Haaga-Helia
ammattikorkeakoulu Oy

# Automatisointi saapuvien sähköisten tilausten käsittelyyn

Felix Hallenberg

Opinnäytetyö
Tietojenkäsittelyn ko.
2019

**Tiivistelmä**

Haaga-Helia ammattikorkeakoulu Oy

**Tekijä(t)**
Felix Hallenberg

**Koulutusohjelma**
Tietojenkäsittelyn koulutusohjelma

**Raportin/Opinnäytetyön nimi**
Automatisointi saapuvien sähköisten tilausten käsittelyyn

**Sivu- ja liitesivumäärä**
32+0

Ohjelmistorobotteja käytetään automatisoimaan tietotyön rutiinitehtäviä. Ohjelmistorobotit työskentelevät käyttöjärjestelmässä ja eri sovelluksissa suorittaen erilaisia toimenpiteitä askel kerrallaan. Ohjelmistorobottien kehittäminen on nopeaa, sillä uuden ohjelmakoodin kirjoittamisen sijasta robotteja konfiguroidaan.

Tämän opinnäytetyön tavoitteena oli luoda automatisaatioprosessi DSV Road Oy:n saapuvien tilausten käsittelyyn. Opinnäytetyö rajattiin koskemaan sähköisiä vientitilauksia. Tilaukset voidaan jakaa tuontitilauksiin, vientitilauksiin ja cross-country-tilauksiin. Tuontitilauksessa tilauksen määrämaa on Suomi ja lähtömaa joku muu. Vientitilauksessa lähtömaa on Suomi ja määrämaa on joku muu. Cross-country -tilauksissa rahdinmaksaja on useimmiten Suomessa, mutta lähtö- ja määrämaa ei ole Suomi. Tämän jälkeen tilaukset voidaan jakaa edelleen manuaalisiin- ja sähköisiin tilauksiin. Manuaalinen tilaus on tilaus, jonka asiakaspalvelija kirjaa järjestelmään. Sähköinen tilaus saapuu suoraan asiakkaalta, asiakkaan operaattorin kautta tai muun DSV:n sähköisen portaalin kautta, kuljetusjärjestelmään.

Työn taustalla oli logistiikka-alan yrityksen DSV Road Oy:n, Customer Care -osaston eli asiakaspalveluosaston tarve. Osastolla työaikaa käytettiin liialti tilausten käsittelyyn, johon haettiin automatisointia määrämuotoisten tilausten osalta. Säästettävä aika voitaisiin kohdentaa haastavampiin tehtäviin. Tilausten käsitteleminen on omiaan ohjelmistorobotiikalle: aineistomäärältä on suuri ja aineisto sisältää määrämuotoista tietoa.

Opinnäytetyön tekijä suunnitteli ratkaisun ja toteutti sen yhdessä toisen asiantuntijan kanssa. Nelihenkinen tiimi kokoontui viikoittain projektin parissa. Opinnäytetyön tekijä oli suunnitellut ja toteuttanut ratkaisusta proof of concept -version syksyn 2018 aikana. Kun ratkaisu todettiin toimivaksi, aloitettiin projekti laajennuksena. Projektin ensimmäinen versio meni tuotantoon 27.03.2019. Lopullisena tuotteena syntyi viidestä ohjelmistorobotista ja yhdestä VBA6 ohjelmistomakrosta koostuva kuljetusten hallintajärjestelmässä saapuvia tilauksia käsittelevä kokonaisuus.

Opinnäytetyö onnistui, sillä kehittäminen pysyi pääosin ennalta asetetun aikataulun mukaisena ja lopullinen tuote onnistui tavoitteessaan eli säästämään aikaa asiakaspalveluosaston toiminnassa. Opinnäytetyötä voidaan kehittää ottamalla mukaan myös sähköiset tuontitilaukset ja tekoälysovellutuksen avulla myös sähköpostitse saapuvat manuaaliset tilaukset. Etiikan kannalta on syytä myös tarkastella millaisina ohjelmistorobotin aikaansaamat muutokset näkyvät työntekijöiden arjessa. Kokevatko työntekijät, että ohjelmistorobotti vähentää rutiinityössä syntyvää kiirettä ja stressiä? Toisaalta kehitetäänkö työntekijöiden osaamista muulla tavalla ja siirretään haastavampiin tehtäviin?

**Asiasanat**
ohjelmistorobotiikka, toimistoautomaatio, kuljetusala

## Sisällys

# 1 Johdanto

Craig, Lacity ja Willcocks (2015b, 3) kertovat, että yritykset voivat karsia kuluja seuraavilla keinoilla: toimitilojen ja budjettien keskittäminen, prosessien standardoiminen liiketoimintayksiköiden välillä, prosessien optimointi, kalliilta alueelta muuttaminen halvemmalle alueelle, teknologian hyödyntäminen ja automatisointi. Viisi ensimmäistä keinoa ovat erittäin käytettyjä. Yritysmaailmassa muuttaminen, palveluiden keskittäminen ja prosessien yhtenäistäminen on arkipäivää. Viimeisenä mainittu automatisointi on toistaiseksi ollut muita pienemmässä käytössä. Craig ym. (2015b, 3–4) jatkavat, että vuonna 2012 oli käännekohta, jolloin ohjelmistorobotiikaksi kutsuttu uusi teknologia on noussut työkaluksi organisaatioiden toiminnan tehostamiseksi.

Ohjelmistorobotiikka on teknologia, jonka avulla voidaan tehdä tietotyöntekijän rutiinitöitä. Ohjelmistorobotiksi kutsuttu ohjelma työskentelee sovelluskerroksessa ohjelmoitujen sääntöjen mukaisesti suorittaen erilaisia toimenpiteitä vaihe vaiheelta kuin ihminen. Ohjelmistorobotiikka on yleistymässä kovaa vauhtia, sillä tietotyötä tehdään paljon. Ohjelmistorobottien kehittäminen on myös halpaa, sillä ratkaisuiden luomista varten ei tarvitse kirjoittaa uusiksi järjestelmien tai rajapintojen ohjelmakoodia: riittää vain, että mallintaa työntekijää.

Tämän toiminnallisen opinnäytetyön tavoitteena oli luoda prosessi, jonka avulla käsitellään Suomen DSV Roadin Customer Care -osaston saapuvia sähköisiä tilauksia. Kutsun osastoa myöhemmin asiakaspalveluosastoksi. Opinnäytetyöni toimeksiantaja on kansainvälinen logistiikka-alan yritys DSV. DSV on maailman viidenneksi suurin logistiikkatoimija, joka toimii yli 80:ssa maassa. Suomen DSV on jaettu kolmeen yritykseen, jotka työllistävät 280 työntekijää.

DSV konsernilla on teknologian ikään nähden pitkät juuret ohjelmistorobotiikan saralla, mutta Suomen DSV:llä ensimmäiset robotit otettiin käyttöön vasta 2018 ja nyt yrityksen työtehtäviä hoitamassa on noin kaksikymmentä robottia. DSV:n listauksen mukaan koko konsernilla on käytössään melkein kaksisataa robottia tuotantoympäristössä. Suomessa työskentelevät robotit on kehitetty DSV Road Oy:n Business Change Management -osastolla, jossa myös tämä opinnäytetyö tehtiin.

Asiakaspalveluosastolla tehdään selvityspyyntöjä ja kirjataan saapuvia tilauksia. Tilaukset voidaan jakaa sähköisiin- ja manuaalisiin tilauksiin. Manuaalinen tilaus on sähköpostitse tai puhelimitse tehtävä tilaus, jonka työntekijä kirjaa järjestelmään. Sähköinen tilaus saapuu sanomana suoraan asiakkaalta, ulkoisen operaattorin kautta tai DSV:n omista

sovelluksista. Vientitilaus määritellään niin, että tilauksessa lähtömaa on Suomi ja määrämaa joku muu. Tuontitilauksessa lähtömaa on joku muu ja määrämaa on Suomi. Tämän lisäksi on olemassa niin sanottuja cross-country -tilauksia. Näissä tilauksissa useimmiten rahdinmaksaja on Suomessa, mutta lähtö- ja määrämaa ei ole Suomi. Opinnäytetyö rajattiin koskemaan prosessin ensimmäistä tuotantoversiota, jossa käsitellään vain sähköisiä vientitilauksia.

Aloitan opinnäytetyön kertomalla ohjelmistorobotiikasta yleisesti. Tämän jälkeen esittelen teknologioita, joita käytin opinnäytetyötä tehdessäni. Käytin DSV:llä olemassa olevia teknologioita opinnäytetyön tekemiseen eikä muita uusia vaihtoehtoja kartoitettu. Esittelen DSV:n kuljetusten hallintajärjestelmän Cargolinkin, Cargolinkin kanssa käytettävän ohjelmointikielen Visual Basic 6:n sekä DSV:n käytössä olevan Kofax Kapow ohjelmistorobotiikkatuotteen. Esittelen myös organisaatioiden välisen tiedonsiirron taustaa, vaikka tämän opinnäytetyön produktin kannalta sillä ei ole suurta merkitystä. Suunnitelman ymmärtämiseksi on tärkeä tietää miten tieto välittyy DSV:n kuljetusten hallintajärjestelmään.

Luvussa 3 avaan ympäröivää liiketoimintaympäristöstä ja projektin taustalla olevaa ongelmaa. Automatisoinnin perimmäinen haaste oli se, että asiakkaiden tilausprosessit erosivat toisistaan ja heillä oli erilaiset valmiudet tuottaa tilaustietoa. Toisekseen asiakkaiden sopimukset saattoivat erota toisistaan, joka vaikuttaa tilausten käsittelyyn. Luvussa myös kerron yksityiskohtaisesti suunnitelmasta, prosessin syntymisestä ja miten testaaminen on suoritettu.

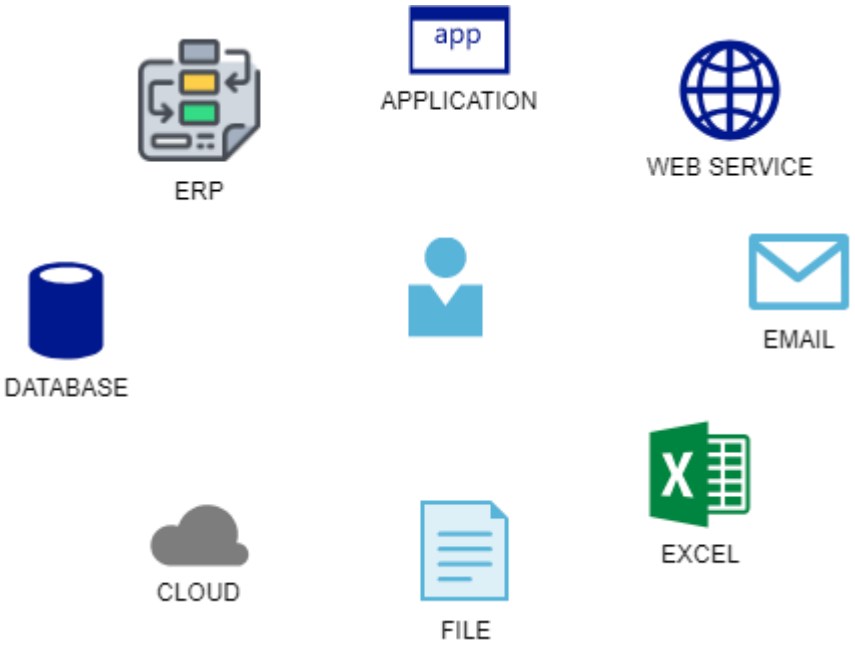
Lopuksi pohdinnassa käsittelen projektin tarpeellisuutta, eettisiä näkökohtia. Kehittämis- ja jatkotutkimukskohteita sivutaan myös nopeasti. Aivan viimeiseksi tiivistän opinnäytetyöprosessista oman oppimisen arvioinnin.

# 2 Ohjelmistorobotiikka ja käytössä olevat teknologiat

## 2.1 Ohjelmistorobotiikka

### 2.1.1 Taustaa

Kolehmaisen (2016) mukaan ohjelmistorobotiikka on kasvava teknologianala, jolla automatisoidaan tietohallinnollisia prosesseja. Ohjemistorobotiikka on oikeastaan huono termi kuvaamaan teknologiaa, sillä ohjelmistorobotiikassa ei ole kyse oikeista fyysisistä roboteista. Holmukhen, Madakamin & Jaiswalin (2019, 4) mukaan ohjelmistorobotiikka tarkoittaa ohjelmistoa, joka tekee ihmisen työtä. Helsingin Yliopiston ja Reaktorin (2018) Elements of AI -kurssin alussakin muistutetaan, että ilman fyysistä rajapintaa edes tekoälyä sisältäviä ohjelmistorobotiikkaratkaisuja ei voida laskea mukaan oikeaan robotiikkaan.




Kuva 1. Toimistoautomaatio

Kuva 1 kuvastaa toimistoautomaatiota. Toimistoautomaatiossa työntekijät liikkuvat eri sovelluksien, tietokantojen, Web-palvelujen, sähköpostin ja muiden työkalujen välillä. Työ sisältää paljon kopioimista, liittämistä, tietojen tallentamista, tarkastamista ja niin edelleen. Tämä vie paljon aikaa ja aiheuttaa organisaatiolle kuluja. Craigin ym. (2015a, 5) mukaan juuri tätä ongelmaa voidaan hoitaa ohjelmistorobotiikalla. Hokhamin ym. (2019, 12) valottavat, että ohjelmistorobotit eivät lepää, eivät tee virheitä eivätkä aiheuta yhtä paljon kustannuksia kuin ihmistyöntekijä. Forrester (2014, 2) jatkaa, että ohjelmistorobotiikalla säästettävän ajan osalta ihmistyöntekijää voidaan kehittää ja siirtää haastavampiin

tehtäviin. Myös työntekijöiden tyytyväisyys lisääntyy, kun toistuva rutiinityö siirretään ohjelmistoroboteille (Lacity ym. 2016, 27).

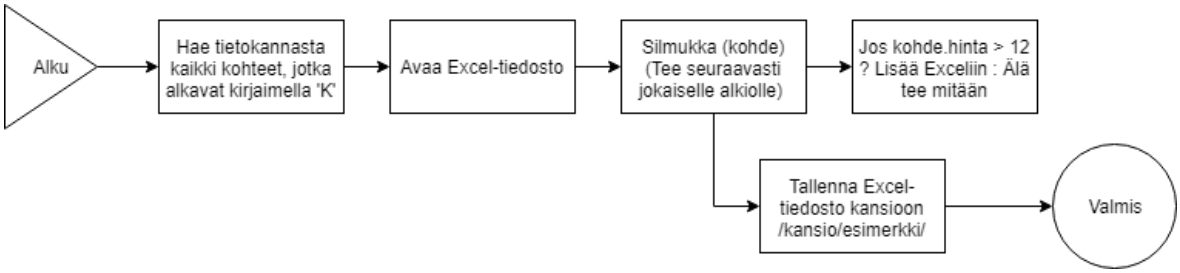
Ohjelmistorobotteja ei ohjelmoida itse, vaan niiden konfiguroimiseen käytetään ohjelmistoa, eli ohjelmoinnin sijasta voitaisiin puhua robottityöntekijöiden kouluttamisesta. Ohjelmistorobottien kehittäminen ei vaadi ohjelmointiosaamista. Ohjelmistorobotiikkatuotteiden käyttöliittymät muistuttavat vuokaaviota, jossa yksi toimenpide muistuttaa palapelin osaa. Osat liitetään toisiinsa muodostamaan toimenpiteiden ketjuja eli prosessia. Kehittäminen on nopeaa, sillä uuden ohjelmakoodin kirjoittamisen sijasta robotteja koulutetaan toimimaan eri järjestelmissä. Tämän takia ohjelmistorobotiikkaa hyödyntävissä organisaatioissa on huomattu, että robottien kehittäminen kannattaa siirtää liiketoimintaa harjoittaviin osastoihin, joissa substanssiosaaminen on vahvempaa. Tällöin ohjelmistorobotiikka tarjoaa ketterän työkalun prosessien tehostamiseen IT-hankkeiden ollessa perinteisesti hitaita sekä kalliita. (Craig ym. 2015a, 6–8.)

Ohjelmistorobotiikka ei kuitenkaan ole ensimmäinen automatisointiin käytetty teknologia, vaan olemassa on myös ohjelmistomakroja sekä komentokieliä. Ohjelmistomakro eli niin sanottu makro on ohjelma, joka suorittaa jonkin toistuvan tehtävän. Makro käynnistetään jonkun toimesta. Yksi makron ominaisuuksista on myös se, että se rajoittuu tiettyyn ohjelmaan. (BeeckerCo 2019.) Toinen ohjelmistorobotin sukulaisista on komentokieli. Komentokieli on yleisemmältä nimeltään skripti. Skriptiksi kutsutaan komennoista koostuvia sarjoja, jotka myös suorittavat jonkun tehtävän (UiPath 2014). Komentokielet eivät ole ohjelmointikieliä vaan komentokieliä ajetaan käyttöjärjestelmissä. Suurin ero ohjelmistorobottien ja aikaisempien sukulaisten välillä on se, että ohjelmistorobotit toimivat kaikissa sovelluskerroksissa. Vahvan integraation lisäksi ohjelmistorobotit tarjoavat helpon käyttöliittymän sekä monipuolisen valikoiman erilaisia toimenpiteitä tehtävien suorittamiseksi.

### 2.1.2 RPA ja SPA

Haikonen (2016) jakaa ohjelmistorobotiikan kahteen kategoriaan: RPA (Robotic Process Automation) eli ohjelmistorobotiikkaan ja SPA (Smart Process Automation) eli älykkääseen ohjelmistorobotiikkaan. Northamin (2017) mukaan RPA ratkaisut tarkoittavat sääntöpohjaista ohjelmaa, jonka työskentely tapahtuu vaihe vaiheelta. Kuvan 2 vuokaaviossa on esitetty ohjelmistorobotin työskentely, jossa jokainen laatikko kuvaa

yhden askeleen kohdalla tapahtuvaa toimenpidettä.




Kuva 2. Ohjelmistorobotin työskentely

Haikonen (2016) jatkaa, että RPA-ohjelma ei osaa varautua muutoksiin tai poikkeuksiin: se osaa toimia vain tarkalleen sille annettujen sääntöjen perusteella. Craigin ym. (2015b, 13) tukevat väitettä kertomalla, että roboteille täytyy antaa tarkemmat ohjeet kuin ihmiselle. Ihminen esimerkiksi ymmärtää että HKI ja HELSINKI on sama asia, mutta ohjelma ei. Toisin sanoen, RPA-roboteilta puuttuu päättelykyky. Tämän takia RPA soveltuu parhaiten prosesseihin, jossa on tarkkaan määritellyt säännöt sekä paljon toistoa. RPA-ratkaisusta oiva esimerkki olisi seuraava prosessi: Excel-taulukosta haettavia tietoja syötetään toiminnanohjausjärjestelmään. Toiminnanohjausjärjestelmästä saadaan edelleen lisätietoja. Uudet tiedot viedään kolmanteen järjestelmään ja sieltä saatavien tietojen perusteella muodostetaan uusi koontitaulukko kaikista prosessin tiedoista.

SPA osaa taas toimia edellistä monimutkaisemmassakin prosessissa. Samalla tavalla kuin RPA-, myös SPA-ohjelmille annetaan ohjeet toimia. Erona on se, että SPA-ohjelmalla on kyky oppia paremmaksi käyttämällä hyödyksi koneoppimista. Järvenpää (2018) kiteyttää koneoppimisen prosessiin, jossa ohjelma suoriutuu tehtävästä ja lopputuloksestaan saaman palautteen perusteella säätää toimintaansa eli oppii. Tämän takia SPA soveltuu parhaiten prosessiin, jossa käsiteltävä tieto on vapaamuotoisempaa ja sisällön perusteella tehdään arvio seuraavasta toimenpiteestä. SPA ohjelma voi rakentaa itselleen prosessin vaatiman päättelylogiikan ja ottaa lopulta työn vastuulleen itse. Chatbot eli ihmisen kanssa keskustelemiseen suunniteltu ohjelma olisi oiva esimerkki SPA-ratkaisusta, jos chatbot sisältäisi mahdollisuuden ohjelmalle analysoida aikaisempia keskusteluja, jotta se voisi kehittyä paremmaksi.

SPA vaikuttaa siis ylivoimaiselta vaihtoehdolta RPA:han nähden. Totuus kuitenkin on, että molemmilla ohjelmistorobotiikan tyypeillä on omat käyttötarkoituksensa. RPA-ohjelma ei pystyisi tunnistamaan kirjaimia kuvasta eikä antamaan tarkkaa arviota siitä, tuleeko tilaus myöhässä perille vai ei. Toisaalta SPA-ohjelma voisi käsitellä tietynlaisen saapuvan tilauksen halutulla tavalla, kunnes joku määrämuotoinen käsittelysääntö muuttuisikin.

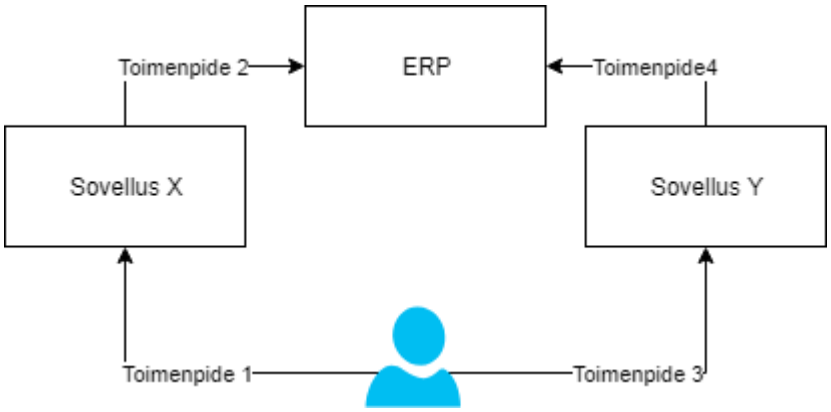
Ohjelman määrittäessä säännöt itse ihmisen olisi vaikea käydä vaihtamassa vain parametria, jotta ohjelma antaisi oikeanlaisen lopputuloksen.

### 2.1.3 Robottien konfigurointi

Ohjelmistorobotteja on helppo konfiguroida, mutta huonojen ratkaisuiden tekeminen on yhtä helppoa. CFB Bots (2018) teroittaa, että Robotin täytyy olla helposti luettava. Ohjelmistorobottien käyttöliittymä on usein kuvan 2 vuokaavion mukainen. Robotit koostuvat palasista, joiden sisällä suoritetaan jokin toimenpide. Nämä palaset voidaan nimetä ohjelmoijan mielen mukaisesti, mutta kannattaa käyttää selkeää nimeämistapaa. Kun robotin askeleet on nimetty selkeästi, ei ohjelmoijan tarvitse tarkastaa jokaista palaa erikseen eikä ajaa robottia nähdäkseen sen toiminnan. ”Kirjaudu sähköpostiin” on esimerkki hyvin nimetystä askeleesta. Hyvin nimetty robotti nopeuttaa virhetilanteiden selvittämistä ja kehitystyötä.

Automatisoitavan prosessin heikkoudet tulevat esiin RPA-ratkaisua kehittäessä. Ohjelmistorobotit toimivat parhaiten prosesseissa, jotka on suunniteltu uudestaan kokonaan tai osittain. (Deloitte 2017, 14.) Samalla kun luodaan RPA-ratkaisua, on mahdollista tunnistaa olemassa olevan prosessin heikkoudet ja muuttaa samalla toimintatapaa. Silloin voidaan myös päättää, automatisoidaanko koko prosessi vai vain osia siitä.

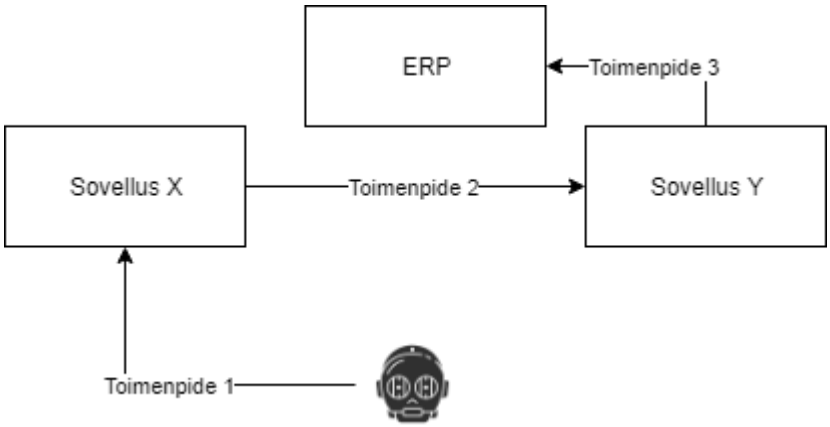
CFB Botsin (2018) mukaan on suositeltavaa, ettei robotin konfiguroimista aloiteta ennen suunnitelmaa. Tämä sisältää prosessien sisältämien komponenttien ja niiden välisten yhteyksien tunnistamisen.




Kuva 3. Ihmisen toiminta prosessissa

Robotti kannattaa suunnitella niin, että se tekee toimenpiteitä loogisessa järjestyksessä. Monesti ohjelmistorobotti mallintaa ihmisen tekemää työtä. Robottia konfiguroidessa kannattaa pysähtyä miettimään, onko ihmisen tapa tehdä kaikista loogisin ja nopein.

Voidaanko kenties joku välivaihe jättää pois kokonaan? Yleensä robotilla tehtävät toimenpiteet saadaan virtaviivaistettua. Esimerkiksi ihminen saattaisi tehtävää tehdessään käydä Kuvan 3 kaavion mukaisesti sovelluksessa X ja seuraavaksi veisi tiedot ERP-järjestelmään. Sitten ihminen jatkaisi sovellukseen Y ja takaisin ERP-järjestelmään.



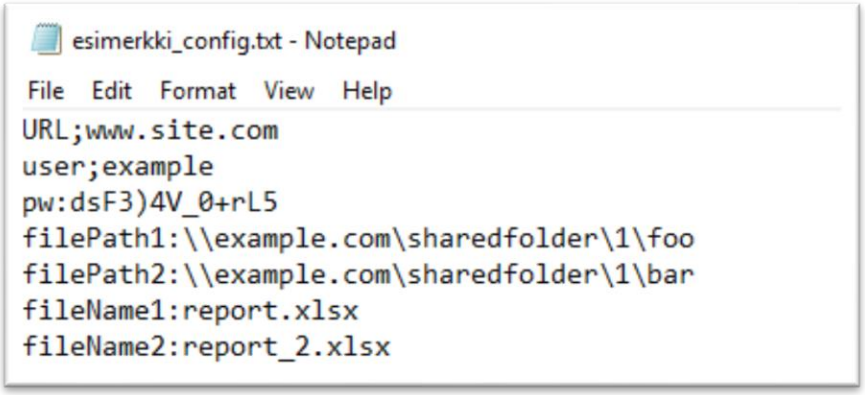

Kuva 4. Robotin toiminta prosessissa

Kuvan 4 kaavion mukaisesti robotti voisi ensin kerätä tiedot sovelluksesta X ja sovelluksesta Y ja mennä aivan viimeiseksi ERP-järjestelmään. Näin säästettäisiin yksi toimenpide kokonaan.

Modulaarisuus on erittäin tärkeä apuväline robottien suunnittelemisessa. Tämä tarkoittaa robottien pilkkominen osiin. Modulaarinen rakenne helpottaa monimutkaisten prosessien kehittämistä, sillä robotit voidaan jakaa useille kehittäjille. (CFB Bots 2018.) Kokemuksieni perusteella hyvällä robotilla on vain yksi tehtävä. Robotin kaatuessa virheen syyn löytäminen on helpompaa, mutta ennen kaikkea helpottuu virheen uusintaminen virheenkäsittelyn lisäämiseksi.

Robotissa täytyy olla virheenkäsittely. Robottiohjelmoijan täytyy miettiä, mitä tapahtuu, kun robotti epäonnistuu suorituksessa. Esimerkiksi robotin kutsuma Web-palvelu ei vastaa. Jos Web-palvelusta saatava tieto on oleellinen robotin toiminnan kannalta, on syytä käsitellä tilanne, jossa tietoa ei saadakaan käytettäväksi. Yksi tapa suoritua esimerkin mukaisesta tilanteesta olisi kirjoittaa logiviesti ja keskeyttää ajaminen tämän jälkeen.

Logiviestejä kannattaa aina sisällyttää ohjelmistorobottiin. Viestit kuten ”Robotti X aloitti toimintansa”, ”Tehdään tietokantahaku X” ja ”Tietoja ei saatavilla”, ovat kaikki hyviä esimerkkejä logiviesteistä, jotka kertovat, mitä ajon aikana on tapahtunut. Ohjelmistorobottien toimiessa palvelimella logiviesti on käytännössä ainoa apuväline virheiden selvittämiseksi.




Kuva 5. Esimerkki konfiguraatiotiedostosta

Kovakoodaamista, eli arvojen kirjoittamista ohjelman sisälle kannattaa välttää. Kovakoodaamista käytettäessä voi käydä niin, että verkkosivun osoite jossa robotti käy, vaihtuu. Silloin täytyy palata robotin konfiguraatioon ja etsiä kaikki askeleet, joissa on määritetty verkkosivun arvo. Jos arvo on useassa askeleessa, voi käydä niin, ettei kehittäjä löydä kaikkia, mikä johtaa virheisiin. Toisaalta robotin konfiguroimiseen ei voi aina palata, esimerkiksi siinä tapauksessa, kun robotti on julkaistuna tuotantoympäristöön. Suositeltavaa olisi käyttää konfiguraatiotiedostoja, jossa määriteltäisiin robotin käyttämiä arvoja. (CFB Bots). Kuvassa 5 on esimerkki konfiguraatiotiedostosta. Rivillä on yksi arvo, jota robotti käyttää ajon aikana. Ensimmäisenä on arvon nimi ja seuraavana on sen arvo. Erotinmerkkinä on :-merkki. Toisin sanoen ensimmäinen arvo on nimeltään URL ja sen arvo on www.site.com. Näin verkkosivun osoitteen muuttuessa ei tarvitsisi kuin vaihtaa konfiguraatiotiedostossa arvo oikeaksi.

Testaaminen on erittäin tärkeää RPA-ohjelmoinnissa. Monesti robotin saama tieto voi tulla eri muodossa kuin oletetaan, ja tämä saattaa aiheuttaa ongelmia. Kunnollinen testaus osoittaa robotin ongelmat, ja ongelmien tutkiminen johtaa parempaan ymmärrykseen robotista. Ohjelmistorobottien nopean ja helpon konfiguroimisen vuoksi testaamisen tärkeys korostuu: kehittäjä voi sokaistua kehittämisen nopeuden vuoksi. Toisaalta robottiohjelmoija voi olla myös aloittelija, jolla ei ole vankkaa ohjelmointirutiinia.

## 2.2 DSV:llä olevat teknologiat

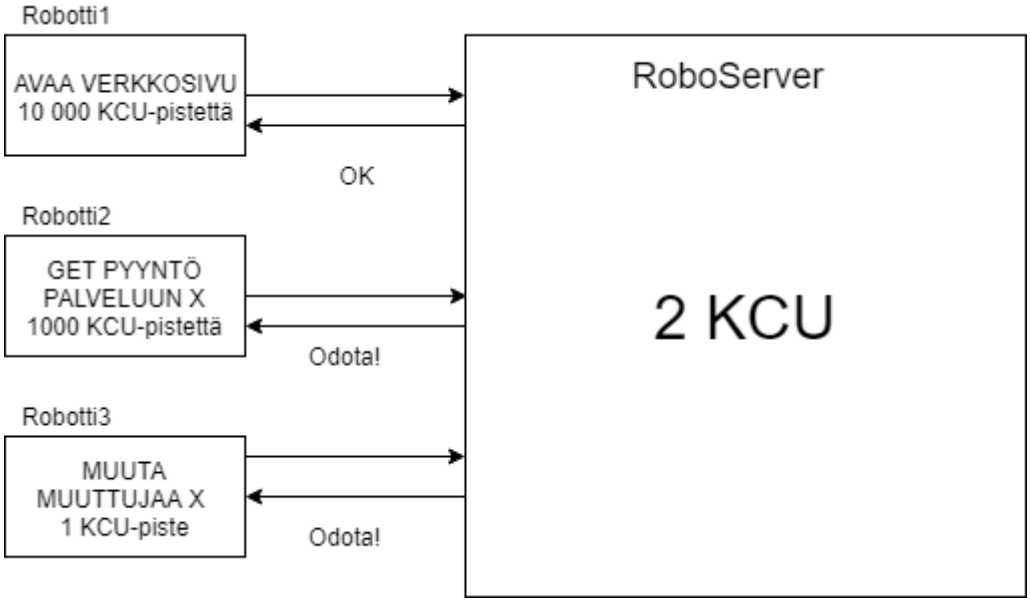
### 2.2.1 Kofax Kapow

Kofax on prosessiautomaatioon erikoistunut ohjelmistoalan yritys, jolla on 1600 työntekijää yli 35 maassa (Kofax 2018a). Kofax Kapow on DSV:n käytössä oleva RPA -

tuote, jolla voidaan luoda ohjelmistorobotteja nopeasti graafisella Design Studio -käyttöliittymällä. Robotit koostuvat vaihe vaiheelta suoritettavista askelista, jotka määritetään käyttöliittymässä. Askelien sisälle voidaan kirjoittaa monimutkaisiakin JavaScript -ohjelmia tiedonkäsittelyä varten, mutta periaatteessa automaation yksinkertaiseen prosessiin voi luoda vaatimattomallakin tietoteknisellä taidolla.

Kapow -robotit julkaistaan erityiselle RoboServer palvelimelle, jossa ne voidaan ajastaa toimimaan tietyin väliajoin tai käynnistää esimerkiksi http-pyynnöllä. Kapow Katalyst RoboServer -palvemella olevia robotteja pääsee myös hallinnoimaan Web-käytöliittymästä. Kapow on erityisen etevä relaatiotietokantojen kanssa ja siinä on yhteensopivuusajurit useimpiin tietokantapalvelimiin kuten MySQL:ään, Microsoft SQL Serveriin, IBM DB:hen ja Oracleen. Kapow:n heikkous on sen huono toimivuus työpöytäsovelluksien ja dynaamisten JavaScriptiä -sisältävien Web-sovellusten kanssa.

Jotkut Kapow-askeleet ovat kalliimpia kuin toiset. Rocziek (2018) kertoi, että yhtä Kapowin suoritusyksikköä määrää *KCU* (Kapow Compute Unit). Tämä kertoo kuinka monta askelta (tai operaatiota) Kapow Katalyst RoboServer voi suorittaa yhdessä sekunnissa. Yksi KCU palvelimella vastaa 5000 *KCU-pistettä* sekunnissa. (Huomaa, että KCU on eri asia kuin KCU-piste.)



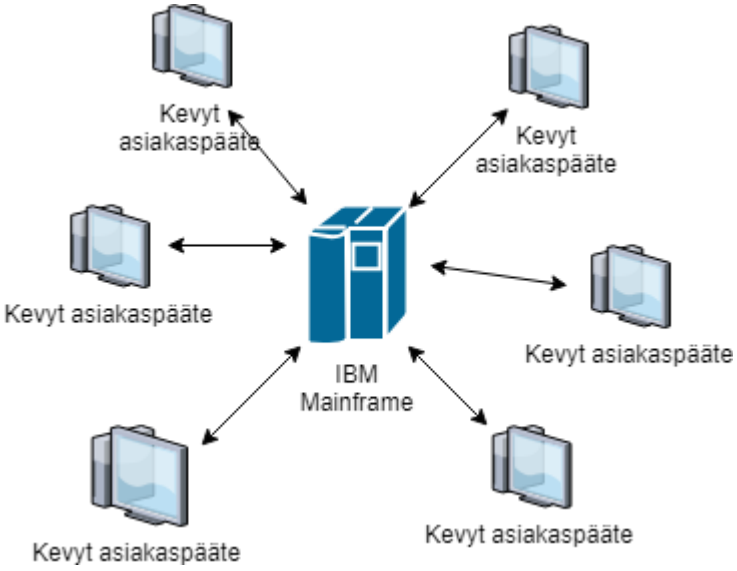

Kuva 6. KCU toimintaperiaate

Tämä tarkoittaa sitä, että jos palvelimella on 5 KCU:ta, niin silloin yhden sekunnin aikana voidaan suorittaa 25 000 KCU-pisteen edestä askelia. Jos palvelimella olevan kapasiteetin rajat ylittyvät, joutuu robotti odottamaan, kunnes KCU:ta vapautuu robotin käyttöön. Sekunnin aikana 5000 KCU-pistettä palautuu käytettäväksi. Askeleet vievät eri suuruisen määrän KCU-pisteitä, ja ohjelmistorobottien nopeuttamiseksi tietynlaiset

ratkaisut toimivat paremmin kuin toiset. Esimerkiksi Web-sivun avaaminen vie 10 000 KCU -pistettä, kun taas Web-palveluun tehtävä REST -pyyntö vie 1000 KCU-pistettä. Kofax (2018b) vahvistaa, että 4:llä KCU:lla voidaan kutsua 20 kertaa REST -rajapintaa ja toisaalta tehdä 20000 arvon muuntamista sekunnissa. Kuvan 6 kaaviossa palvelimella on 2 KCU:ta käytössään, joten Robotti 1:n toimenpide Web-sivun avaaminen vie koko palvelimen kapasiteetin ja Robotti 2 sekä Robotti 3 joutuvat odottamaan KCU-pisteiden vapautumista sekunnin ajan ennen kun voivat toimia. Tarkan KCU-pisteiden kulutuksen eri askeltyyppiä kohden löytää Design Studio -käyttöliittymästä.

### 2.2.2 Cargolink

Globaalin DSV Roadin kuljetusten hallintajärjestelmä Cargolink pohjaa teknologisesti IBM:n Mainframe -järjestelmään. IBM Mainframet ovat suuria tietojärjestelmiä, joita IBM on kehittänyt vuodesta 1952. Mainframe kuvaa yhtä tietokonetta, johon muut tietokoneet ovat yhteydessä jotta, ne voivat jakaa Mainframen toiminnallisuuksia. (IBM 2019).




Kuva 7. IBM Mainframe kuvaus

Kuvassa 7 keskellä on IBM Mainframe johon sitä ympäröivät kevyet asiakaspäätteet ottavat yhteyden jakaakseen keskustietokoneen toiminnallisuuksia.

Sama pätee myös Cargolinkissa, jossa yhteys otetaan Mainframeen, joka muodostaa istunnon. Operointi järjestelmässä tapahtuu tämän istunnon kautta. Cargolinkia voidaan kutsua myös samalla niin sanotuksi *Legacy*-järjestelmäksi. Bisbalin (1999, 103) mukaan legacy-järjestelmäksi voidaan kutsua mitä tahansa tietojärjestelmää, jonka kehitystä ei tapahdu ollenkaan tai se tapahtuu hyvin hitaasti. Legacy-järjestelmät ovat usein organisaatiolleen taakka, koska

–järjestelmät pyörivät yleensä vanhalla laitteistolla, jota voi olla hidasta tai kallista ylläpitää
–järjestelmän ylläpito voi olla kallista, koska dokumentaatio ja ymmärrys järjestelmästä on usein vajavaista
–integraatiot toisiin järjestelmiin voivat olla hankalia selkeiden käyttöliittymien puuttumisen vuoksi
–järjestelmää on hankala ellei mahdoton kasvattaa vastaamaan organisaation tarpeita liiketoiminnan muutoksissa.

Vuodesta 2016 eteenpäin Cargolinkin tukea on jatkettu IBM:n kanssa aina vuosi kerrallaan. Ylläpito on kallista ja vanha järjestelmä ei vastaa DSV:n nykyisiin ja alati muuttuviin liiketoimintatarpeisiin. Tämän takia ei olekaan yllättävää, että Cargolink on päätetty korvata. DSV:n mukaan Cargolink Way Forward (myöhemmin CWF) on projekti, jonka tuloksena odotetaan uutta kuljetusten hallintajärjestelmää Cargolinkin tilalle.

CWF rakennetaan moduulipohjaisesti, ja sen kehitys kestää useamman vuoden ajan. Makin (2017) mukaan moduulipohjainen ohjelmointi tarkoittaa sitä, että ohjelman eri toiminnallisuudet koostuvat erillisistä moduuleista, jotka integroidaan rajapinnan kanssa toisiinsa. Etuina on se, että kehittäjät voivat kehittää moduuleita samanaikaisesti erillään ja jokainen moduuli on helppo julkaista erikseen.

CWF:ää on pilotoitu vuonna 2018 kahdessa maassa: Liettuassa ja Itävallassa. Aikataulua uuden järjestelmän jalkauttamiseksi muihin maihin odotetaan DSV:llä kiinnostuneina.

### 2.2.3 Organisaatioiden välinen tiedonsiirto

Organisaatioiden välinen tiedonsiirto, eli OVT on vanha teknologia, jota käytetään organisaatioiden väliseen viestintään. OVT:n juuret ulottuvat 60-luvulle kun ensimmäisen kerran yhteyden ottaminen tietojärjestelmästä toiseen tuli mahdolliseksi. Kotimaisten kielten keskuksen (2019) sivuilla Englanninkielinen vastine EDI (Electronic Data Interchange) on käytössä OVT:n sijasta sekä EDI on kokemukseni mukaan myös vahvasti alan jargonissa, joten käytän sitä opinnäytetyössä jatkossa.

Ensimmäiset EDI viestit lähetettiin vuonna 1965 Hollantilais-Amerikkalaisen höyrylaivan toimesta. Viesti lähetettiin trans-Atlantisia laivausmanifesteja käyttäen telex-viestejä, joiden avulla pystyttiin lähettämään sivun verran tietoa noin kahden minuutin aikana. Nämä viestit muunnettiin kasetiksi, jotka pystyttiin lopulta lataamaan tietokoneeseen.

1968 joukon Yhdysvaltalaisen rautatieyritysten toimesta perustettiin Transportation Data Coordinating Comittee (TDCC) kehittämään standardia EDI -formaattia, sillä laivaus-, rautatie-, lento-, ja kuljetusyritykset välittivät toistensa kanssa sähköisiä viestejä. 1975 mennessä File Transfer Proctocol (FTP) oli keksitty ja ensimmäiset EDI standardit julkaistiin.1981 ANSIX12 -standardit julkaistiin sisältäen elintarvike-, logistiikka- ja pankkialan. 1985 Yhdistyneet kansakunnat kehittivät EDIFACT EDI standardin edesauttamaan kansainvälistä teknologista kehitystä. Data Communications Solutions (2016) kertoo, että nämä kaksi standardia ovat suurimmat EDI-viestinnässä. ANSIX12 on enemmän käytössä Yhdysvalloissa, kun taas EDIFACT on suositumpi Yhdysvaltojen ulkopuolella (McCarthy, 2013).

```
UNB+UNOA:1+010100002530O1+O0013000093SCHA-Z59+991006:1902+PAYO0012101221'
UNH+1+INVOIC:D:97A:UN'
BGM+381+1060113800026+9'
DTM+137:199910060000:102'
NAD+BT+VAUXHALL MOTORS LTD::91'
RFF+VA:382324067'
NAD+SU+2002993::92'
RFF+VA:123844750'
CUX+2:EUR'
PAT+1'
DTM+140:19991031:102'
LIN+++090346642:IN'
QTY+12:54:PCE'
MOA+203:1960.29'
PRI+AAA:3630.1724::NTP:100:C62'
RFF+SI:165480'
DTM+11:199909280000:102'
RFF+ON:X18V00003'
RFF+TN:AB1'
TAX+7+VAT+++:::0'
NAD+ST+023::92'
UNS+S'
MOA+77:1960.29'
TAX+7+VAT'
UNT+24+1'
UNZ+1+PAYO0012101221'
```

Kuva 8. EDIFACT INVOIC D97A

Kuvan 8 EDI sanoma on EDIFACT standardin mukainen esimerkki laskulle.

Sanoma koostuu jaetaan ’-merkillä jaetuista segmenteistä. Segmenttien sisällä tiedot jaetaan elementteihin ja koostetietoihin. Elementit jaetaan toisistaan +-merkillä ja koostetiedot :-merkillä. Kuvan 8 segmentit on rivitetty luettavuuden vuoksi, sillä oikea EDI sanoma on yhdellä rivillä. Rivillä 1-2 on UNB segmentti, joka on sanoman ylätunniste. UNB sisältää yleistä tietoa lähetyksestä ja yhdessä sanoman lopussa olevan UNZ segmentin kanssa sulkee viestin. Kuvan 8 sanoman rivillä 3 on UNH segmentti, jossa kerrotaan aina viestin tyyppi. Tässä tapauksessa viesti on INVOIC:D:97A:UN ja tarkoittaa, että viesti on EDIFACT INVOIC D97A standardin mukainen laskusanoma. UNH sulkee viestin sisällön UNT segmentin kanssa. Yhdessä sanomassa voi olla useita erilaisia UNH/UNT segmenttejä, eli monenlaisia viestejä. Rivillä 5 on taas DTM eli

Date/Time/Period, vapaasti suomennettuna päiväämäärä/aika-segmentti. Segmentin ensimmäinen elementti kertoo mitä DTM aikaa segmentti koskee. Tässä tapauksessa 137 kertoo, että on kyse sanoman luomisajasta. Seuraavana elementtinä on itse aika ja viimeisenä on ajan formaatti. Esimerkin 102 on formaattina CCYYMMDD.

EDI on ollut tulostaan asti avainasemassa organisaatioiden välisessä viestinnässä ja tänään jopa 85% kaikesta sähköisestä sanomaviestinnästä tapahtuu EDI -teknologialla (Edicom 2019.) Toisaalta Huemer (2000, 1) kertoo, että EDI:ä hyödyntävien organisaatioiden määrä on suhteellisen pieni verrattuna organisaatioiden kokonaismäärään. EDI:n pystyttämiseen ja käyttämiseen vaadittava hinta on liian suuri pienille- ja keskisuurille yrityksille. Huemerin (2000, 3) mukaan toista sanomamuotoa, XML-teknologiaa hyödyntävät sovellukset tekivät tuloaan 2000 luvun taitteessa ja siitä povattiin EDI:n korvaajaa. XML on EDI:ä joustavampi ja luettavampi sanomamuoto. XML:n etuina on myös sen kyky siirtää myös muuta tietoa, kuten kuvia. EDI hyötyy kuitenkin yhtenäisistä standardeista, jota XML:llä ei ole. (Huemer 2000, 3–4).

### 2.2.4 Visual Basic 6

Visual Basic on Microsoftin kehittämä ohjelmointikieli, jolla voi tehdä graafisia käyttöliittymiä Windows-käyttöjärjestelmiin. Ensimmäinen versio, Visual Basic 1.0 julkaistiin 1991. Visual Basic 6, jota kutsun tästä eteenpäin VB6:ksi, julkaistiin 1998. (Cleverism 2018.) Ispliter (2014) kertoo, että 2002 julkaistiin Visual Basic .NET, joka on kielen nykyinen tuettu versio. Microsoft lopetti VB6:n ohjelmointiympäristöjen tukemisen vuonna 2008, mutta yritys on jatkanut tukea VB6 -applikaatioille heidän käyttöjärjestelmissään vuoteen 2025 asti. (Microsoft 2019)

Cargolinkiin voidaan kehittää ohjelmistomakroja VB6-kielellä. DSV:llä on paljon VB6 ohjelmistomakroja tehostamassa työntekijöiden toimintaa. Desmond (2012) sanoo, että tämä ei ole poikkeuksellista, sillä VB6 sovelluksia on olemassa monissa legacy-järjestelmissä. Sekä makrot että ohjelmistorobotit sopivat tehostamaan hyvin legacy-järjestelmiä, sillä niiden kehittämiseksi ei tarvitse kajota järjestelmän ohjelmakoodiin.

Anderson (2017) muistuttaa myös, että Microsoft Officen tuotteissa voi kirjoittaa Visual Basic -ohjelmistomakroja Visual Basic for Applications -kielellä, joka nimestään huolimatta on melkein sama kuin VB6.

### 2.2.5 Kevyt asiakaspääte (Thin Client)

Kevyt asiakaspääte eli Thin Client on tietokonepääte, jossa ei ole tuuletinta eikä kovalevyä. Thin Client -arkkitehtuuri toimii niin, että kaikki kevyet päätteet ottavat yhteyttä yhteen keskusyksikköön, jonne kaikki sovellukset ja tiedot on tallennettu. Keskusyksikkö jakaa päätteille virtuaalisen työpöydän. Näin päätteet pääsevät käsiksi sovelluksiin ja tiedostoihin. (Devonit 2019.) Kevyitä asiakaspäätteitä perustellaan seuraavista syistä

–Omiin tietoihin voi päästä käsiksi miltä vaan päätteeltä tai tietokoneelta.
–Tiedostoista ei tarvitse ottaa varmuuskopioita, sillä kaikki tieto on keskusyksikössä.
–Kevyt asiakaspääte vähentää sähkönkulutusta.
–Laitteistoa ei tarvitse päivittää niin usein.
–Tietoturvan taso on parempi kuin tavallista tietokonetta käytettäessä.
–Standardoidussa ympäristössä tukea saa nopeasti.

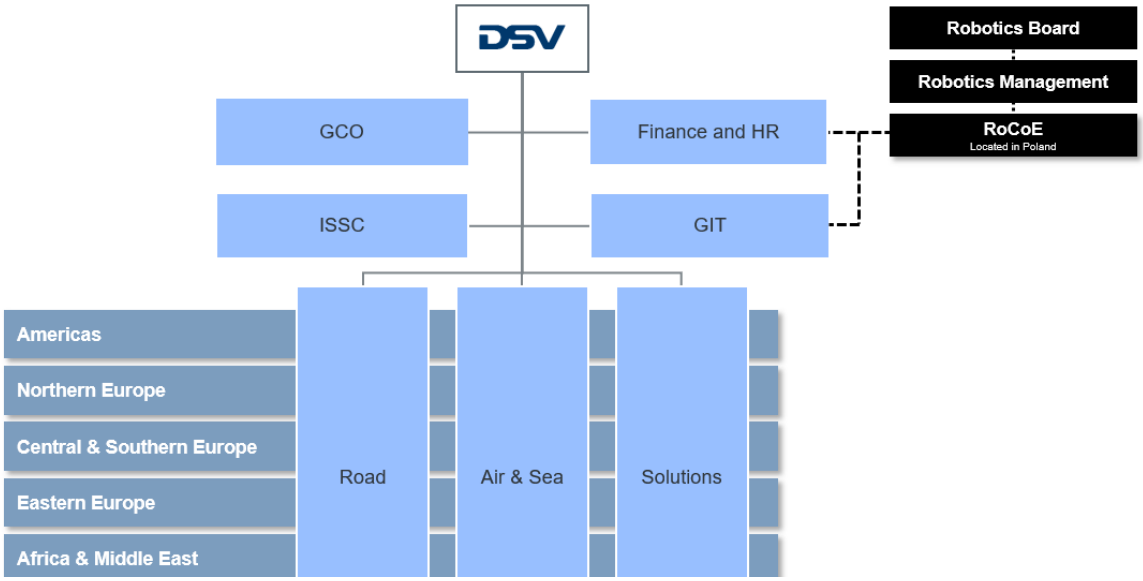
Kevyillä asiakaspäätteille ei voi asentaa eikä sieltä voi ajaa luvattomia ohjelmia. Tietokonetta ei tarvitse vaihtaa usein, eikä vaihto ole kallista. Myös varkaudet vähenevät, sillä kevyellä asiakaspäätteellä ulkopuolinen ei tee mitään. Sähkönkulutuskin laskee, joka alkaa näkyä suurissa määrissä kevyitä asiakaspäätteitä. Kun kaikki tiedot tallentuvat keskusyksikköön, niin varmuuskopiointi täytyy suorittaa vain yhdessä paikassa. Näiden syiden vuoksi on järkevää käyttää kevyitä asiakaspäätteitä silloin, kun kyseessä on suuri yritys, jolla on monia työntekijöitä ja keskitetty tuki. (Forcepoint, 2019.)

DSV:llä tiedostot sijaitsevat verkkolevyillä, jossa on käytännössä kaikki DSV:n tiedostot. Levyille tarjotaan käyttäjäkohtaisia oikeuksia ja käyttäjä voi tarpeesta riippuen pyytää pääsyä eri levyille. DSV:llä suurimmalla osasta työntekijöitä on käytössään kevyt asiakaspääte. Mikäli työ ei vaadi erikoisohjelmien käyttämistä tai matkustamista, niin työntekijälle annetaan työvälineeksi kevyt asiakaspääte.

# 3 Automatisaatio tilausten käsittelyyn

## 3.1 Kohdeyritys

DSV on kansainvälinen kuljetus- ja logistiikkatoimija. DSV:llä työskentelee 60 000 työntekijää yli 90 maassa, ja se on viidenneksi suurin alan toimija. DSV on jaettu kolmeen divisioonaan Air & Sea (lento- ja merikuljetukset), Road (maantiekuljetukset) ja Solutions (varastointiratkaisut), mikä näkyy kuvassa 9. DSV toimii globaalin huolitsijan roolissa. (DSV 2019.) Suomen Huolinta- ja logistiikkaliiton (2019) mukaan huolitsijan pääasiallinen tehtävä on toimitusketjun hallinta kokonaisuudessaan. Tavaran kuljettamisen lisäksi tarjotaan lisäpalveluita, joita voisivat olla tullaukseen ja verotukseen liittyvät palvelut tai viestintätekniikkaa hyödyntävät palvelut kuten toimituksen seuranta. Huolitsija ei yleensä omista omaa kuljetuskalustoa vaan liiketoimintaa harjoitetaan hyödyntämällä alihankkijoiden verkostoa. Suomen DSV:llä on kuitenkin myös omaa kalustoa, terminaaleja ja joissain muissa maissa työllistetään myös omia kuskeja.




Kuva 9. DSV:n organisaatiorakenne

Suomen DSV:n liikevaihto oli 241 miljoonaa vuonna 2017. Suomessa toimivat DSV:n divisioonat on jaettu omiin yrityksiin, jotka toimivat pääasiassa samassa toimipisteessä ja tekevät keskenään yhteistyötä. Suomen DSV:llä työskentelee yhteensä 280 työntekijää, joista 212 Roadin, 44 Air & Sea:n ja 24 Solutionsin palveluksessa. Sen lisäksi yhteistyössä on monia alihankkijoita ja kumppaneita.

Opinnäytetyö tehtiin DSV Road Oy:n, Business Change Management -osastolla (BCM), joka toimii jaettuna palveluna kaikille Suomen DSV:n yrityksille. Jaetut palvelut on konsepti, jossa liiketoimintayksikkö toimii koko organisaation käytössä ja kustannukset jaetaan esimerkiksi käytön mukaisesti. BCM vastaa operatiivisten ohjelmistojen

päivittäisestä tuesta, ylläpidosta, asiakaslähtöisen sähköisen liiketoiminnan edistämisestä ja kehityksestä yhteistyössä GIT:in kanssa sekä verkostoitumalla yrityksiin ja yhteisöihin. BCM:llä on myös vahva rooli sisäisten prosessien automatisoinnin kehittämisessä. BCM opastaa henkilöstöä, asiakkaita ja toimittajia hyödyntämään sähköisiä palveluja ja työvälineitä. Toiminta tukee DSV:n strategian toteutumista.

Kuvassa 9 näkyy myös GIT, joka on DSV:n Global IT. Global IT:ssä tehdään kaikki tietotekninen kehitystyö poissulkien ohjelmistorobotiikka. Ohjelmistorobottien ympäristöjä hoitaa GIT:in osasto Robotics Centre of Excellence (myöhemmin RoCoE), joka tarjoaa paikallisille kehittäjille ympäristöt, koulutusta, tukea sekä tarkastavat robottiin liittyvän dokumentaation liiketoimintaprosesseineen ja suorittavat robottien julkaisemisen esituotanto- ja tuotantoympäristöön. Forresterin (2014, 5–6) mukaan juuri näitä Centre of Excellence -osastoja kannattaa ottaa käyttöön organisaatioissa ohjelmistorobotiikan haasteiden ja kehittämisen kohtaamiseksi.

Opinnäytetyön kohteena oli Roadin asiakaspalveluosasto, joka nimensä mukaisesti hoitaa asiakaspalvelua ja tilausten vastaanottamista.

DSV:llä EDI-viestit ovat EDIFACT -standardia. Muita käytettyjä sanomamuotoja ovat XML .csv. Myös flat fileä eli tietuetiedostoa, josta puuttuu tunnistekentät, voidaan joissain tapauksissa käyttää. Toisaalta tietuetiedoston tietosisältö voi perustua sijaintiin, esimerkiksi tietyllä rivillä oleva merkkijono voisi kertoa asiakkaan viitteen. Sanomia lähetetään EDI -muuntimien kautta Cargolinkiin ja niitä voidaan vastaanottaa sähköpostitse, HTTP-pyynnöillä tai SFTP- ja FTP-siirroilla. EDI:ä käytetään DSV:llä kahteen suuntaan viestimisen välineenä yhteistyökumppanien/asiakkaiden ja eri järjestelmien sekä Cargolinkin välillä.

### 3.2 Ongelmat ja kehittämistehtävä

DSV Road Oy:n Asiakaspalveluosaston työajasta suuri osa meni tilausten käsittelyyn. Sähköinen tilaus voidaan tehdä suoran EDI-yhteyden, asiakkaan operaattorin tai DSV:n tarjoaman sovelluksen kautta. Sähköisistä tilauksista eroavat manuaaliset tilaukset, jotka tehdään sähköpostitse. Manuaaliset kuljetustilaukset ovat leikkimielisesti sanottuna olleet Suomen logistiikka-alan standardi, nimittäin vuonna 2012 vain 40% kuljetustilauksista luotiin sähköisesti Suomessa. Vastaava luku samaan aikaan oli 90–95% muissa Pohjoismaissa. Tähän ongelmaan vastattiin jo vuonna 2012 Tietoyhteiskunnan kehittämiskeskuksen, Tieken, SÄTKY-hankkeella (Sähköisten toimintamallien käytön lisääminen logistiikassa), johon DSV monien muiden toimijoiden ohella osallistui.

Tavoitteena oli kasvattaa sähköistä tiedonsiirtoa eri toimijoiden välillä muiden Pohjoismaiden tasolle luomalla yhteiset suositukset. (Tietoyhteiskunnan kehittämiskeskus, 2019). Silti yllättävän suuri osa kuljetustilauksista siirtyy vielä tänäkin päivänä sähköpostilla. Pekka Aaltosen (2017) mukaan logistiikka-alan suurien toimijoiden kuljetustilauksista 90% saadaan sähköisesti. Samaan aikaan pienten ja keskisuurten sähköisyyden aste on vieläkin alle puolet. DSV Road Oy:n sähköisten kuljetustilausten aste on tällä hetkellä 75%, eli vaikka kehitystä on tapahtunut, niin kirittävää vielä löytyy. Samalla kun manuaalisesta kuljetustilauksesta siirrytään sähköiseen, niin avautuu uusi mahdollisuus: automatisointi.

Mitä tahansa sähköistä tilaustapaa käytetäänkään, niin tilaus päätyy Cargolinkiin jonkinasteisilla perustiedoilla. Tilaus kartoitetaan moninaisten asetusten perusteella tietylle liikenneosastolle ja niiden sisällä eri liikenteille. Asiakaspalvelun asiakkaiden osalta tietenkin asiakaspalvelun liikenneosastolle ja liikenteelle. Liikenneosastojen ja liikenteiden avulla tilauksia voidaan siirtää näkymään oikeille henkilöille Cargolinkissä. Liikenneosasto on ylempi kategoria, ja liikenne sen alakategoria. Tilaus käsiteltiin sitten kuljetusjärjestelmässä, jossa työntekijä haki käsittelemättömiä tilauksia ja alkoi purkamaan niitä järjestelmässä. Tilauksen käsittely on säännönmukaista, vaikka erilaisia sääntöjä ja käsittelytapoja onkin riippuen asiakkaan kanssa tehdystä sopimuksesta.

Eroavasta käsittelytavasta esimerkkinä voidaan pitää tilannetta, jossa asiakkaalle on sovittu erikoisrahditusehto: rahditustiedot poimitaan merkkijonona kuljetuksen hallintajärjestelmän vapaasyöttökentästä. Näin toimitaan, koska asiakkaan järjestelmästä ei pystytä lähettämään oikeanmuotoista sanomaa, jotta rahditus saataisiin kartoittumaan oikeaan kenttään Cargolinkiin. Toinen esimerkki eroavasta käsittelytavasta on tilanne, jossa asiakkaalla on oma toimitusaikataulu. Asiakkaan kanssa on sovittu oma standardista poikkeava aikataulu, jonka perusteella tilauksille lasketaan arvioitu toimituspäivämäärä.

Ongelma on, että suuri osa resursseista kului toistuvaan työhön. Tämä aiheutti kiireisenä ajankohtana stressiä että turhautumista. Osastolla ei voitu ottaa uusia kehityskohteita, koska valtava osa työntekijöiden kapasiteetista kului rutiinityössä. Kun työntekijällä oli haastava selvityspyyntö, hän stressaantui joutuessaan samaan aikaan huolehtimaan monien tilauksien peruskäsittelystä. Manuaalinen työ tuo myös riskejä, sillä työntekijä voi tehdä lyönti- ja huolimattomuusvirheitä käsitellessään tilauksia.

Ennen opinnäytetyöprojektia automatisointia ei ollut vielä rakennettu pääosin siitä syystä, että tilausten käsittelyä varjosti kirjava käsittelysääntöjen suma. Asiakkaiden sopimukset,

tilausprosessit ja kyky tuottaa tietoa eriävät toisistaan. Tilaustietojen ollessa erilaisia ja käsittelyn ollessa sidonnainen asiakkaan sopimukseen, oli vaikea tehdä yhdenlaista ohjelmistomakroa käsittelemään saapuvia tilauksia.

Tavoitteena oli luoda prosessi, jonka läpi kaikki saapuvat sähköiset vientitilaukset kulkisivat ja joka mahdollisimman hyvin poistaisi toistoa aiheuttavan työntekemisen osastolta. Tarkoituksena oli rakentaa erilaisten tietomatriisien avulla tilauksia käsittelevä RPA ohjelmistorobotti, jota voisi säätää vaihtamalla matriiseissa olevia arvoja, kun sopimus asiakkaan kanssa vaihtuisi. Tietomatriisilla tarkoitetaan riveistä ja sarakkeista muodostuvia tauluja, joiden avulla voidaan mallintaa sääntöjä ja arvoja robotin avuksi. DSV:llä käytössä oleva Kofax Kapow lyötiin lukkoon prosessin kulmakiveksi suunnitteluvaiheessa.

## 3.3 Annukka-projekti

### 3.3.1 Tausta

Opinnäytetyö sai alkunsa kesällä 2018, kun olin vielä työharjoittelujaksolla DSV:llä. Projektin ensimmäisen suunnittelupalaverin päätteeksi esimieheni kysyi minulta, haluaisinko tehdä aiheesta opinnäytetyön. Lähdin mielelläni toteuttamaan opinnäytetyötä, koska lopputuotteelle oli oikea tarve ja sain projektille suunnitteluvastuun. Työharjoittelujaksoni päättyi pian projektin käynnistämisen jälkeen. Jatkoin opinnäytetyön parissa palkkatyönä.

Opinnäytetyöprojekti sai suunnitteluvaiheessa työnimen Annukka, jota on käytetty alusta alkaen DSV:llä puheessa. Annukka sai nimensä siksi, että haluttiin luoda mielikuva tilauksia väsymättömästi käsittelevästä iloisesta asiakaspalvelijasta, joka olisi kuin yksi osaston työntekijä. Tosin varhaisessa vaiheessa kävi ilmi, ettei Annukka ole yksi ohjelmistorobotti, vaan monista ohjelmistoroboteista ja ohjelmistomakrosta koostuva kokonaisuus.

Auksi loin proof-of-concept (POC) –ratkaisun yhden asiakkaan vientitilauksille. Proof of concept on tuote, jolla testataan ideaa tai olettamusta. Tässä tapauksessa testattiin, onko ohjelmistorobotiikka tarpeeksi suorituskykyinen ratkaisu tilauksien käsittelyä varten. POC osoittautui toimivaksi ja oli lopullisen ratkaisun edeltäjänä mainio, sillä täysivaltaisen prosessin luominen tyhjästä olisi saattanut ajaa projektin liian syville vesille. POC oli oma kokonaisuus, joka käsittelee tällä hetkellä 300-400 tilausta viikossa. Tämä oli valmis lokakuun alussa ja sen jälkeen toteutettu opinnäytetyö hyödynsi POC:in arkkitehtuuria suunnittelussa.

### 3.3.2 Työskentelytavat

Robotteja kehittäessä tehdään pyyntö RoCoE:lle, jotta he avaavat projektin ja antavat kehittäjille oikeuden RoboServerin testiympäristöön. Kehitystyö tapahtuu paikallisesti. Kun halutaan siirtyä testiympäristöstä tuotantoon, toimitetaan RoCoE:lle suunnitteludokumentti (Solution Design Document). Dokumentissa kerrotaan projektin liiketoimintaprosessi, sovelluskuvaus sekä toimenpiteet, jotka pitää tehdä kun siirretään toiseen ympäristöön.

Toimenpiteitä voisi olla se, että roboteille pitää sallia pääsyt tietyille verkkolevyille tai tietokanta pitää pystyttää valmiiksi. RoCoEn työntekijä tarkastaa projektin teknisestä näkökulmasta – suorittaen koodin katselmoinnin. Toinen työntekijä tarkastaa projektin liiketoimintaprosessin näkökulmasta. Kun nämä on hyväksytty, siirretään projektin robotit esituotantoympäristöön ja alkaa ohjelmistotestaus, englanniksi User Acceptance Testing (UAT). Ohjelmistotestaus on lyhyt vaihe esituotantoympäristössä, jossa robotin kehittäjä tai projektin omistaja toteaaa, että robotit toimivat oikealla tavalla. Projektien julkaiseminen tuotantoon tapahtuu aina RoCoE:n toimesta keskiviikkoisin. Robottien kehittäjät ovat vastuussa robottien toiminnasta tuotantoympäristössä ja seurantaa suoritetaan seuraamalla robottiajojen logiviestejä.

Työtä opinnäytetyön eteen syksystä 2018 – alkuvuoteen 2019 tein keskiarvolla noin 50 tuntia kuukaudessa. Olin DSV:n toimistolla yhden tai kaksi kertaa viikossa ja tein tarvittaessa työtä etänä. Tein työtä yhdessä kollegani kanssa.

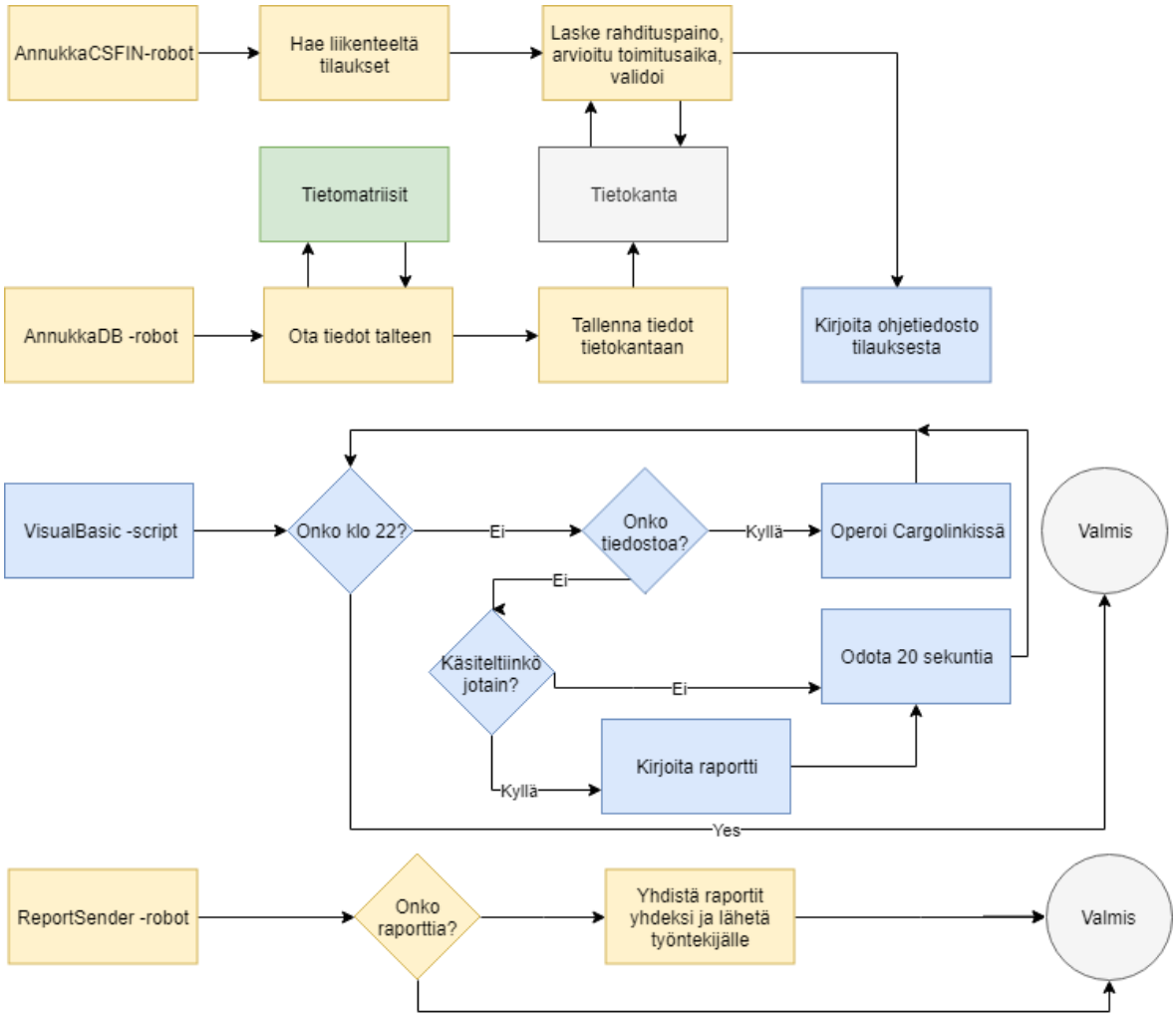
Työtavassa oli paljon ketterän kehittämisen piirteitä kuten viikottainen kokoontuminen. Kokoonnuimme projektin omistajan, asiakaspalvelupäällikön, sähköisiin tilauksiin erikoistuneen asiakaspalveluasiantuntijan, BCM-osaston päällikön sekä kollegani BCM-osaston kehittäjän kanssa viikoittain tunnin mittaiseen palaveriin, jossa kävimme läpi projektin tilannetta ja vaatimuksia. Kaikki vaatimukset eivät olleet selvillä työtä aloittaessa. Uusia vaatimuksia tuli vähitellen mikä hiljalleen osaltaan toi lisätyötä. Joskus vaatimuksia odotettiin ja toisinaan aiemmin toteutettu ratkaisu täytyi korvata uusien vaatimuksien tullessa. Toisaalta viikoittainen tapaaminen toi paljon etuja. Ongelmiin pystyttiin puuttumaan nopeasti ja kaikki olivat tietoisia siitä, missä tilanteessa projektin osalta mennään.

#### Suunnitelma

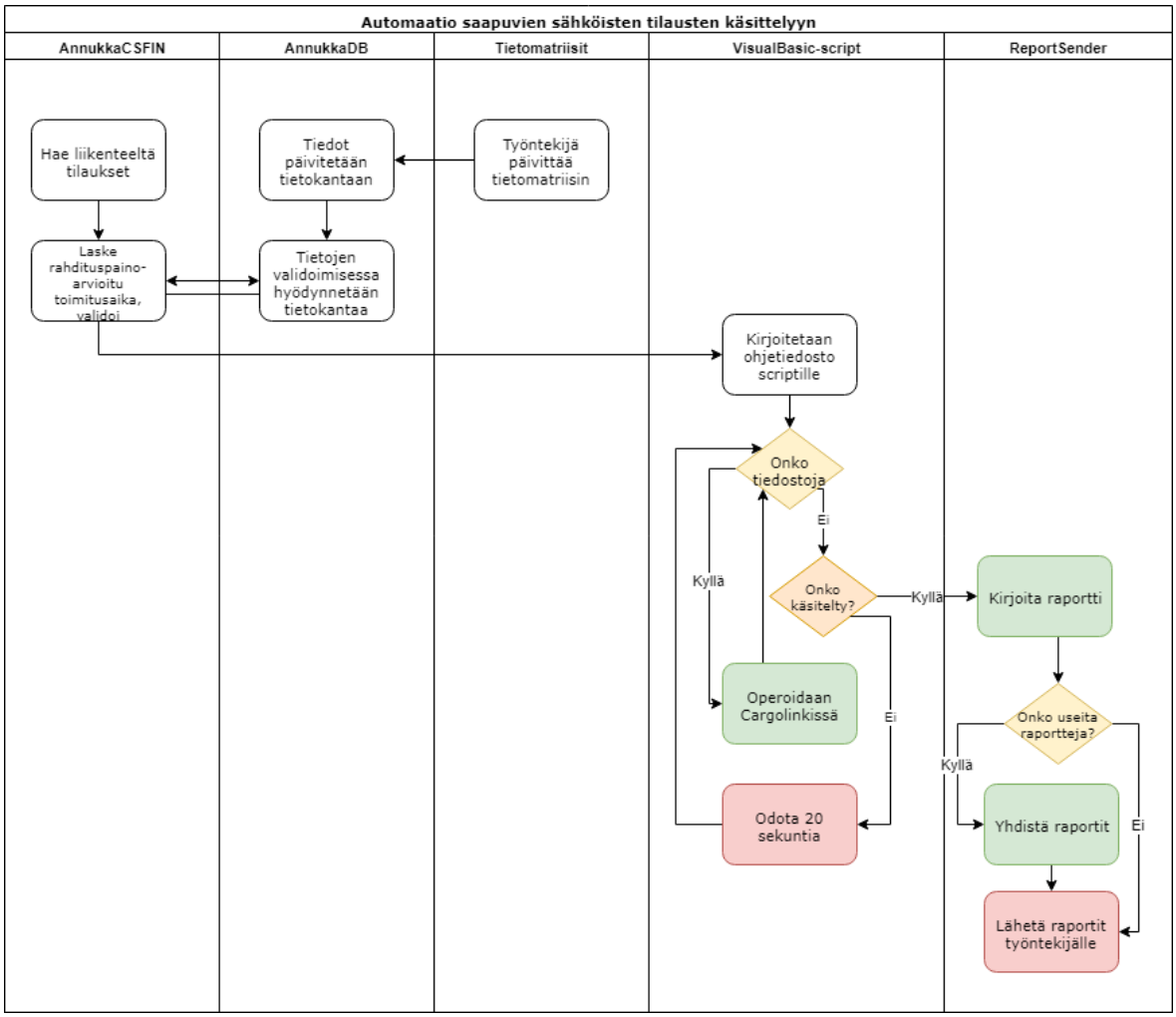
Annukkaa varten perustettiin oma liikenne, jonne kartoitettiin saapuvia tilauksia asiakas kerrallaan. Ensimmäiseksi käsiteltävät tilaukset haetaan tietokannasta ja käsitellään

ohjelmistorobotin avulla. Robotin askelissa suoritetaan tietokannasta saapuvien sekä tietomatriiseista löytyvien arvojen perusteella laskentaa rahdituspainosta, toimituspaikasta sekä arvioidusta saapumisajasta. Virheellisillä tiedoilla saapunut tilaus menee ihmisen käsittelyyn. Virheellinen tieto voi olla esimerkiksi syöttämättä jätetty rahdituspaino, väärä postinumero toimitusosoitteessa tai liian aikaiseksi arvioitu toimitusaika. Asiakkaasta riippuen tietojen pakollisuus vaihtelee: yhdellä asiakkaalla on pakollisena tietona rahdituspaino, jollakin toisella asiakkaalla asiakkaan ilmoittamaa ei edes oteta huomioon vaan rahdituspaino lasketaan tilauksen mittatiedoista.

Cargolinkin tietokantaan ei saa kirjoittaa suoraan, vaan kaikki tilaukset on käsiteltävä käyttöliittymässä. Tämän takia seuraava askel prosessissa on kirjoittaa jokaisesta käsiteltävästä tilauksesta ohjetiedosto VB6 –ohjelmistomakrolle. Ohjetiedosto on tavallinen tekstitiedosto, joka sisältää kaikki tiedot, mitä tilauksen käsittely vaatii kuljetusjärjestelmässä. Tietojen erotinmerkkinä toimii ;-merkki. Tämä VB6 –ohjelmistomakro vahtii nimitettyä verkkolevyä ja aina robotin kirjoittaman ohjetiedoston ilmestyessä kyseiselle verkkolevylle, lukee ohjelmistomakro tiedoston ja operoi tietojen perusteella Cargolinkissa. Verkkolevyn tyhjentyessä ohjelmistomakro kirjoittaa raportin käsitellyistä tilauksista.




Kuva 10 Vuokaavio Annukan toiminnasta




Kuva 11. Swim Lane -kaavio Annukan toiminnasta

Kuvassa 10 keltaisella merkitty AnnukkaCSFIN-robot, tekee edellä mainittua tilauksien hakemista ja validoimista. Sinisellä merkitty VisualBasic -script, kuvastaa ohjelmistomakron toimintaa Cargolinkissä. Kuvassa 10 on myös robotit AnnukkaDB-robot ja ReportSender-robot. AnnukkaDB-robot pitää yllä tietomatriiseista koostuvaa tietokantaa, jotta robotit voivat suorittaa tilauksien laskennan oikein ja ReportSender-robot, joka koostaa ohjelmistomakron tuottamista raporteista yhden lähettäen sen asiakaspalvelu osaston sähköpostilaatikkoon. Kuvan 11 Swim Lane-kaavio vielä havainnollistaa komponenttien yhteistoiminnan. AnnukkaDB-robotti sekä AnnukkaCSFIN-robotti tuottavat tiedoston ohjelmistomakrolle, joka operoi Cargolinkissä. Viimeiseksi ReportSender-robotti yhdistää useat raportit ja välittää ne työntekijälle.

### 3.3.3 Perustelut

Suunnitelmassa projektin osat pilkottiin modulaarisen kehittämisen keinoin pienempiin kokonaisuuksiin, jolloin syntyi monia robotteja ja ohjelmistomakro. Tämä oli hyvä tapa,

sillä kehittämistyö voitiin jakaa helposti kahden kehittäjän välille ja yksittäisten komponenttien testaaminen helpottui.

RoCoE tarjoaa Web-palveluna Cargolink emulaattorin. Tämän käyttäminen Kapow:lla on hidasta ja KCU-pisteitä kuluttavaa eikä ohjelmisto sovellu työpöytäjärjestelmien käyttämiseen. Siksi päädyin ratkaisuun, jossa kuljetusjärjestelmässä operoiminen hoidetaan VB6 –ohjelmistomakrolla.

Ensimmäinen suunnitelma oli se, että robotti kirjautuisi virtuaalitietokoneelle ja sitä kautta käynnistäisi ohjelmistomakron. Ongelmaksi muodostui se, että RoCoE ei voinut asentaa Cargolinkiä virtuaalitietokoneille lisenssiteknisistä syistä. Meillä oli oma virtuaalitietokone testausvaiheessa, jolle olimme asentaneet Cargolinkin ja kehitystyö oli jo pitkällä, kun saimme tietää tästä. Pyysimme silloin, voisimmeko saada yhden oman virtuaalitietokoneen, jossa olisi Cargolink.

Emme voineet käyttää kenenkään asiakaspalveluosaston työntekijän tietokonetta käynnistämään makroa, sillä heillä on tietokoneiden sijasta kevyet asiakaspäätteet. Makron täytyy olla päällä koko ajan, ja kevyen asiakaspäätteen mennessä näyttölukitukseen, yhteys Cargolinkiin katkeaa. Emme pystyneet ajamaan kevyissä asiakaspäätteissä skriptejä, jotka olisivat pitäneet näytön auki koko päivän ajan. Lopulta hankimme ylimääräisen tietokoneen osastollemme vain ohjelmistomakrojen ajamiseen. Käytämme samaa laitetta muissakin vastaavissa prosesseissa.

Ohjelmistorobotti toimi prosessissa ikään kuin integraationa tietokannan tietojen ja Cargolinkissä toimivan ohjelmistomakron välillä. POC-versiossa tietomatriisit luettiin suoraan ohjelmistorobottiin, jossa laskenta suoritettiin. Tämä osoittautui kuitenkin KCU-kulutusta ajatellen suorituskyvyttömäksi ratkaisuksi. RoCoEn Kapow koulutuksessa lokakuussa 2018 kuulin, että kehittäjillä on mahdollisuus käyttää RoCoE:n tarjoamaa Microsoft SQL Server relaatiotietokantaa projekteissa. Tästä syystä POC:in jälkeen suunnitellussa opinnäytetyössä tietomatriisien tiedot luettiinkin robotin sijasta relaatiotietokantaan, josta tilauksen käsittelyyn tarvittavat tiedot haettiin. Myöhemmin päivitin myös POC:in toimimaan relaatiotietokannan avulla.

### 3.3.4 Testaaminen

Testaaminen suoritettiin Cargolinkin testiympäristössä. Suoritin projektin toisen BCM-osaston kehittäjän kanssa testaamista kopioimalla oikeita tuotantotilauksia Cargolinkin

testiympäristöön. Tämä tehtiin siirtämällä oikeita jo saapuneita EDI-sanomia testiympäristön FTP-laatikoihin.

Tilauksia käsiteltiin samalla tavalla kuin Annukka tuotantoympäristössäkin käsittelisi. Robotit hakivat liikenteestä tilauksia ja muodostivat niistä ohjetiedostoja. Ohjelmistomakroa testattiin robottien muodostamilla ohjetiedostoilla. Tuotantotilauksia piti myös toisinaan muokata, jotta saatiin testattua erilaisia rajatapauksia. Tämä tehtiin joko muokkaamalla käsin EDI-sanomia tai manipuloimalla arvoja robottien sisällä.

Toisinaan testaaminen suoritettiin vajavaisesti: pienien päivityksien jälkeen saattoi ilmetä erikoisia virheitä. Näitä testivirheitä tuli paljon, mutta vähenivät projektin loppua kohden, kun ymmärsin testaamisen tärkeyden.

### 3.3.5 Aikataulu

Suunnitelmana oli saada ratkaisu valmiiksi vuoden 2019 alkupuolella. Projekti oli tuotantoympäristössä 28.3.2019. Projekti oli kuitenkin jo aiemmin käynnissä, kun tilauksia käsiteltiin pilotoinnissa jo helmikuun lopusta asti. Maaliskuun aikana korjasin esille nousseet virheet ja projekti julkaistiin tuotantoympäristöön. Tämän lisäksi tein itsenäistä projektisuunnitelman ja toiminnallisen opinnäytetyöraportin kirjoittamista.

### 3.3.6 Ajansäästö

Asiakaspalvelupäällikkö Maija Naumasen (2019) haastattelun perusteella työntekijä käsittelee keskimäärin 30 sähköistä vientitilausta tunnissa. Työajansäästö on suoraan verrannollinen siihen, kuinka monta tilausta Annukka käsittelee. Tällä hetkellä ratkaisu on tuotannossa kahdessa eri toimintaympäristössä. POC-Annukassa saavutetaan n. 95 % tehoaste tilausten käsittelyssä, millä vapautetaan 0,7 ihmisen päivittäinen työaika muihin tehtäviin. Toisessa toimintaympäristössä, joka on tämän opinnäytetyön produkti, saavutetaan n. 75 % tehoaste, millä vapautetaan 0,8 ihmisen päivittäinen työaika muihin tehtäviin. (Naumanen, M. 2019). Tehoasteen vaihtelu johtuu siitä, että POC-Annukan ympäristö on suljettu, jossa käsitellään vain yhden asiakkaan tilauksia. Toisen toimintaympäristön piirissä on useiden asiakkaiden tilauksia ja eri asiakkaiden vaihtelevien käytäntöjen johdosta kaikkea Annukka ei pysty vielä käsittelemään.

Ne tilaukset, joita Annukka ei pysty käsittelemään alusta loppuun, jäävät virheeseen. Virheeseen jäävällä tilauksella tarkoitetaan sellaista tilausta, jonka Annukka siirtää ihmisen käsiteltäväksi. Tämä voi johtua virheellisen osoitetiedon tai liian aikaiseksi toivotun toimituspäivämäärän vuoksi. Virheeseen jäävien tilausten syyt ovat vaillinaiset

tilaustiedot, tilaukset jotka sisältävät vaarallisia aineita sekä tilaukset, joissa arvioitu saapumisaika on sovitusta tiukempi.

### 3.3.7 Produkti

Lopullisena tuotteena syntyi VB6-ohjelmistomakro ja viidestä Kofax Kapow -ohjelmistorobotista koostuvan automaatioprosessin ensimmäinen tuotantoversio, joka käsittelee satoja tilauksia viikossa. Sen lisäksi tuotin dokumentaation robotin tietokannan päivittämisestä käyttäjille, jotta tilausten käsittelyä voitaisiin muuttaa muuttamalla matriiseista koostuvan tietokannan arvoja. Näin ei tarvitsisi tehdä muutoksia ohjelmistorobotteihin. Dokumentaatiossa on ohjeet erityissääntöjen käyttämisestä, joita voi syöttää mm. tietyn asiakasnumeron, toimituspaikan, vastaanottajan asiakasnumeron tai jonkun kombinaation avulla. Lisäksi syntyi myös RoCoEn vaatima Solution Design Document.

Kapow-robotit eivät suinkaan ole ilmaisia, vaan miljoonan KCU-pisteen kulutus kuukaudessa maksaa osastollemme 4.50 euroa, jotka kohdennetaan ratkaisuiden tilaajille. Tällä hetkellä Annukka-prosessi maksaa muutamia kymmeniä euroja osastolle ja näin ollen on erittäin halpa ja tehokas ratkaisu.

## 3.4 Yhteenveto

Automaation kohteena on DSV:n sisäinen osasto, jonka ongelma on ohjelmistorobotiikalle tyypillisin: toistuva säännönmukainen tietotyö. Kehittäminen oli halpaa ja nopeaa, sillä se hoidettiin talon sisäisesti eikä uutta ohjelmistoa tarvinnut kirjoittaa. Pieniä muutoksia ei tarvitse myöskään ohjelmallisesti muuttaa, sillä automaatioprosessissa olevia arvoja käsitellään päivitettävissä olevien tietomatriisien avulla. Työ toteutettiin nelihenkisessä tiimissä, jossa päävastuussa oli opinnäytetyön tekijä. Apuna olivat toinen BCM-osaston kehittäjä, tuotteen omistaja, sekä loppukäyttäjä. Tiimi kokoontui viikoittain keskustelemaan projektista. Alkuperäinen suunnitelma ei toteutunut täysin odotetusti, kun virtuaalitietokoneympäristöissämme ei ollutkaan Cargolinkiä, mutta saimme ratkaistua ongelman muuttamatta ratkaisun arkkitehtuuria.

# 4 Pohdinta

## 4.1 Tulosten tarkastelu

Annukka projektina oli tekijälle ensimmäinen oma projekti. Käytössä olevat teknologiat sopivat hyvin opinnäytetyön tekijälle ja haasteeksi ei noussut opinnäytetyön tekijän osaamattomuus, mutta kokemattomuus. Monet asiat kuten tietokannan käyttäminen apuvälineenä, tuli mukaan vasta projektin kehitystyön ollessa jo pitkällä. Syy oli yksinkertainen: kaikkia vaihtoehtoja ei ollut kehittäjän tiedossa.

Toinen asia, joka hidasti kehitystyötä kaiken kaikkiaan oli vähitellen saapuvat vaatimukset. Kun projektitiimi tapasi viikoittain ja kokoontui keskustelemaan projektista niin uusia vaatimuksia ei tullut joka kerta. Vaatimuksia välillä odotettiin viikkoja ja joskus aikaisemmin toteutettu ratkaisu joutui vaihtoon seuraavien vaatimuksien tullessa. Hyvänä esimerkkinä voidaan pitää tilannetta, jossa ohjelmistomakro törmää virheeseen puuttuvan tiedon vuoksi. Ilmenee, että Cargolinkissä on jollakin tietyllä asiakkaalla pakkosyöttökenttänä tieto, joka ei ole vaatimuksissa. Muutos pitää tehdä robottiin, joka kirjoittaa ohjetiedostoa ohjelmistomakrolle sekä ohjelmistomakroon, jotta tilaus käsiteltäisiin oikein.

Onnistuin kaikesta huolimatta mallikkaasti ja se näkyy jo DSV:llä työajan vapautumisena asiakaspalveluosastolla. Annukka vapauttaa ihmisten työaikaa muihin asiakaspalvelutehtäviin, johon tarvitaan luovuutta ja ongelmanratkaisutaitoa. Nyt käytetty ohjelmistorobotiikka ei teknisesti kykene kuin mekaaniseen työhön. Annukka myös tasoittaa työn kuormittavuutta ja jakautumista niin sanotuissa piikkitilanteissa, missä tilausmäärät ovat hetkittäin keskiarvoa korkeammat ja paine käsittelyssä on huomattava. Näillä molemmilla on vaikutus paitsi työstä suoriutumiseen, myös työssä viihtymiseen. Annukka myös nopeuttaa tilausten käsittelyä, jolla on myös suora vaikutus kuljetusketjun toimintaan laajemmin ja osaston ulkopuolella: entistä nopeampi tilauskäsittely sallii kuljetustapahtuman optimoinnin ja toimeenpanemisen aiempaa paremmin tuotannon puolella. Myös tilauskäsittelyn laatu on parempi, jolloin toteutus onnistuu todennäköisemmin poikkeamitta. (Naumanen, M. 2019).

Tulos ei ole yllättävä, sillä prosessissa manuaalisen työn siirtäminen kokonaan automatiikan pariin säästää aikaa juuri sen verran kun manuaalista työtä voidaan ihmiseltä siirtää pois. Merkitys tulee vain kasvamaan tulevien päivitysten myötä, kun suurempi osa käsiteltävistä tilauksista menee Annukan hoiviin.

### 4.2 Eettiset näkökohdat

Eettisiä kysymyksiä tämän projektin lopputuotoksesta on se, että millä tavalla DSV aikoo hyödyntää tulevan tuottavuuden nousun asiakaspalveluosaston osalta. Allokoidaanko työntekijät vaativampiin tehtäviin vai jätetäänkö kenties hakematta työntekijä seuraavan vapautuvan työpaikan tilalle? Toisaalta voidaan ajatella, että koska ratkaisu on kehitetty nimenomaan tietojenkäsittelyn rutiinityön korvaamiseen, työntekijöiden viihtyvyys lisääntyy.

Asiakaspalveluosastolla Annukka on otettu hyvin vastaan, ikään kuin työntekijänä osaston jäseneksi jo suunnitteluvaiheessa. Annukka ei lopulta toimikaan täysin yksin: Annukka tarvitsee vahdin, joka seuraa muodostuneita raportteja ja ilmoittaen virheistä kehittäjälle sekä mahdollisesti asiakkaalle, jotta heidän tilauksensa eivät jäisi ”putkeen”.

### 4.3 Johtopäätökset sekä kehittämis- ja jatkotutkimusehdotukset

Sähköisten vientitilauksien käsittelemiseen sopi RPA –ratkaisu. Tilaukset ovat säännönmukaisia ja niitä on paljon. Manuaaliseen käsittelemiseen menevä aika on suuri. Seuraava luonnollinen jatkokehitys prosessiin on tuontitilauksien mukaan ottaminen. Tämä tapahtuukin opinnäytetyöntekijän toimesta, mutta on rajattu tämän opinnäytetyön ulkopuolelle. Asiakaspalveluosastolla käsitellään jatkossakin manuaalitilauksia, eli puhelimitse tai sähköpostitse saapuvia tilauksia.

Annukkaa voitaisiin laajentaa käsittelemään myös manuaalisia tilauksia, poissulkien puhelimitse saapuvat. Manuaalisten tilausten käsittely on otollinen tapaus älykkäälle ohjelmistorobotiikalle. Silloin tekoälysovellus joutuisi arvioimaan tilaustietoja ja muodostamaan niistä oikean tilauksen. Muodostetun tilauksen voisi käsitellä samaan tapaan kuin nyt Annukka käsittelee.

Mielenkiintoista on nyt odottaa Cargolinkin järjestelmän poistumista seuraavan Cargolink WayForward -työnimellä olevan järjestelmän tieltä. Kun uusi järjestelmä tulee käyttöön, on taas myllerryksen aika ja automaatioprosessi täytyy luoda uudestaan.

### 4.4 Opinnäytetyöprosessin ja oman oppimisen arviointi

Opinnäytetyöprosessi meni suurimmalta osin omalla painollaan. Projekti edistyi odotetulla tahdilla ja valmistui lähes aikataulussa. Opin paljon tehdessäni ja ratkaisun toinen versio oli hurja parannus POC:iin.

Kun olin jo lähettänyt ensimmäisen kerran ratkaisun esituotantoon niin projektin liiketoimintaprosessia tarkasteleva RoCoE:n BCM-osaston manageri halusi jutella kanssani prosessista. Sivuhuomiona hän kysyi, että minkä takia me käytämme tähän ohjelmistorobotiikkaa, sillä kaiken olisi voinut tehdä myös Visual Basic 6:lla.

Kun aloitimme projektin, Kapow oli niin kovasti jo lyöty lukkoon teknologiaksi, että minulla ei käynyt mielessänikään, että kaiken olisi voinutkin tehdä suoraan ohjelmistomakrolla. Mielestäni loin erinomaisen suunnitelman, mutta olin katsonut prosessia kuitenkin erittäin kapeasta näkökulmasta. Mikäli olisin suunnitellut kaiken toimimaan VB6:lla, olisimme säästäneet rahaa, sillä ohjelmistomakron käyttäminen ei maksa mitään.

Toisaalta kehittäminen oli nopeampaa ja joustavampaa käyttäen Kapowia ja Asiakaspalveluosaston kevyiden asiakaspäätteiden vuoksi olisimme kuitenkin joutuneet hankkimaan uuden palvelimen makron ajamista varten. Uuden palvelimen hankkiminen ei olisi kuulostanut järkevältä vaihtoehdolta suunnittelemisen alkuvaiheessa.

Testaaminen olisi pitänyt hoitaa suunnitelmallisemmin. Monesti uusien vaatimuksien jälkeen testaaminen suoritettiin vajavaisesti ja virheitä syntyi käytännössä aina. Projektin loppupuolella opinnäytetyön tekijä ymmärsi testaamisen tärkeyden ja piti aina isoa testimassaa, jotka piti läpäistä uusien päivityksien alla.

Työn dokumentaation olisi voinut hoitaa paremmin, mutta sitä ei unohdettu kokonaan. Kirjoitin opinnäytetyödokumenttia aina viime tingassa. Viikkopalaverista lähetettiin toisinaan muistio sähköpostilla kaikille osallistujille.

Olen onnekas, kun sain tehdä opinnäytetyön oikeaan tarpeeseen. Kun tekee tarpeellista työtä, jonka valmistumista odotetaan, on mahdollista saada taakseen hyvät tukijoukot. Sain sopivasti neuvoja saaden silti viimeisen sanan projektin toteuttamisessa. Kaiken kaikkiaan koin, että projekti onnistui kokonaisuutena.

## Lähteet


Aaltonen, P. 12.6.2017. Sähköinen tilaaminen vähentää virheitä ja lisää tehokkuutta. Teknologiainfo. Luettavissa: https://www.teknologiainfo.com/logistiikka/sahkoinen-tilaaminen-vahentaa-virheita-ja-lisaa-tehokkuutta/ Luettu: 3.12.2019

Anderson, T. 2.2.2017 Is it the beginning of the end for Visual Basic? Microsoft to focus on 'core scenarios': Hello darkness my old friend. The Register. Luettavissa: https://www.theregister.co.uk/2017/02/02/our_strategy_for_visual_basic_has_shifted_microsoft_to_focus_on_core_scenarios/ Luettu: 25.2.2019

BeeckerCO. 12.2.2019 Macros VS Robotics: when to implement automation. Luettavissa: https://www.beeckerco.com/blog/macros%20vs%20robotics%20when%20to%20implement%20automation Luettu: 18.4.2019

Bisbal, J., Lawless, D., Wu, B & Grimson, J. 1999. Legacy Information systems: Issues and directions. IEEE software X, X,103–111.

CFB Bots. 7.4.2018. 10 Best Practices for New Robotic Process Automation (RPA) Developer. Luettavissa: https://medium.com/@cfb_bots/10-best-practices-for-new-robotic-process-automation-rpa-developer-220f7ccdf33. Luettu: 16.4.2019

Craig, A. Lacity, M. & Willcocks, L. 2015a. The IT Function and Robotic Process Automation. Luettavissa: http://eprints.lse.ac.uk/64519/1/OUWRPS_15_05_published.pdf

Craig, A. Lacity, M. & Willcocks, L. 2015b. Robotic process automation at Telefonica O2. Luettavissa: http://eprints.lse.ac.uk/64516/1/OUWRPS_15_02_published.pdf

Cleverism. 2018. Visual Basic. Luettavissa: https://www.cleverism.com/skills-and-tools/visual-basic/ Luettu: 25.2.2019

Data Communication Solutions. 29.3.2016. Knowing the Difference Between Ansi X12 And Edifact. Luettavissa: https://www.dcs-is-edi.com/2016/03/knowing-the-differences-between-ansi-x12-and-edifact/ Luettu: 27.2.2019

Desmond, M. 2012 Editor's Note – Old Soldiers Never Die MSDN Magazine, 27, 6. Luettavissa: https://msdn.microsoft.com/en-us/magazine/jj133813.aspx Luettu: 25.2.2019

Devonit. 2019. What is a Thin Client? Luettavissa: http://www.devonit.com/thin-client-education Luettu: 15.3.2019

Deloitte 2017. The robots are ready. Are you? Untapped advantage in your digital workforce. Luettavissa: https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology/deloitte-robots-are-ready.pdf Luettu: 22.4.2019.

Edicom. 2019. Get Started with EDI. Luettavissa: https://www.edicomgroup.com/solutions/edi/what_is.html Luettu: 27.2.2019

DSV. 2019. Meistä. Luettavissa: http://www.fi.dsv.com/about-dsv Luettu: 24.4.2019

Forcepoint. 2019. Thin Client Defined. Luettavissa: https://www.forcepoint.com/cyber-edu/thin-client Luettu: 15.3.2019

Forrester 2014. Building A Center Of Expertise To Support Robotic Automation – Preparing For The Life Cycle Of Business Change. Luettavissa: http://neoops.com/wp-content/uploads/2014/03/Forrester-RA-COE.pdf Luettu: 18.4.2019.

Haikonen, M. 23.9.2016. Osa 2: Ohjelmistorobotiikka – mitä se on? LindekIn. Luettavissa: https://www.linkedin.com/pulse/osa-2-ohjelmistorobotiikka-mist%C3%A4-siin%C3%A4-kyse-mika-haikonen?trk=portfolio_article-card_title Luettu: 15.2.2019

Holmukhe, R. Madakam, S. & Jaiswal D. 2019. The Future Digital Work Force: Robotic Process Automation (RPA). Luettavissa: http://www.scielo.br/pdf/jistm/v16/1807-1775-jistm-16-e201916001.pdf

Helsingin yliopisto & Reaktor 2018. Exercise 3: Examples of tasks. Elements of AI -verkkokurssi. Luettavissa: https://course.elementsofai.com/1/2. Luettu: 15.2.2019

Huemer, C. 2000. XML vs. UN/EDIFACT or Flexibility vs. Standardisation. Luettavissa: https://pdfs.semanticscholar.org/c187/6f9e4ee352a5e77413a4a0cac09a91a98dfd.pdf Luettu: 23.4.2019

IBM 2018. IBM Mainframes. Luettavissa: https://www.ibm.com/ibm/history/exhibits/mainframe/mainframe_intro.html Luettu: 27.2.2019

Ispliter. 14.1.2014. Visual Basic 6.0: A giant more powerful than ever. Codeproject. Luettavissa: https://www.codeproject.com/Articles/710181/Visual-Basic-6-0-A-giant-more-powerful-than-ever Luettu: 25.2.2019

Järvenpää, L. 21.8.2018. Mitä on koneoppiminen? Lamia. Luettavissa: https://lamia.fi/blog/mita-on-koneoppiminen Luettu: 27.2.2019

Kofax. 2018a. About us. Luettavissa: https://www.kofax.com/About/our-company/about-us Luettu: 15.2.2019

Kofax. 11.7.2018b. Kofax Kapow User's Guide – Version: 10.3.2. Luettavissa: https://knowledge.kofax.com/Robotic_Process_Automation/00_Product_Documentation__User_Guides/00_Kapow_Product_Documentation_Master/00_Kofax_Kapow_103_Product_Documentation/All_Kofax_Kapow_10.3_Product_Documentation Luettu: 15.3.2019

Kotimaisten kielten keskus. 29.3.2019. Lyhenneluettelo: E. Luettavissa: http://www.kielitoimistonohjepankki.fi/haku/OVT/ohje/373. Luettu: 29.3.2019

Lacity, M. Willcokcs, L. Leslie P. 2016. The Next Transformation Lever for Shared Services. Luettavissa: http://www.umsl.edu/~lacitym/OUWP1601.pdf

Mak, S. 24.3.2017. Modules vs microservices. O'Reilly. Luettavissa: https://www.oreilly.com/ideas/modules-vs-microservices Luettu: 27.2.2019

McCarthy, B. 19.8.2013. EDI History. The Logicbroker Blog. Luettavissa: https://blog.logicbroker.com/blog/2013/08/19/edi-history Luettu: 27.2.2019

Microsoft. 2019. Visual Basic 6.0 Resource Center. Luettavissa: https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-basic-6/visual-basic-6.0-documentation Luettu: 25.2.2019

Naumanen, M. 24.10.2019. Senior manager. DSV Road Finland Oy. Sähköposti.

Northam, C. 2.4.2017. RPA (Robotic Process Automation) and SPA (Smart Process Automation) are not the same thing! LinkedIn. Luettavissa: https://www.linkedin.com/pulse/rpa-robotic-process-automation-spa-smart-same-thing-chris-northam Luettu: 15.2.2019

Rocziek, K. Robotics Training for Developers. 21.10.2018. DSV Robotics Centre of Excellence. Koulutus. Varsova.

Suomen Huolinta- ja Logistiikkaliitto ry. 2019. Mitä huolinta on? Luettavissa: http://www.huolintaliitto.fi/tietoa-alasta/mita-huolinta-on.html Luettu: 22.4.2019

Tietoyhteiskunnan kehittämiskeskus. 2019. Tehokkuutta logistiikka-alan toimintoihin tiedonvälityksen sähköistämisellä. Luettavissa: https://oma.tieke.fi/pages/viewpage.action?pageId=18940371. Luettu: 21.11.2019

UiPath. 14.9.2014. What's the Difference Between Robots and Macros? Luettavissa: https://www.uipath.com/blog/whats-the-difference-between-robots-and-macros Luettu: 18.4.2019