla. (Rasmusson, 2016)

Kun automatisoitavat testitapaukset on määritelty ja tiedetään, onko kyseessä yksikkö-, integraatio- vai käyttöliittymätesti, valitaan automatisointiin soveltuva työkalu. Tällä hetkellä markkinoilla on useita käyttöliittymätestaukseen tarkoitettuja automatisointiohjelmistoja. Ne eivät vaadi koodaustaitoja vaan automatisoinnin pystyy tekemään esimerkiksi nauhoittamalla. Nauhoittamalla tehty testitapaus on kuitenkin usein toimivuudeltaan epävarma ja saattaa rikkoutua esimerkiksi resoluution vaihtuessa. Tämän vuoksi nauhoitettu testausautomaatio saattaa antaa jopa virheellistä testitulosta. (Rasmusson, 2016)

Toinen syy miksi automatisointi olisi hyvä olla koodattu eikä nauhoitettu, on saadun palautteen laatu. Koodilla tuotettu automatisointi yleensä antaa selkeämmän palautteen, joka ei vaadi lisätestauksia. Kolmantena syynä suosia koodia automatisoinneissa on sen tarjoama monipuolisuus monistettavuutena ja ymmärrettävyytenä. (Rasmusson, 2016)

4 Järjestelmät

Luvussa esitellään ohjelmistoja Excel ja UiPath sekä Visual Basics for Applications makroja, joita käytetään muun muassa Excel taulukoiden automatisoinnissa. Luvussa tarkastellaan myös ohjelmistojen ja automatisointien soveltuvuutta yrityksille kustannussäästöjen ja järjestelmien käyttöönoton näkökulmasta.

4.1 Microsoft Excel ja VBA-makrot

Microsoftin kehittämä Excel on taulukko-ohjelmisto, joka on nykyisin osa Microsoft 365 tuoteperhettä. Microsoft Excel on julkaistu ensimmäisen kerran vuonna 1985 ja Excelistä tuli markkinoiden johtava taulukkolaskentaohjelma 1990-luvun puolessa välissä. Markkinoille on tullut Excelin julkaisemisen jälkeen useita ohjelmistoja, joilla pystyy tekemään samankaltaisia toiminteita kuin Excelillä, mutta Excel on siitä huolimatta säilyttänyt asemansa ja nykyään yksi käytetyimmistä tietokoneohjelmistoista. (Habraken, 2002; Microsoft, ei pvm.; thecodest, 2018)

Excelin käyttäminen tietokantana on perusteltua, jos liiketoiminta ei tarvitse kehittyneempiä toiminteita. Excelin etuna on sen joustavuus ja tunnettuus maailmalla. Moni yritys vaatii työnhakijoiltaan kokemusta Excelin käsittelystä ja tunnettuuden vuoksi Excel taulukoissa

olevien tietojen tuonti ja vienti eri järjestelmiin onnistuu oletustoiminteiden avulla. (thecodest, 2018)

Excelin toimiessa tietokantana, yritys voi automatisoida työtehtäviä Visual Basic for Applications (VBA) makrojen avulla. Makrot ovat ohjelmia, jotka sisältävät käskysarjoja eli niitä komentoja, jotka käyttäjä antaisi suorittaessaan tehtävää. Tämän vuoksi toistuvien yksinkertaisten työtehtävien ohjelmointi makroille on kannattavaa. Makrot ovat sisäänrakennettuna Excelissä ja niiden käyttöönotto onnistuu aktivoimalla kehittäjätyökalut Excelin asetuksista. (Korol, 2019)

4.2 UiPath ja robotiikka-automaatio

UiPath on johtava Robotic Process Automation (RPA) ohjelmistoratkaisuja tarjoava yritys, joka on perustettu vuonna 2005 Romaniassa. Yritys tarjoaa työkaluja ja palveluita liiketoimintaprosessien automatisointiin järjestelmäriippumattomasti, mikä auttaa yrityksiä tehostamaan toimintaansa, kun yksinkertaiset usein toistuvat työt voi antaa RPA:lle. UiPathin tavoitteena on vapauttaa AI:n ja automaation potentiaali, joka mahdollistaa asiakkaille uusia mahdollisuuksia, suurempien tavoittelun ja saavuttaa enemmän. (UiPath Inc, 2025)

RPA:lla tarkoitetaan automaatiota, joka jäljittelee ihmisen toimia eri järjestelmissä. Automaatiota suorittava robotti toimii sääntöperusteisesti ja se ei pysty simuloimaan ihmisen päättelyprosessia perustuen saatuun tietoon. Automaatio RPA:lla soveltuu tehtäviin, joilla on toistuvuutta, selkeät säännöt ja vaativat paljon manuaalista työtä. Näiden työtehtävien automatisointi on perusteltua, sillä se vapauttaa työntekijöiden aikaa muun muassa päättelyä vaativiin tai valvontaa vaativiin tehtäviin. (Tripathi, 2018; UiPath Inc, 2025)

RPA:n ja testausautomaation yhtäläisyydet näkyvät kustannussäästöissä, jotka tulevat muun muassa manuaalisesti suoritettavien tehtävien vähentämisestä. Automatisoimalla liiketoimintaprosesseja ja järjestelmätestauksia, voidaan lisätä henkilöstön tyytyväisyyttä, sillä automaation avulla voidaan keventää työkuormaa ja siirtää huomio arvoon perustuviin tehtäviin. Vaikka automatisoinnin tekniikka ja saavutettavat hyödyt ovat osittain samoja, yhtenä tärkeänä erona on automaation tarkoitus. Liiketoimintaprosessien automatisointi

RPA:n avulla pyrkii muuttamaan työtehtävien laatua, kun testausautomaatiolla pyritään tehostamaan ohjelmistokehitystä. (Kit, 2023; Singureanu & Chernyak, 2023)

Etenkin testaukseen näkökulmalla on vaikutusta ja RPA:lla ei yksinään pystytä suorittamaan kattavia testauksia, minkä vuoksi RPA voi olla osa testauskokonaisuutta, mutta se ei pysty yksinään korvaamaan muun testauksen tarvetta. Tämä johtuu tekniikan rajoituksista, sillä RPA pystyy hyödyntämään yksinkertaisia ehtolausekkeita ja kopioimaan testaajan toimia järjestelmässä, kuten aikaisemmin todettiin, mutta se ei pysty analysoimaan tuloksia. (Singureanu & Chernyak, 2023)

5 RoboTech, sen tekninen ympäristö ja järjestelmät

Luvussa esitellään RoboTech yrityksenä ja sen tekninen ympäristö. Tämän lisäksi käydään läpi laskutukseen liittyvä prosessi, joka on yrityksen kriittisin automatisoitu prosessi yrityksen aloittaessa. Tämän jälkeen esitellään yleisesti Excel ja UiPath, joiden avulla laskutusprosessi on automatisoitu.

5.1 Yrityksen esittely ja tekninen ympäristö

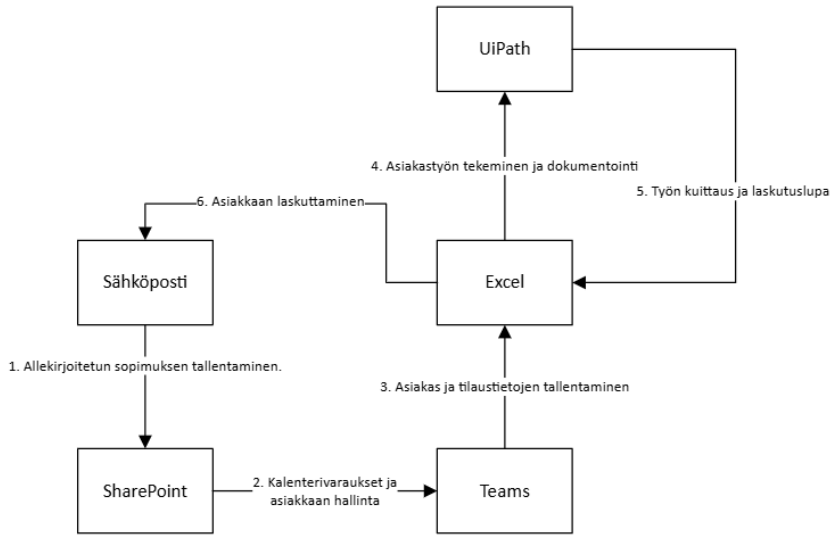
RoboTech ei ole vielä virallinen yritys vaan sen perustaminen on suunnitteluasteella. Yritys tulee keskittymään RPA-palveluiden tuottamiseen, koulutukseen ja konsultointiin yrityksille, jotta he pystyvät tehostamaan toimintaansa ja keskittymään oman liiketoiminnan kannalta oleellisiin työtehtäviin, kun prosessien automatisointi tarpeiden kartoitus ja kehittäminen on ulkoistettu RoboTechille.

Yrityksessä tulee työskentelemään kaksi ihmistä osapäiväisesti. Ensimmäinen perustaja on toiminut useissa erilaisissa liiketoiminta ja järjestelmäkehityksen rooleissa. Viimeisen viiden vuoden ajan henkilö on vastannut eri kokoisten yritysten tilaus-toimitus-laskutus-prosessien kehityspäällikkönä. Toisella perustajajäsenellä on yli yhdeksän vuoden kokemus teollisuuden liiketoimintaprosesseista ja yli viiden vuoden kokemus prosessiautomaatioista. Aikaisempaan kokemukseen perustuen RoboTech pystyy tarjoamaan kattavia konsultointi ja prosessikehityspalveluita.

Tavoitteena on päästä markkinoille muutaman asiakkaan turvin, kun perustajat ovat vielä kokopäivä töissä, mutta vuoteen 2030 mennessä yrityksen tulisi olla voitollinen, jotta perustajat pystyvät siirtymään täysipäiväisesti yrityksen palvelukseen.

Yritys pyrkii rakentamaan kulurakenteen mahdollisimman kevyeksi, minkä vuoksi vain pakolliset lisenssit ja ohjelmistot sekä taloushallintoon ja raportointiin liittyvät palvelut tullaan hankkimaan. Yritys pyrkii hyödyntämään tilaus- ja laskutusprosesseissa Exceliä tietokantana, jonka tulisi olla yrityksen ensimmäisten vuosien aikana riittävä hallinnallisesta näkökulmasta. Tästä ei aiheudu yritykselle kuluja, sillä perustajat omistavat riittävän Microsoft Office lisenssin. Lisenssiin sisältyy myös sähköposti, Teams ja SharePoint, joiden päälle perustuu asiakassuhteiden hallinta ja töiden aikataulutus. Kuva 2 on kuvattuna yrityksen käyttämät järjestelmät ja niiden osuus tilaus- ja laskutusprosesseissa.

Kuva 2. Hahmotelma yrityksen teknisestä ympäristöstä yhdistettynä tilaus- ja laskutusprosessiin.



Yritys tulee ulkoistamaan maksatukseen, kirjapitoon ja raportointiin liittyvät toimet, sillä näihin liittyen henkilöstöllä ei ole riittävän kattavaa kokemusta ja alue on tarkasti säädelty ja valvottu viranomaisten toimesta.

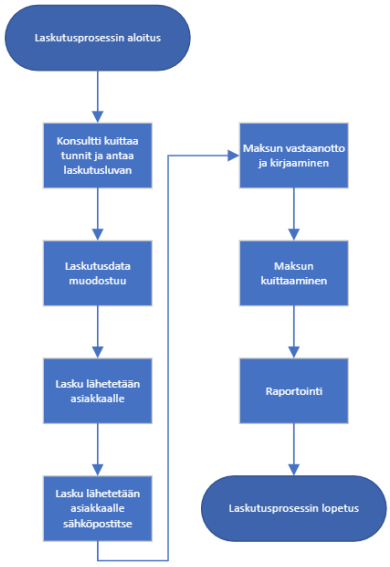
5.2 Töiden laskuttaminen RoboTechillä

Liiketoiminnan kannattavuuden ja yrityksen teknisen ympäristön kannalta liiketoimintakriittisin prosessi on laskutus, sillä se sisältää automaatiota ja yrityksen työntekijöiden näkökulmasta prosessin tulisi sitoa mahdollisimman vähän työtunteja, jotta työntekijät pystyvät keskittymään tuottavaan asiakastyöhön.

Laskun lähettäminen mahdollisimman pian laskutusluvan saatua, yritys varmistaa saavansa sovitun maksun mahdollisimman aikaisessa vaiheessa, mikä parantaa yrityksen kassavirtaa, sillä rahat siirtyvät oman yrityksen tilille kasvamaan korkoa. Tehokas laskutus antaa myös ammattimaisemman kuvan yrityksen toiminnasta, kun laskut saapuvat ajallaan ja laskussa olevat tiedot ovat oikein. Tämä vähentää selvitystyötä asiakkaan ja RoboTechin välillä, mikä parantaa asiakastyytyväisyyttä ja vapauttaa RoboTechin resursseja asiakastöiden suorittamiseen. (Yrittäjät.fi, ei pvm.)

RoboTechillä laskutusprosessi käynnistyy, kun konsultti kuittaa tehdyt tunnit ja antaa laskutusluvan. Hinnoitteluperusteena on pääsääntöisesti tuntihinta ja laskun summa muodostuu kaavalla tehdyt tunnit kerrottuna tuntihinnalla. Laskutusdatan muodostuttua lasku lähetetään asiakkaalle sähköpostitse. Asiakkaan suoritukset tarkistetaan manuaalisesti yrityksen tilille tulleiden suoritusten perusteella ja lasku kuitataan suljetuksi, kun viitenumero tiliotteessa vastaa avoimen laskun tilinumeroa. Laskutustietokanta toimii arkistona ja kerran kuussa oleellinen materiaali tietokannasta lähetetään taloushallintoon erikoistuneelle kumppanille, joka luo ja toimittaa tarvittavat raportit. Kuva 3 on kuvattuna yrityksen laskutusprosessi.

Kuva 3. RoboTech laskutusprosessi.



Laskutusprosessista on erotettu maksumuistutuksiin liittyvä prosessi, sillä maksumuistutusten lähetys tullaan suorittamaan tapauskohtaisesti. Tämä johtuu yrityksen ennustetusta koosta ja asiakkaiden rajallisesta määrästä. Maksumuistutukset liittyvät asiakkaan hallintaan ja tietyissä tilanteissa yritykselle voi olla kannattavampaa olla lähettämättä maksumuistutuksia. Lähettämisen sijaan yritys voi tuoda esiin maksamattomat laskut asiakkaan kanssa pidettävissä kuukausipalavereissa.

6 RoboTech testausstrategia ja -automaatio

Luvussa vastataan kaikkiin kolmeen tutkimuskysymykseen ja käytännön kehitys perustetaan vastauksiin. Luvussa luodaan testitapaukset Visio-ohjelmistoa hyödyntäen ja tapausten perusteella luodaan automaatio UiPathilla.

6.1 Kehitys- ja testausstrategia yrityksessä

Yrityksen liiketoiminta perustuu IT-palveluiden tuottamiseen yrityksille, minkä vuoksi yrityksellä tulee olla toimiva ja selkeä linjaus kehityksessä käytettävän viitekehyksen osalta sekä toimiva kehitysprosessi.

Yrityksen toimialalla on vallitsevana viitekehyksenä agile-viitekehys, minkä vuoksi yritys valikoi sen myös omaksi kehitysmallikseen. Kehitystiimin vetäjänä toimii yrityksen toinen perustajajäsen, kehittäjänä toinen ja asiakkaan yhteyshenkilö on myös osa tiimiä. Hänen vastuullaan on osallistua vaatimusten määrittelyyn ja UAT-testaukseen. Näin pystytään varmistamaan, että asiakkaan toiveet tulevat täytetyksi sekä mahdollisiin ongelmiin löydetään vastaukset nopeasti.

Sisäisessä kehitystyössä käytetään myös agile-mallia, joka määrittelee myös yrityksen testausstrategiaa. Kuten teoriaosuudessa mainittiin, startup-yritysten ei tule määritellä raskasta testausdokumentaatiota, sillä ongelmia ratkaistaan lennosta. Yrityksen jäsenten perusajatuksena on kuitenkin noudattaa ketterää testausstrategiaa, jotta yritys varmistaa kaikkien sidosryhmien osallistumisen ja näkökulmat testauksessa.

6.2 Testausautomaatio laskun lähetyksessä

Kustannusten karsimiseksi yritys ei käytä perinteisiä ohjelmistoja laskutuksessa ja tarjoaa vain sähköpostilaskutusta asiakkailleen. Tämän vuoksi ennen yrityksen perustamista tulee varmistaa kehitetyn järjestelmän toimivuus. Yrityksen tarkoituksena on luoda automatisoidut skenaariot käyttöliittymätestaukseen, jotka pyörivät UiPath-ohjelmiston päällä. Näin voidaan varmistaa, että Excelissä olevat VBA-makrot toimivat oletetusti ja kaikki laskutettavat työt päätyvät laskutukseen. Automatisoitujen testitapausten yhteydessä ei käytetä yksikkötestauksia VBA-koodin yksinkertaisuuden ja puuttuvien relaatioiden vuoksi.

Järjestelmän toimivuuden ja hyvinvoinnin lisäksi, toinen etu UiPathilla tehdyissä testitapauksissa, joiden avulla yritys pystyy automatisoimaan järjestelmätestaustaan, on niiden myynnillinen arvo. Kun yritys pystyy näyttämään asiakkaalle, miten he ovat luoneet käyttöliittymään automatisointeja, on asiakkaan helpompi nähdä RPA:n tuottama arvo sekä yrityksen henkilöstön asiantuntijuus. Tällä pystytään nopeuttamaan myyntiprosessia ja lisäämään todennäköisyyttä kaupan toteutumiselle.

Tilaus-laskutus-prosessin työnkulussa ensimmäiset vaiheet ovat manuaalisia ja ihminen vaikuttaa niiden onnistumiseen, mutta kun tilaukselle annetaan laskutuslupa, sitä seuraavat vaiheet ovat täysin automaattisia. Kuva 4 on piirretty tilaus-laskutus-työnkulku, jossa

manuaaliset työvaiheet näkyvät viivalla ja automaattiset työvaiheet ovat merkitty katkoviivalla.

Kuva 4. Tilaus-laskutus työnkulku.

Kuten aikaisemmin mainittiin, Pareto-periaatteen mukaan tulee löytää 20 % testitapauksista, jotka aiheuttavat 80 % vaikutuksista. Järjestelmä ei ole vielä aktiivisessa käytössä, joten analyysi tapahtuu liiketoimintariskin perusteella. Haavoittuvin työvaihe on

VBA-makro, sillä se ei ymmärrä uusia tai virheellisiä syötteitä ja jos käyttäjä ei ole täyttänyt Exceliä annettujen ohjeiden mukaisesti tai on siirrellyt sarakkeiden järjestystä, makro hajoaa ja yritykseltä ei lähde laskuja asiakkaalle. Tämän vuoksi VBA-makron ajo ja tulosten tarkasteleminen on pääasiallinen automatisoinnin kohde. (Dames, 2017)

Tilaus-laskutusprosessin ollessa kevyt, on houkutus lähteä automatisoimaan koko prosessin testaus, mutta teorian perusteella sitä ei suositella. Tämä johtuu siitä, että prosessi on nuori ja todennäköisyydet sen kehittämiselle on suuria, minkä vuoksi testaukseen liittyvien automatisointien elinkaari olisi hyvin lyhyt ja yrityksen työntekijät joutuisivat käyttämään kohtuuttomasti resursseja automatisointien päivittämiseen ja ylläpitoon, joka on pois tuottavasta asiakastyöstä. Tämän vuoksi toistaiseksi testauksen automaatiot keskittyvät VBA-makron luotettavuuteen. (Dames, 2017; Lowenthal, 2020)

6.3 VBA-makron toiminta ja automatisoidut testitapaukset

VBA-makron toiminta laskutusprosessissa perustuu tietojen koostamiseen eri välilehdiltä yhteen, jotta RPA pystyy poimimaan tiedot asiakkaalle lähetettävään laskuun. Automatisoinnilla saadaan vähennettyä manuaalisen työntarvetta laskutusprosessissa ja nopeutettua yrityksen kassavirtaa, koska laskut lähtevät ajallaan työn kuittaamisen jälkeen. Yrityksen ei tarvitse myöskään palkata erillistä henkilöä hoitamaan laskutusta tai ulkoistaa prosessia, mikä säästää kustannuksissa.

Testausautomaation elinkaari kestää niin pitkään, kun yritys käyttää Exceliä tilaus- ja laskutustietokantana. Kun yritys ottaa käyttöön kehittyneemmän ohjelmiston, kuten esimerkiksi toiminnanohjausjärjestelmän, Exceliin luodut testauksen automatisoinnit voi poistaa käytöstä. Yrityksen poistaessa käytöstä olemassa olevat automatisoinnit, tulee sen analysoida automatisointitarpeet uudelleen. Todennäköistä on, että uusi järjestelmä tukee työvaiheita paremmin ja testaus on kannattavampaa automatisoida eri työvaiheiden osalta.

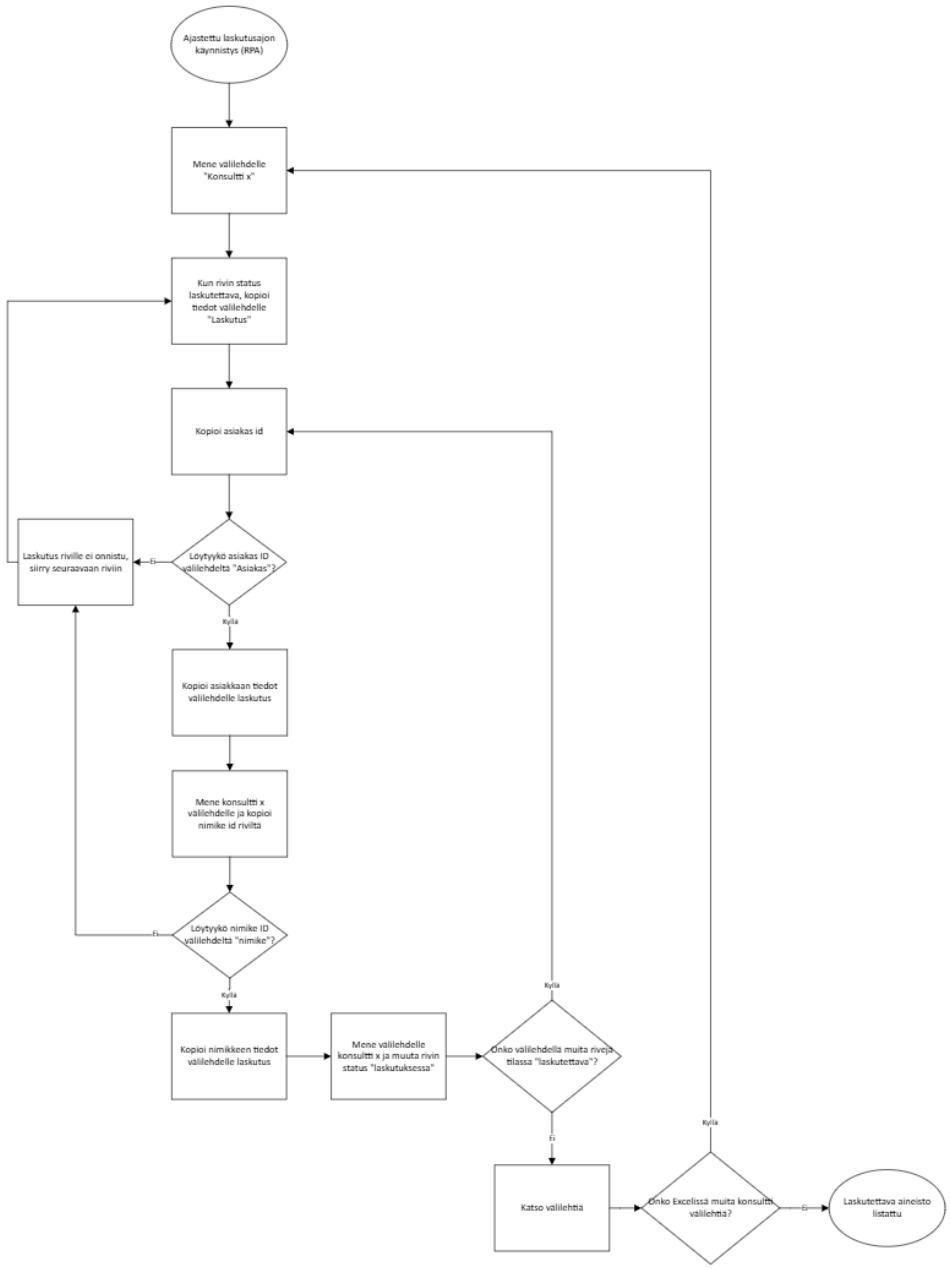
Testaukset suoritetaan aina testijärjestelmässä, jotta voidaan varmistaa, etteivät testauksessa luotu aineisto vaikuta tuotannon toiminteisiin. Tuotannossa testaamisessa suurena riskinä on testauksessa luodun laskutusaineiston lähteminen asiakkaalle asti, mikä työllistää asiakasta, sillä virheellinen lasku työllistää asiakkaan ostoreskontraa sekä osto-osastoa, jonka pitää selvitellä virheellistä laskua. Tämä vaikuttaisi negatiivisesti

RoboTechin ja asiakkaan välisiin suhteisiin sekä antaisi yrityksestä ammattitaidottoman kuvan.

Toinen huomioitava asia automatisoitujen testitapausten luonnissa, on tietoturvaan liittyvät käytänteet. Järjestelmässä on asiakastietokanta, johon on kerätty muun muassa laskutusasiakkaiden sähköpostiosoitteita. Jotta vältytään henkilötietoihin liittyviltä haasteilta, testauksissa käytetään tunnettujen sarjakuvasankarien nimiä. Tarkoituksena on nimien helppo tunnistettavuus, joka viittaa suoraan siihen, ettei kyseessä ole todellinen laskutusaineisto vaan testitapaus. Toinen etu tunnistettavan nimen käytössä on se, että sarakkeen tarkoitus on tunnistettavissa, vaikka otsikkoa ei olisi näkyvissä. Sanojen kuten ”testi” tai ”testaaja” kanssa, sarakkeen tarkoitusta ei saisi selville ilman otsikon lukemista.

Kuva 5 on kuvattuna VBA-makron työnkulku, joka on yrityksen liiketoiminnan kannalta kriittinen, minkä vuoksi se toimii perustana testausautomaatiolle. Työnkulku liittyy laskutettavan aineiston siirtämiseen ”Konsultti x”-välilehdiltä ”Laskutus”-välilehdelle ja jos työnkulku ei toimi oikein, yritys ei laskuta asiakkaitaan tehdyistä töistä, mikä on perusteena liiketoimintakriittisyydelle.

Kuva 5. VBA-makron työnkulku.

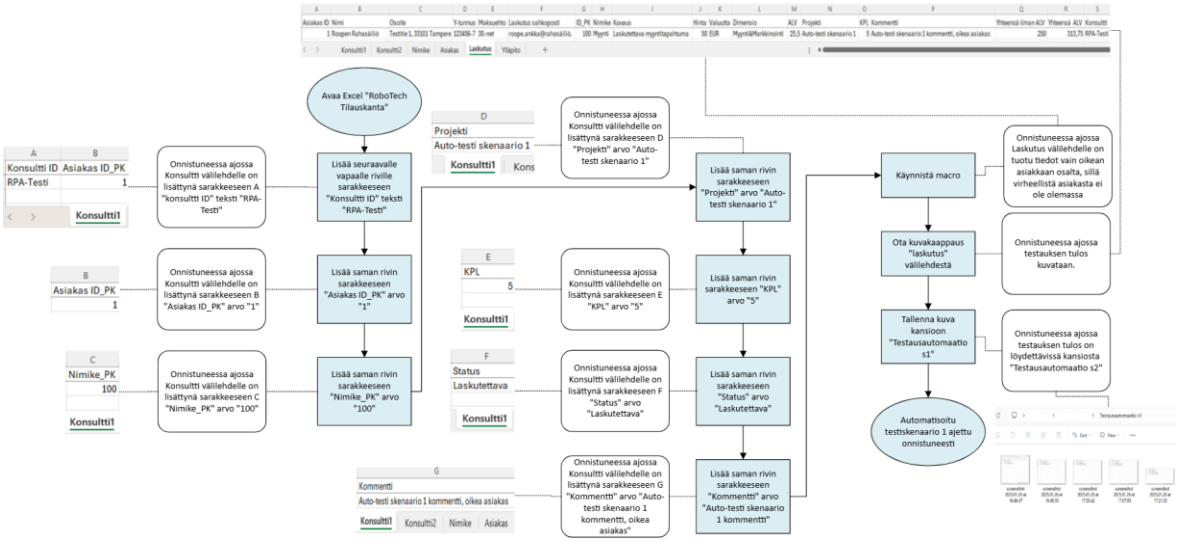


Liitteessä 2 on yrityksen käyttämä koodi laskutusaineiston luonnille Excelissä VBA-makrolla luotuna. Koodin avulla vähennetään manuaalista työtä yhdistämällä dataa eri välilehdiltä ja valvomalla, että laskutettavat tunnit päätyvät laskutusaineistoon. Jos tässä koodissa tulisi haasteita, riskinä on, ettei yritys laskuta kaikkia tunteja.

Ensimmäinen automatisoitava testitapaus on varmistaa, että oikein täytetty laskutettava rivi päätyy laskutusvälilehdelle. Tällä vahvistetaan, että makro toimii oikein ja se poimii kaikki

laskutuksessa tarvittavat tiedot kuten pitää. Kuva 6 on kuvattuna ensimmäinen automatisoitava testiskenaario.

Kuva 6. Ensimmäinen automatisoitu testiskenaario



Testaukseen liittyvien parhaiden käytänteiden mukaan, testattaessa tulee myös huomioida tilanteet, joissa käyttäjä ei toimi oikein. Tällä pyritään varmistamaan, että järjestelmä selviytyy kyseisistä tilanteista kaatumatta. Tarkastellessa makron työnkulkua, meillä on neljä ehtolauseketta, jolloin riski ohjelman väärälle toiminnalle tai kaatumiselle on suurin. (Dames, 2017)

Testausautomaatioita ei suositeltu olevan liian paljon, koska ne syövät resursseja ja aiheuttavat kustannuksia, tämä on prioriteettina myös RoboTechin tilanteessa, etenkin kun testausautomaatio käyttää samaa ohjelmistoa, kuin tuottava liiketoiminta. Tämän vuoksi valitessa testitapausta käytetään Pareto-periaatetta. Tällä kertaa kulmana on mikä aiheuttaa 80 % ongelmista, eikä mikä on liiketoimintariskinä suurin. (Dames, 2017)

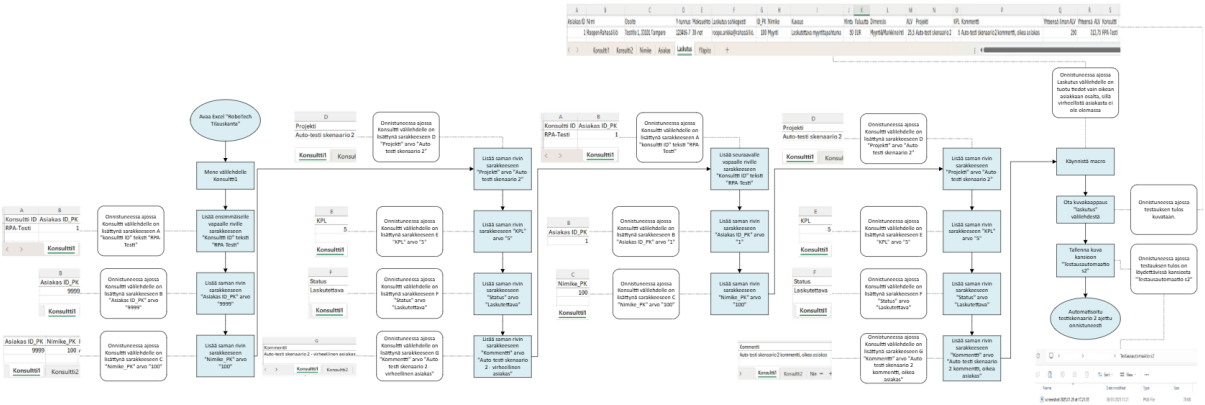
Syy sille, että automatisoinnissa ei keskitytä liiketoimintariskiin, johtuu siitä, että liiketoiminnan kannalta suurimmat ongelmat aiheutuvat, jos koodi ei huomaa kaikkia konsulttivälilehtiä, jolloin luonnollisesti rivit eivät päädy laskutukseen. Nämä virheet ovat kuitenkin helpommin huomattavissa päivittäisessä työssä ja juurisyy on tekninen, minkä vuoksi korjauksen koodiin tulisi poistaa ongelma ja toistuvuutta ei pitäisi olla, minkä vuoksi perustetta testauksen automatisoinnille ei ole.

Excelissä yksi mahdollinen syy koodin toimimattomuudelle on käyttäjän tekemät virheet ja datan laatu, minkä vuoksi testauksen automatisoinnin tulisi kohdistua asiakas- tai nimikedataan. Riski koodin toimimattomuuteen on korkea, sillä Excel ei ole sovelluksen kaltainen tietokanta, jossa pystyy määrittelemään minkälaisia käyttäjäsyötteitä, hyväksytään ja yritys ei ole vielä tehnyt rajoitteita, joilla suojellaan taulukkoa käyttäjän tekemiltä muutoksilta. Esimerkiksi käyttäjä pystyy toistaiseksi siirtämään tai poistamaan sarakkeita, jolloin koodi lakkaa toimimasta. (Microsoft, ei pvm.)

Datan laatuun liittyen, yritys ylläpitää ja säilyttää asiakas- ja nimiketietokantaa. Näistä kahdesta eniten datan päivitystä tapahtuu asiakaslistassa, koska tavoitteena on saada useita uusia asiakkaita ja mahdollista on myös, että osa asiakkaista poistuu, kun taas tuotteet pyritään pitämään mahdollisimman vakioituina ja muutoksia tapahtuu harvoin. Tämän vuoksi toiseksi automatisoitavaksi testiskenaarioksi valikoitui tilanne, jossa asiakasta ei ole vielä perustettu järjestelmään, vaikka yritys on tehnyt töitä asiakkaalle ja laskutuslupa on annettu. Oletuksena on, ettei järjestelmä kaadu, vaan siirtyy seuraavalle käsiteltävälle riville.

Toinen automatisoitava testitapaus pystyy hyödyntämään samoja komponentteja kuin ensimmäinen testitapaus, mikä nopeuttaa automatisoinnin luomista ja vähentää automatisoinnin käyttöönoton kustannuksia. Kuva 7 on kuvaus toisen testitapauksen työkulusta.

Kuva 7. Toinen automatisoitu työnkulku



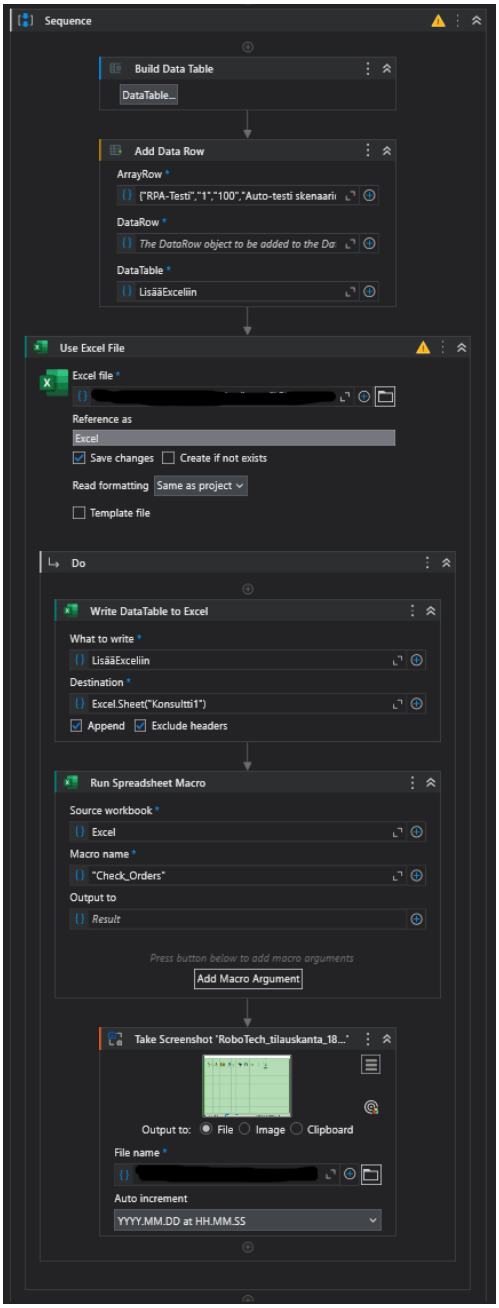
Ennen yrityksen perustamista ja järjestelmien käyttöönottoa, yrityksen tulee tehdä kattava testaus, joka pitää sisällään tuotannon simuloinnin alusta loppuun sekä kattavan UAT-testauksen. Kahden testitapauksen automatisointi, ei korvaa ihmisen tekemää testausta, mikä on testauksen automatisointiin liittyvän teorian vahvana viestinä. (Dames, 2017)

6.4 Skenaarioiden automatisointi UiPathilla

Ensimmäinen skenaario luotiin sekvenssinä, koska työnkulku on lyhyt ja sisältää vain muutaman aktiviteetin. Työnkulku alkaa taulun rakentamisella UiPathin sisällä, jonka jälkeen taululle annetaan vakioarvot, jotka on määritelty testin suunnitteluvaiheessa. Kun arvot on annettu, avataan RoboTechin tilauskanta Excelissä, ja arvot kirjoitetaan ensimmäiselle vapaalle riville 'Konsultti1' välilehdelle.

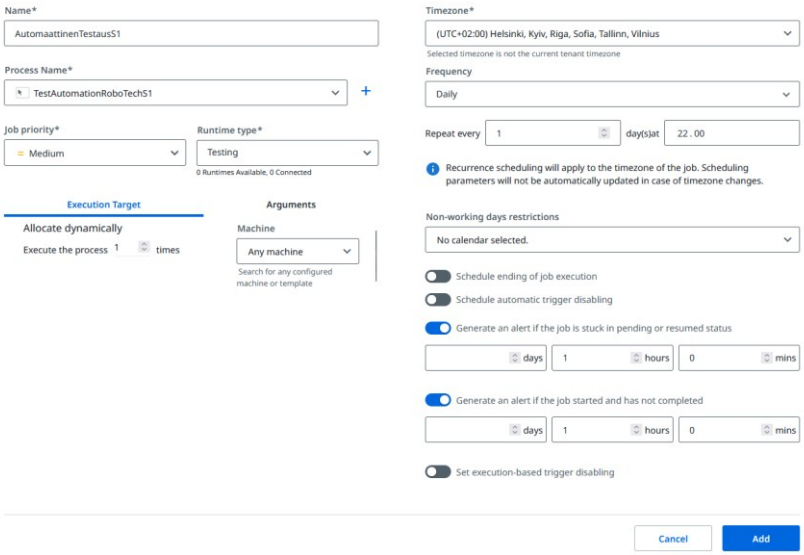
Tämän jälkeen käynnistetään Excelin sisällä oleva makro, joka on pääasiallisena testauksen kohteena, UiPathin aktiviteetilla ”Run Spreadsheet Macro”. Kun makro on ajautunut, otetaan Excelistä kuvakaappaus, joka tallennetaan kansioon ”Testausautomaatio s1”. Tämä toiminne päättää työnkulun. Kuva 8 on kuvakaappaus työnkulusta UiPathissa.

Kuva 8. UiPath työkulku ensimmäiselle skenaariolle



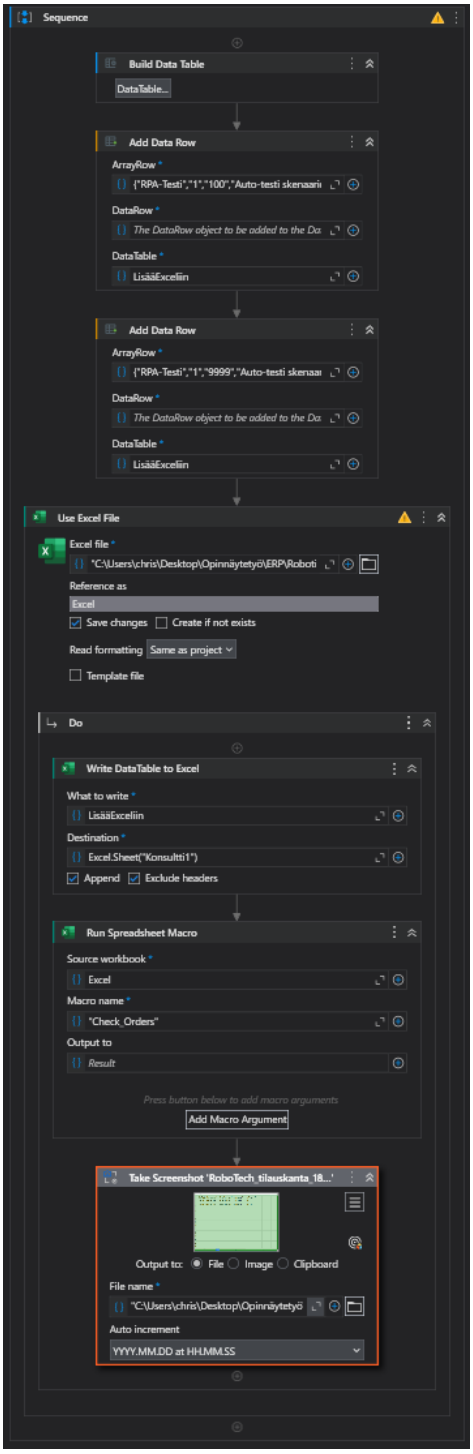
Robotti on ajastettu ajamaan aina kello 22:00 (UTC +2), jotta testaukset tapahtuvat laskutusajojen jälkeen. Seuraavana työpäivänä on mahdollista varmistaa kuvakaappauksesta, että testaus on suoritettu oikein. Kuva 9 on kuvakaappaus työnkulun ajastuksesta.

Kuva 9. Työnkulun ajastus ensimmäiselle automatisoidulle testiskenaariolle



Toinen testiskenaario rakentuu samalla periaatteella kuin ensimmäinen. Erona on tilausrivien lisäyksessä virheellisen asiakasnumeron käyttö. Ensimmäinen automatisoitu testitapaus on kopioitu ja lisätty toinen ”Add data row” aktiviteetti. Ensimmäisestä testiskenaariosta päivitetään myös kentät ”Projekti” ja ”Kommentti” kuvaamaan toista testiskenaariota. Kuvakaappaus tallennetaan myös toiseen kansioon ja polku tulee päivittää vastaamaan uutta kansiota. Kuva 10 on kuvakaappaus toisesta UiPathilla luodusta työnkulusta.

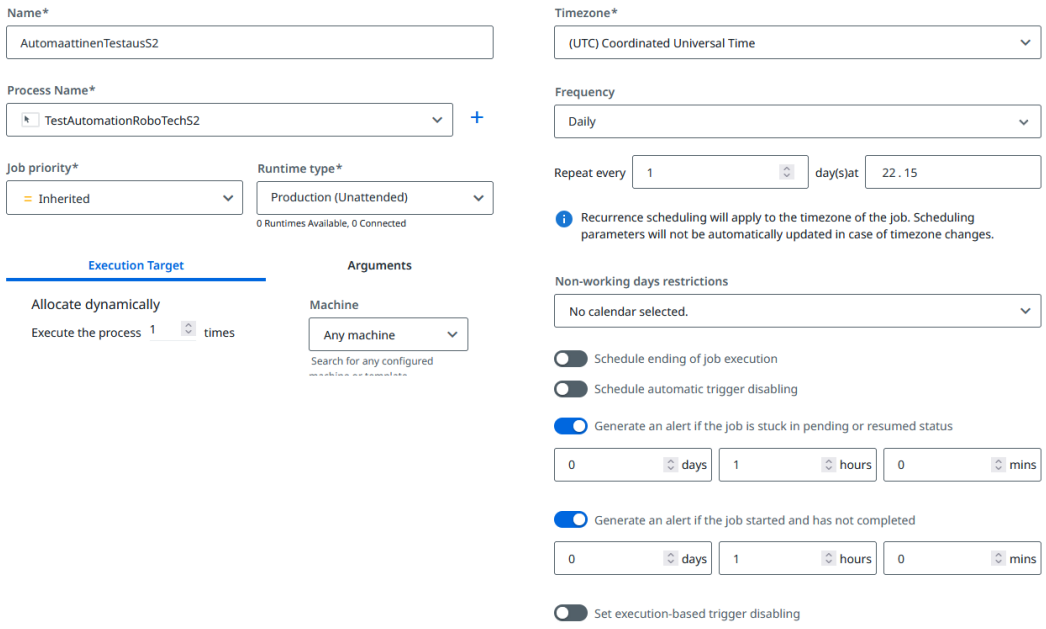
Kuva 10. UiPath työkulku toiselle skenaariolle



Robotti on ajastettu ajautumaan 22:15 (UTC +2), jotta ensimmäinen testausskenaario on ehtinyt ajautumaan ennen seuraavan ajon aloitusta. Automatisointien ajo kestää joitain sekunteja, mutta pidempi väli turvaa toista testiautomaatiota. Kun automatisoituja tapauksia

on enemmän, tulee ajastukset olla tehokkaammin järjestetty ja ajojen välillä lyhyempi turvaväli. Kuva 11 on kuvakaappaus testauksen ajastuksesta UIPath Orchestratorista.

Kuva 11. Työnkulun ajastus toiselle automatisoidulle testiskenaariolle



7 Tutkimuksen analysointi

Luvussa käydään läpi tutkimuksen tuloksia sekä niihin vastaamisen onnistuneisuutta. Tutkimuksen lähtökohtana oli vastata kolmeen kysymykseen, joiden avulla RoboTech pystyisi varmistamaan, että testaukseen liittyen yritys noudattaisi alan parhaita käytänteitä. Tämän varmistaminen on oleellista, sillä yritys pyrkii tarjoamaan konsultointi ja kehityspalveluita prosessiautomaatioihin liittyen, minkä yrityksen uskottavuuden vuoksi olisi tärkeää.

7.1 Ensimmäisen tutkimuskysymyksen analyysi

Ensimmäinen tutkimuskysymys oli ”Mitä suositellaan testausstrategiaksi startup-yritykselle?”. Teorioiden mukaan startup-yrityksille ei suositella testausstrategiaa, sillä se on turhan raskas. Teoriassa mainittiin, että henkilöiden lukumäärän vuoksi testaustyylin ja -laajuuden kommunikointi onnistuu usein helposti tietoa tarvitseville työntekijöille. Teorian

esittämä ajatus poikkesi täysin ennakkokäsityksistä, joiden perusteella jokaisella yrityksellä, joka tekee järjestelmäkehitystä asiakkaille tai sisäisesti tulisi olla testausstrategia. (AltexSoft, 2024)

Testausstrategiaan liittyvät teoriat ja suositukset ovat kuitenkin hyödyllistä tietoa yritykselle, sillä ne säästivät henkilöstöä käyttämästä aikaa dokumentaation luontiin ja elinkaaren määritykselle. Tutkimus lisäsi myös tietoisuutta yrityksen sisällä ja sen perusteella pystyttiin luomaan linjavetoja testauksissa käytettävistä malleista. Tulevaisuudessa kun yritys toivottavasti kasvaa tavoitteiden mukaan ja henkilöstöä tulee lisää, pystytään testausstrategian luontiin palaamaan.

7.2 Toisen tutkimuskysymyksen analyysi

Toisena tutkimuskysymyksenä oli ”Millaista testausautomaatiota suositellaan RoboTechille teorioiden perusteella?”. Tähän liittyen käytännönosassa tehtiin teorian suositusten vastaisesti, sillä teorian mukaan yrityksen tulisi automatisoida helppoja ja edullisia testiskenaarioita, jotka ovat käytännössä yksikkötestauksia. Kuitenkin yrityksen testausautomatisointi perustuu käyttöliittymätestaukseen, jota teorian mukaan tulisi olla määrällisesti vähiten. (Rasmusson, 2016)

Työssä perusteltiin päätöstä koodin yksinkertaisuudella, minkä vuoksi käyttöliittymätestaus on riittävä. Jälkikäteen ajateltuna yrityksen olisi kannattavaa analysoida järjestelmän käyttämä koodi ja miettiä olisiko siellä tapoja testata kriittisiä osia koodista, joka lähettäisi viestin heti kun koodi ei toimi oikein. Tällöin pystyttäisiin varmistamaan liiketoiminnan kannalta kriittisen prosessin toimivuus ja ongelmiin pystyttäisiin puuttumaan ennen kuin niillä on vaikutusta liiketoimintaan tai yrityksen kassavirtaan.

Muuten automatisointitapa noudatti teorian suosituksia, sillä teoriassa kehotettiin automatisoimaan toistuvaa testausta ja tähän automatisoidut testitapaukset soveltuivat. Ilman automatisointia, ne tulisi testata henkilön toimesta säännöllisesti, johtuen työnkulun toimimattomuuden aiheuttavista riskeistä yrityksen liiketoiminnalle. Automatisointeja ei myöskään luotu nauhoittamalla, jonka luotettavuus on huono ja sen antamaan palautteeseen ei aina pysty luottamaan. (Garousi & Mäntylä, 2016; Rasmusson, 2016; Smartbear, ei pvm.)

7.3 Kolmannen tutkimuskysymyksen analyysi

Kolmantena tutkimuskysymyksenä oli ” Mitkä ovat liiketoimintakriittiset automatisoitavat testitapaukset yrityksen tilaus- ja laskutusjärjestelmässä?”. Automatisoitavien testitapausten tunnistamisessa Pareto-periaate toimii hyvin. Periaatetta on käytetty myös muissa yrityksissä usein järjestelmäkehityksissä, kun kehityksen halutaan korjaavan 80 % ongelmista kehittämällä 20 % toiminteista, jotta saadaan resursseja käytettyä mahdollisimman tehokkaasti. (Dames, 2017)

Sen arvioiminen onnistuttiinko työssä löytämään oleelliset 20 % automatisoitaviksi kohteiksi on hieman haastavaa yrityksen elinkaaren ollessa alussa ja näin ollen arvioinnin pohjana ei voida käyttää testauksessa tai prosesseissa ilmenneitä ongelmia. Uskoisin automatisoitavien prosessien valinnan osuneen oikeaan, mutta yrityksen tulisi analysoida omia prosessejaan sekä saatuja asiakaspalautteita säännöllisesti ja tarvittaessa automatisoida saatujen tietojen pohjalta uusia prosesseja.

Regressiotestaus ohjelmistotestauskäytänteenä sopii testitapausten määrittelyyn tulevaisuudessa, mutta kun yritys on vasta ottamassa järjestelmiä käyttöön, aikaisemmin suoritettujen testien dataa ei ole käytettävissä. Kuitenkin kun yritys lähtee kehittämään sisäisiä järjestelmiään, tulisi heidän hyödyntää aikaisempia testauksia automatisoinneissa ja varmistaa etteivät testausten tulokset muutu järjestelmän kehittyessä.

8 Yhteenveto

Tutkimuskysymyksiin vastaaminen onnistui pääsääntöisesti kattavasti ja tutkimuskysymyksiin vastattiin kolmen eri dimension avulla, joita olivat teoria, käytäntö ja ohjelmistot. Tutkimuskysymykset toimivat opinnäytetyön runkona erilaisia teorioita lukiessa, ne toimivat ankkurina, joiden avulla työn teoriaosuus ei lähtenyt leviämään liikaa. Ohjelmistotestaukseen liittyvää kirjallisuutta ja artikkeleita on paljon tarjolla, minkä vuoksi opinnäytetyön kirjoituksessa tärkeää oli pitää aihe tutkimuksen kannalta oleellisissa teorioissa.

Toinen dimensio eli käytäntö huomioitiin soveltaessa teorioita käytäntöön, sillä moni teorioista pohjautui useamman työntekijän ja osaston yritysten näkökulmaan. Tällöin teorioita ei pystytty ottamaan käyttöön suoraan vaan niiden ajatuksia piti soveltaa. Osa teorioiden suosituksista voidaan ottaa käyttöön, kun yritys on ollut toiminnassa muutaman kuukauden ja työnkulut ovat muotoutuneet paremmin kokonaisuuksiksi.

Kolmantena dimensiona toimi ennakkoon valitut ohjelmistot, joita yritys tulee käyttämään. Tutkimuskysymyksiin vastatessa ohjelmistojen ominaisuudet huomioitiin ja testaukseen liittyvien automatisointien tuli soveltua ajettavaksi valituilla ohjelmistoilla. Ohjelmat vaikuttivat myös tutkittuihin teorioihin.

Henkilökohtaisesti ohjelmistotestaus on ollut suurena kiinnostuksen kohteena jo useiden vuosien ajan ja aikoinaan osallistuminen järjestelmätestaukseen testaajan roolissa, sai siirtymään järjestelmäkehitysten pariin. Opinnäytetyön tekeminen on syventänyt osaamista ja ymmärrystä eri testausautomatisointi tyyppeihin liittyen, vaikka automatisoinnit tehtiin tutummalla käyttäjätestaus-mallilla. Tutkimus on myös lisännyt itsevarmuutta testauksen alalla, sillä osa käytännössä opituista totuuksista ja malleista saivat vahvistuksen teorioista. Toki tutkimusta tehdessä oli myös tilanteita, jossa päädyin korjaamaan aikaisempia ajatuksia.

Tulevaisuudessa pystyn hyödyntämään opittua sekä päivätyössäni, että RoboTechin kehittämisessä. Etenkin RoboTechin osalta on tärkeää, että yritys toimii alan parhaiden käytänteiden mukaan ja saa mainetta asiantuntijuudesta sekä tehokkaista toimintatavoista. Tämä tuo aloittavalle yritykselle uskottavuutta ja toivottavasti saa vakautettua aseman markkinoilla.

Lähteet

Aalto University. (1997, kesäkuuta 2). *Testaussuunnitelma*.

http://www.soberit.hut.fi/tik-76.115/96-97/palautukset/groups/apina/p1/documents/ts.html

AltexSoft. (2024, syyskuuta 20). *Test Strategy: Does it Make Sense and How to Document It*. AltexSoft. https://www.altexsoft.com/blog/test-strategy/

Baumgartner, M., Klonk, M., Mastnak, C., Pichler, H., Seidl, R., & Tanczos, S. (2021). *Agile Testing: The Agile Way to Quality*. Springer Nature. https://books.google.fi/books?id=omNCEAAAQBAJ&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Crispin, L., & Gregory, J. (2008). *Agile Testing: A Practical Guide for Testers and Agile Teams*. Pearson Education. https://www.google.fi/books/edition/Agile_Testing/68_IhPvoKS8C?hl=fi&gbpv=1&dq=Agile+Testing:+A+Practical+Guide+for+Testers+and+Agile+Teams&printsec=frontcover

Dames, K. (2017, heinäkuuta 20). A Lean Approach to Automation Testing. *Teal Times*. https://medium.com/teal-times/a-lean-approach-to-automation-testing-ad6fcdcf41e3

Garousi, V., & Mäntylä, M. V. (2016). When and what to automate in software testing? A multi-vocal literature review. *Information and Software Technology*, *76*, 92–117. https://doi.org/10.1016/j.infsof.2016.04.015

Habraken, J. W. (2002). *Microsoft Excel 2002: 10 Minute Guide*. Que Publishing.

Hinton, R. (2022, heinäkuuta 13). *How to Write a Test Strategy Document for Software Testing—NearForm*. https://www.nearform.com/blog/how-to-write-software-test-strategy-document/

Ite, Wiki. (2019, toukokuuta 27). *Testiautomaatio | Ite wikin digitalisoinnin opas*. https://www.itewiki.fi/opas/testiautomaatio/

Kaner, C., Bach, J., & Pettichord, B. (2011). *Lessons Learned in Software Testing: A Context-Driven Approach*. John Wiley & Sons. https://www.google.fi/books/edition/Lessons_Learned_in_Software_Testing/byZmT73R1a8C?hl=fi&gbpv=1&dq=Practical+Test+Design+Selection+of+traditional+and+automated+test+design+techniques&printsec=frontcover

Katalon. (2023, joulukuuta 10). *How To Write Test Strategy? Complete Guide With Sample*. Katalon.Com. https://katalon.com/resources-center/blog/test-strategy

Kit, I. (2023, marraskuuta 14). *RPA's Impact on Employee Morale: Expert Analysis by Scimus*. https://thescimus.com/blog/impact-of-rpa-on-employee-morale-and-satisfaction/

Korol, J. (2019). *Microsoft Excel 2019 Programming Pocket Primer*. Stylus Publishing, LLC.

Lowenthal, J. (2020, marraskuuta 19). *How to Get Started Testing: Best Test Cases to Automate*. SmartBear.com. https://smartbear.com/blog/how-to-get-started-testing-best-test-cases-to-auto/

Marick, B. (1998). *When Should a Test Be Automated?* http://www.exampler.com/testing-com/writings/automate.pdf

Microsoft Excel | Kuvaus ja historia—Teknologia | Tammikuu 2025. (ei pvm.). Noudettu 18. tammikuuta 2025, osoitteesta https://fi.gov-civ-guarda.pt/microsoft-excel

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing*. John Wiley & Sons. https://malenezi.github.io/malenezi/SE401/Books/114-the-art-of-software-testing-3-edition.pdf

Pesonen Teemu. (2022, elokuuta). *Ohjelmistoautomaatio*. VALA. https://www.valagroup.com/wp-content/uploads/2022/08/testiautomaatio-opas-2022.pdf

Rasmusson, J. (2016). *The Way of the Web Tester: A Beginner's Guide to Automating Tests*. Pragmatic Bookshelf. https://www.google.fi/books/edition/The_Way_of_the_Web_Tester/jg9QDwAAQBAJ?hl=fi&gbpv=1&dq=The+Way+of+the+Web+Tester:+A+Beginner%27s+Guide+to+Automating+Tests&printsec=frontcover

Rice, R. (2023, joulukuuta 10). *How to Create a Software Test Strategy? Complete Guide With Examples* [Article]. https://www.practitest.com/resource-center/article/how-to-create-a-software-test-strategy

Shinde, V. (2012, joulukuuta 7). *How To Write Test Strategy Document (With Sample Test Strategy Template)*. Software Testing Help. https://www.softwaretestinghelp.com/writing-test-strategy-document-template/

Singh, R. (2023, tammikuuta 8). *How to Create a Test Strategy Document: A Comprehensive Guide*. https://www.headspin.io/blog/a-step-by-step-guide-to-mastering-test-strategy-documents

Singureanu, C., & Chernyak, A. Z. (2023, heinäkuuta 31). RPA vs. Testausautomaatio—Yhtäläisyydet ja erot. *https://www.zaptest.com/fi*. https://www.zaptest.com/fi/rpa-vs-testausautomaatio-yleiskatsaukset-yhteiset-piirteet-erot-ja-risteyskohdat

Smartbear. (2023, joulukuuta 9). *Test Automation Best Practices | SmartBear*. https://smartbear.com/learn/automated-testing/best-practices-for-automation/

Solutions, V. (2025). Mikä on regressiotestaus? Määritelmä, työkalut ja parhaat käytännöt. *Visure Solutions*. https://visuresolutions.com/fi/blogi/mik%C3%A4-on-regressiotestauksen-m%C3%A4%C3%A4ritelm%C3%A4-ja-parhaat-ty%C3%B6kalut/

Soni, M., & Soni, J. (2024). *Basics of Microsoft Excel*.

thecodest. (2018, marraskuuta 20). *ONKO EXCEL-TIETOKANTA VANHAA KOULUKUNTAA? KYLLÄ VAI EI - The Codest*. Codest. https://thecodest.co/fi/blog/onko-excel-tietokanta-vanhan-koulukunnan-kylla-vai-ei/

Tiukova, A. (2024, huhtikuuta 18). *Software Product Testing Strategy and Best Practices*. MobiDev. https://mobidev.biz/blog/software-testing-strategy-best-practices

Tripathi, A. M. (2018). *Learning Robotic Process Automation: Create Software robots and automate business processes with the leading RPA tool – UiPath*. Packt Publishing Ltd.

Tuohisto, V., & Saari, L. (2023, lokakuuta 24). *Ohjelmistotestauksen johtaminen liiketoiminnan näkökulmasta*. VALA. https://www.valagroup.com/fi/blogi/ohjelmistotestauksen-johtaminen-liiketoiminnan-nakokulmasta/

UiPath Inc. (2025, tammikuuta 25). *About UiPath—Accelerate Human Achievement | UiPath*. https://www.uipath.com/about-us

Unadkat, J. (2023, maaliskuuta 22). *Automated UI Testing: Benefits, Challenges & Solution*. BrowserStack. https://browserstack.wpengine.com/guide/what-is-automated-ui-testing/

Vala. (2023a, lokakuuta 24). *Mitä on ohjelmistotestaus ja mitä hyötyä siitä on?* VALA. https://www.valagroup.com/fi/blogi/mita-on-ohjelmistotestaus-ja-mita-hyotya-siita-on/

Vala. (2023b, lokakuuta 24). *UAT testaus, eli hyväksymistestaus: Mitä se tarkoittaa ja miksi se on tärkeää?* [Blogi]. VALA. https://www.valagroup.com/fi/blogi/hyvaksymistestaus/

Vertaa kaikkia Microsoft 365 -palvelupaketteja | Microsoft. (ei pvm.). Noudettu 18. tammikuuta 2025, osoitteesta https://www.microsoft.com/fi-fi/microsoft-365/business/compare-all-microsoft-365-business-products

Yrittäjät.fi. (ei pvm.). *Laskutus*. Yrittajat.fi. Noudettu 8. helmikuuta 2025, osoitteesta https://www.yrittajat.fi/tietopankki/verot-ja-talous/laskutus/

Yushkevich, N. (2024, syyskuuta 9). *How to create a test strategy: Types, components, and template*. Easy Way to Create a Test Strategy: Key Components, Types and Impact on the Project. https://www.zebrunner.com/blog-posts/easy-way-to-create-a-test-strategy-key-components-types-and-impact-on-the-project

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyön aineiston kuvaus

Opinnäytetyön aineistona on käytetty yrityksen järjestelmäarkkitehtuuriin liittyviä piirustuksia sekä työnkulkujen kuvauksia. Kaikki suunnitteluun liittyvä aineisto on luotu Microsoft Visiolla. Aineistot ovat luotu osana yrityksen liiketoimintamallien luomista, mutta niitä on hyödynnetty opinnäytetyössä, sillä työ perustuu työnkulkujen testaukseen. Kuvat on tallennettu PDF-tiedostoina.

Yrityksen käytössä oleva VBA-koodi on luotu työnkulkujen perusteella ja koodi on tallennettuna tekstitiedostona, joka toimii myös koodin varmuuskopiona. Yrityksen toinen omistaja on myös opinnäytetyön tekijä ja hän omistaa myös aineiston.

Aineiston tallennus ja säilytys

Aineistoa käsitellään pääsääntöisesti opinnäytetyön tekijän omalla koneella paikallisesti tallennettuna. Visio-kuvausten osalta on hyödynnetty opiskelijatunnuksia ja nämä tiedostot ovat tallennettuna myös opiskelijan OneDrivelle. OneDrive on suojattu kaksivaiheisella tunnistautumisella, jotka ovat salasana ja Microsoft Authenticator. Konetta ei ole suojattu salasanalla, mutta sitä säilytetään vain yrityksen tiloissa, joka on murtohälyttimellä suojattu.

Aineistolla ei ole säännöllistä varmuuskopiointia ja koneen kaatuessa, yrityksen tietokanta ei olisi palautettavissa. VBA-makro on kuitenkin jaettu opinnäytetyössä ja sen palauttaminen onnistuisi, mutta tietokannassa olevat tiedot katoaisivat.

UiPathin kautta luotu aineisto testausautomaatioihin liittyen pystytään luomaan uudestaan työssä olevien kuvakaappausten perusteella tarvittaessa ja automaatiota ei ole toistaiseksi varmuuskopioitu. Tietoturvallisen ja toimivan varmuuskopioinnin luonti liittyen sekä teknisiin toiminnallisuuksiin, että datan käsittelyyn on yrityksen seuraava kehityskohde.

Henkilötietojen ja arkaluonteisten tietojen käsittely

Opinnäytetyössä ei käsitellä henkilötietoja.

Aineiston omistajuus

Aineiston omistajuus on RoboTechillä, jolle tutkimustyö tehtiin.

Aineiston jatkokäyttö työn valmistumisen jälkeen

Opinnäytetyön aineistoa hyödynnetään RoboTechin liiketoiminnan perustamisessa ja käyttöönotossa. Tutkimuksen kautta saatuja tietoja tullaan hyödyntämään prosessien ja toimintamallien kehityksessä.

Liite 2: VBA-makron koodi

```
Sub Check_Orders()

Dim konsultti1WS As Worksheet
Dim konsultti2WS As Worksheet

    Set konsultti1WS = ThisWorkbook.Sheets("Konsultti1")
    Set konsultti2WS = ThisWorkbook.Sheets("Konsultti2")

    Call Check_Konsultti_Orders(konsultti1WS)
    Call Check_Konsultti_Orders(konsultti2WS)

End Sub

-----------------------

Sub Check_Konsultti_Orders(ws As Worksheet)

Dim konsulttiLastrow As Long
Dim LaskutusWS As Worksheet
Dim laskutusLastrow As Long
Dim i As Long
Dim cellValue As Variant
Dim sumValue, AlvValue As Double
Dim priceValue, qtyValue As Double
Dim PrimaryKey As Variant

    Set LaskutusWS = ThisWorkbook.Sheets("Laskutus")

    laskutusLastrow = LaskutusWS.Cells(LaskutusWS.Rows.Count,
"A").End(xlUp).Row
    konsulttiLastrow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

    ' Go through rows in Konsultti sheet
    For i = 2 To konsulttiLastrow

        cellValue = ws.Cells(i, 6).Value

        If LCase(cellValue) = "laskutettava" Then

            ' Copy needed values from Asiakas to Laskutus
            PrimaryKey = ws.Cells(i, 2).Value
            Call Fetch_Asiakas_data(LaskutusWS, laskutusLastrow,
PrimaryKey)

            ' Copy needed values from Nimike to Laskutus
            PrimaryKey = ws.Cells(i, 3).Value
            Call Fetch_Nimike_data(LaskutusWS, laskutusLastrow,
PrimaryKey)

            ' Copy needed values from Konsultti to Laskutus
            Call Fetch_Konsultti_data(LaskutusWS, laskutusLastrow,
ws, i)

            ' Calculate price wo ALV
```

```
            priceValue = CDbl(LaskutusWS.Cells(laskutusLastrow +
1, 10).Value)
            qtyValue = CDbl(LaskutusWS.Cells(laskutusLastrow + 1,
15).Value)

            ' Calculate sum
            sumValue = priceValue * qtyValue
            LaskutusWS.Cells(laskutusLastrow, "A").Offset(1, 16) =
sumValue

            ' Calculate price with ALV
            AlvValue = CDbl(LaskutusWS.Cells(laskutusLastrow + 1,
13).Value) / 100 + 1
            LaskutusWS.Cells(laskutusLastrow, "A").Offset(1, 17) =
sumValue * AlvValue

            ' Change item status to Laskutuksessa in Konsultti
sheet
            ws.Cells(i, 6) = "Laskutuksessa"

        End If

        ' Update last row index from Laskutus sheet
        laskutusLastrow = LaskutusWS.Cells(LaskutusWS.Rows.Count,
"A").End(xlUp).Row

    Next i

End Sub

-----------------

Sub Fetch_Asiakas_data(LaskutusWS As Worksheet, laskutusLastrow As
Long, PrimaryKey As Variant)

Dim AsiakasWS As Worksheet
Dim i, j As Long
Dim asiakasLastrow As Long
Dim cellValue As Variant

    Set AsiakasWS = ThisWorkbook.Sheets("Asiakas")

    asiakasLastrow = AsiakasWS.Cells(AsiakasWS.Rows.Count,
"A").End(xlUp).Row

    For i = 2 To asiakasLastrow

        If AsiakasWS.Cells(i, 1).Value = PrimaryKey Then

            ' Copy values from each column from the row to
Laskutus
            For j = 1 To 6

                cellValue = AsiakasWS.Cells(i, j).Value
                LaskutusWS.Cells(laskutusLastrow, "A").Offset(1, j
- 1) = cellValue

            Next j

            Exit For
```

```
        End If

    Next i

End Sub

-----------------

Sub Fetch_Nimike_data(LaskutusWS As Worksheet, laskutusLastrow As
Long, PrimaryKey As Variant)

Dim NimikeWS As Worksheet
Dim i, j As Long
Dim nimikeLastrow As Long
Dim cellValue As Variant

    Set NimikeWS = ThisWorkbook.Sheets("Nimike")

    nimikeLastrow = NimikeWS.Cells(NimikeWS.Rows.Count,
"A").End(xlUp).Row

    For i = 2 To nimikeLastrow

        If NimikeWS.Cells(i, 1).Value = PrimaryKey Then

            ' Copy values from each column from the row to
Laskutus
            For j = 1 To 7

                cellValue = NimikeWS.Cells(i, j).Value
                LaskutusWS.Cells(laskutusLastrow, "A").Offset(1, j
+ 5) = cellValue

            Next j

            Exit For

        End If

    Next i

End Sub

------------------

Sub Fetch_Konsultti_data(LaskutusWS As Worksheet, laskutusLastrow
As Long, KonsulttiWS As Worksheet, rowIndex As Long)

Dim cellValue As Variant

    ' Copy needed values to Laskutus
    ' Konsultti
    cellValue = KonsulttiWS.Cells(rowIndex, 1).Value
    LaskutusWS.Cells(laskutusLastrow, "A").Offset(1, 18) =
cellValue

    ' Projekti
    cellValue = KonsulttiWS.Cells(rowIndex, 4).Value
    LaskutusWS.Cells(laskutusLastrow, "A").Offset(1, 13) =
cellValue
```

```
    ' KPL
    cellValue = KonsulttiWS.Cells(rowIndex, 5).Value
    LaskutusWS.Cells(laskutusLastrow, "A").Offset(1, 14) =
cellValue

    ' Kommentti
    cellValue = KonsulttiWS.Cells(rowIndex, 7).Value
    LaskutusWS.Cells(laskutusLastrow, "A").Offset(1, 15) =
cellValue

End Sub
```