Haaga-Helia
ammattikorkeakoulu Oy

Robotiikkakehittämisen tekninen laatu, laatuvaatimukset ja ohjeistukset henkivakuutusyhtiössä

Johannes Riitijoki

Opinnäytetyö
Finanssi- ja talousasiantuntijan koulutusohjelma
2020

Tiivistelmä

Haaga-Helia ammattikorkeakoulu Oy

Tekijä(t)
Johannes Riitijoki

Koulutusohjelma
Finanssi- ja talousasiantuntijan koulutusohjelma

Raportin/Opinnäytetyön nimi
Robotiikkakehittämisen tekninen laatu, laatuvaatimukset ja ohjeistukset henkivakuutusyhtiössä

Sivu- ja liitesivumäärä
33 + 4

Tämän opinnäytetyön tarkoituksena on selvittää, miten robotiikkakehittämisen tekninen laatu ja laatuvaatimukset tulisi kuvata henkivakuutusyhtiön näkökulmasta. Työssä pyritään selvittämään, mitä tulee ottaa huomioon kehittämisen tehokkuuden, elinkaaren aikaisten virheselvitysten ja jatkokehityksen näkökulmasta.

Tietoperustassa kerrotaan ohjelmistorobotiikasta, toimeksiantajan käytössä olevasta Blue Prism- ohjelmistosta, sekä teknisen laadun muodostumisesta ohjelmistokehityksessä.

Tutkimus toteutettiin teoriaohjaavan sisällönanalyysin avulla. Tutkimuksen lähtökohtana oli toimeksiantajayrityksen konsernin ohjelmistorobotiikan kehityksen ohjeistukset. Työssä haastateltiin toimeksiantajayrityksen työntekijöitä, ja arvioitiin näiden haastattelujen ja tietoperustassa opittujen ohjelmiston laadun mittareiden avulla konsernin ohjeistuksien soveltuvuutta toimeksiantajayritykselle. Toimeksiantaja toteuttaa sekä ohjelmistokehitystä että robotiikkaa ketterällä toimintatavalla.

Tutkimuksen tuloksista selvisi, että nykyisellääänkin konsernin ohjeistukset tukevat toimeksiantajan toimintaa suurilta osin. Ilmi tuli kuitenkin myös muutamia tekniseen kehitykseen liittyviä asioita, jotka eivät ole tietoperustassa esiteltyjen ohjelmiston laadun mittareiden kanssa samassa linjassa. Tuloksissa ilmeni myös, että muutoksentekologiikka konsernin ohjeistuksissa on liian raskas toimeksiantajan ketterän toiminnan kannalta.

Asiasanat
ohjelmistorobotiikka, ohjelmistokehitys, ohjelmistokehityksen laatu, henkivakuuttaminen

Sisällys

1 Johdanto 1
1.1 Toimeksiantajan kuvaus 1
1.2 Aiheen laajempi kokonaisuus ja liitännäisyydet 2
1.3 Tutkimuksen tavoitteet ja rajaus 2
1.4 Opinnäytetyön rakenne 3
2 Ohjelmistorobotiikka 4
2.1 RPA-ohjelmistot 5
2.2 Center of Excellence 5
2.3 RPA:n hallintamallit 5
2.4 RPA-ydintiimin roolit 6
3 Blue Prism 8
3.1 Blue Prismin kehitystilat ja tiloissa hyödynnettävät vaiheet 8
3.2 Control Room toiminnallisuus ja jonot Blue Prismissä 11
3.3 Muuttujat ja parametrit Blue Prismissä 12
3.4 Blue Prismin sisäänrakennettu laadun seuranta 13
3.5 Poikkeukset Blue Prismissä 13
4 Teknisen laadun muodostuminen ohjelmistokehityksessä 14
4.1 Koodin laatu ja sen tekijät 14
4.2 Keinoja parantaa koodin laatua 18
4.3 Tekninen velka 19
4.4 Ketterä kehitys 20
5 Yritys X:n konsernin ohjeistusten arviointi yritys X:n näkökulmasta 21
5.1 Yritys X:n konsernin robotiikan ohjeistukset ja vaatimukset 21
5.2 RPA yritys X:ssä 25
5.2.1 Tutkimuksen toteutus ja tutkimusmenetelmät 26
5.2.2 Aineiston hankintamenetelmä 26
5.2.3 Haastattelun tulokset 27
5.3 Konsernin ohjeistusten tarkastelu teoriaohjaavan sisällönanalyysin avulla 28
5.4 Yhteenveto 30
6 Pohdinta 32
6.1 Johtopäätökset sekä tutkimuksen kehittämis- ja jatkotutkimusehdotukset 32
6.2 Tutkimuksen luotettavuus sekä opinnäytetyön ja oman oppimisen arviointi 32
Lähteet 34
Liitteet 41
Liite 1. Haastattelujen vastaukset 41
Liite 2. Yritys X:n konsernin RPA-operointimalli 44

1 Johdanto

Tämä opinnäytetyö on tutkimus, jossa tarkastellaan ohjelmistorobotiikkakehityksen (RPA, Robotic Process Automation) menetelmiä ja niiden ohjeistusta. Tässä työssä yritys X on osa finanssikonsernia, jolla on yhteinen robotiikka-alusta ja robotiikkakehityksen ohjeistus. Konsernin ohjeistus on laadittu erityisesti suurten, kerralla RPA-tekniikalla automatisoitavien prosessikokonaisuuksien rakentamiseen. Yritys X:n kannalta robotiikkakehitys on strateginen kyvykkyys osana ketterää kehitystä, eikä osa konsernin ohjeistuksesta tue täysin sen tavoitteita.

Tässä työssä tarkastellaan yritys X:n tilannetta ja verrataan robotiikkakehitystä ohjelmistokehitykseen ja sen teknisen laadun määritelmiin. Näiden perusteella tehdään suositus teoriaohjaavan sisällönanalyysin avulla yhtiö X:n robotiikkakehityksen teknisen laadun laatuvaatimuksiksi.

1.1 Toimeksiantajan kuvaus

Toimeksiantajana tälle työlle toimii suomalainen henkivakuutusyhtiö, joka on osa isompaa finanssialan konsernia. Yhtiöön viitataan tässä työssä yritys X:nä. RPA-tiimissä on neljä täysipäiväistä työntekijää. Kaksi kehittäjää, prosessinomistaja ja kontrolleri/analysoija. Tiimiin kuuluu myös kontrollereita, jotka käyttävät vain osan ajastaan robotiikan parissa. Tämän opinnäytetyön tekijä toimii toisena kehittäjänä. Tekemistä robotiikkatiimissä kuvaillaan ketteräksi ja nopeaksi. Robotiikkatekeminen on läheistä liiketoiminnan kanssa, ja vastuu roboteista säilyy tiimillä tuotantoympäristöön viennin jälkeenkin.

Yritys X:n mallissa robotiikan tekijät ovat yhteisvastuullisia sen prosessin kanssa, johon automatisointi suoritetaan. Tämä eroaa konsernin käytössä olevasta keskitetystä hallintamallista, jossa vastuu robottien hyödyntämisestä rajataan tilaajan puolelle. Henkivakuutusyhtiöllä sekä teknisen laadun valvontavastuu että elinkaaren hallinnanvastuu on itsellä. Tämän vuoksi tarvitaan tutkimusta yllä mainituista asioista, jotta voidaan toimia ammattimaisella tavalla. Esimerkiksi sellaisiin kysymyksiin, kuten miten asiat menevät sulavasti läpi, mistä laatu robotiikassa muodostuu, sekä mistä elinkaaren hallinnan tehokkuus muodostuu, etsitään vastauksia. Henkivakuutuksessa asiakassuhteiden sopimukset elävät kymmeniä vuosia, joten ratkaisujen tulee olla sellaisia, että niitä voidaan käyttää pitkään. Työn tuloksia aiotaan hyödyntää yhtäläisyyksien hakemisessa eri alustoilla tapahtuvan kehittämisen kanssa. Yritys X vastaa omien vakuutusjärjestelmiensä kehittämisestä ketterän kehityksen malleilla ja tavoitteena on hyödyntää samoja periaatteita robotiikkakehityksessä.

1.2 Aiheen laajempi kokonaisuus ja liitännäisyydet

Käsiteltävä aihe liittyy osaan yritys X:n robotiikkaprosessin toimintamallikuvausta. Tutkimuksen tarkoituksena on konsernin yhteisten robotiikkakehitysmallien arviointi henkivakuutusyhtiön näkökulmasta. Tarkoituksena on myös hakea liittymäkohtia ohjelmistokehityksen laadun arviointiin.

Yritys X:n konsernissa on keskitetty robotiikan toimintamalli, jonka keskiössä on robotiikan osaamiskeskus (Center of Excellence, CoE), joka vastaa robotiikkaympäristöistä ja menetelmistä, sekä niiden käyttämisestä. Henkivakuutusyhtiössä on erilaisia painotuksia kuin koko konsernilla, ja sen takia se on ottamassa enemmän vastuuta oman robotiikkakehittämisensä laadusta.

Aihe on tärkeä, koska toimeksiantajayrityksessä robotiikan koetaan olevan strateginen kyvykkyys tulevaisuuden henkivakuutusyhtiölle. Työ on ajankohtainen toimeksiantajan alettua määrittämään omaa tavoitetilaansa robotiikan hyödyntämiselle. Työ on tärkeä osa kokonaisuutta, jossa halutaan kehittää omaa robotiikan toimintamallia ja varmistaa robotiikkatekemisen laatu.

1.3 Tutkimuksen tavoitteet ja rajaus

Opinnäytetyön pääkysymys on:

- Miten robotiikkakehittämisen tekninen laatu ja laatuvaatimukset tulisi kuvata henkivakuutusyhtiön näkökulmasta?

Opinnäytetyön alakysymykset ovat:

1. Mitä kehityksessä tulee ottaa huomioon kehittämisen tehokkuuden näkökulmasta?
2. Mitä tulee ottaa huomioon robottien käytön elinkaaren aikaisten virheselvitysten näkökulmasta?
3. Mitä tulee ottaa huomioon robottien jatkokehityksen näkökulmasta?

Tarkoituksena ei ollut tehdä yleistä robotiikkamallia vaan sovittaa nykyistä finanssikonsernin toimintamallia yritys X:n toimintamalliin sopivaksi. Työhön sisältyy vain mallin kuvaaminen. Jatkokehittäminen ei sisälly työhön. Työn rajauksena toimii myös yritys X:llä käytössä oleva RPA-ohjelmisto Blue Prism. Yritys X on sitoutunut hyödyntämään konsernin robotiikka-alustaa, joka pohjautuu Blue Prism -työkaluihin. Työ siis rajataan kyseiseen ohjelmistoon, eikä tutkimukseen oteta mukaan muita RPA-ohjelmistoja.

1.4 Opinnäytetyön rakenne

Opinnäytetyö koostuu johdannosta, tietoperustasta, empiirisestä osiosta ja pohdinnasta. Johdannossa kerrotaan toimeksiantajasta sekä työn taustoista ja käyttötarkoituksesta. Tietoperustassa perehdytään ohjelmistorobotiikkaan tarkemmin ja kerrotaan Blue Prism -ohjelmistosta. Tietoperustassa kerrotaan myös ohjelmistokehityksestä ja siitä, miten laatu muodostuu ohjelmistokehityksessä. Empiirisessä osuudessa kerätään tietoa yritys X:n konsernin robotiikan ohjeistuksista, sekä haastatellaan yritys X:n robotiikkatiimin jäseniä ja tietohallintojohtajaa. Näiden pohjalta tehdään analyysiä, jonka avulla voidaan arvioida konsernin ohjeita yritys X:n näkökulmasta. Pohdinnassa näiden tietojen pohjalta tehdään johtopäätöksiä sekä tutkitaan, onko saatu vastauksia tutkimuskysymyksiin. Pohdinnan lopuksi arvioidaan omaa oppimista ja tutkimuksen luotettavuutta.

2 Ohjelmistorobotiikka

Ohjelmistorobotiikka, eli RPA, on ohjelmallisesti suoritettavaa prosessiautomaatiota. Ohjelmistorobotiikassa tarkoituksena on automatisoida rutiininomaisia töitä ja toimia paikkauksena tilanteissa, joissa ohjelmistokehitys ei ole ajallisesti järkevää toteuttaa tai vaatii liikaa resursseja. (Staria 2020; Azets 2020; Digital Workforce 2020.)

Ohjelmistorobotiikka on suhteellisen uusi käsite sen yleistyttyä vasta viime vuosina. Ala kasvaa vuosi vuodelta, Grand View Research on arvioinut sen olevan 10,7 miljardin dollarin arvoinen bisnes vuoteen 2027 mennessä. (Grand View Research 2020.)

Ohjelmistorobotit hyödyntävät olemassa olevia käyttöliittymiä, eivätkä robotit vaadi erillisiä ohjelmointirajapintoja (Staria 2020). Ohjelmistorobottien kehittäminen on nopeaa ja ketterää. Ohjelmistorobotiikka soveltuukin hyvin yritysten suurten perusjärjestelmien korjauksien korvaajaksi. (Digital Workforce 2020.)

Ohjelmistorobotiikassa automatisoitava ohjelma/ohjelmat mallinnetaan automatisoitavien elementtiensä osalta. Tämän jälkeen automatisoitava prosessi rakennetaan vuokaaviooperiaatteella toimivalla ohjelmalla. Robotiikkatyökalusta löytyvät perinteiseen ohjelmointiin liittyvät osat, kuten laskutoimitukset, muuttujat, toistorakenteet, if-lauseet ja tietotaulukot. (UiPath s.a.c.)

Joistain RPA-ohjelmistoista löytyy myös ohjelmistokehityksestä tutut objektit, joihin ohjelmistorobotiikan tapauksessa mallinnetaan automatisoitavia ohjelmia ja niiden elementtejä. Objektien sisälle pystyy luomaan erilaisia robotin käyttämiin perusohjelmistoihin liittyviä toimintoja, kuten klikkauksia, kenttiin kirjoittamista ja tekstien tai taulukoiden lukemista. Näitä toimintoja pystyy uudelleenkäyttämään useissa eri prosesseissa. (Tutorialspoint s.a.a.) Tämä nopeuttaa uusien prosessien kehittämistä, mikäli niiden on mahdollista hyödyntää olemassa olevia objekteja ja niiden toimintoja.

RPA-ohjelmistoissa on myös mahdollista kirjoittaa koodia, kuten esimerkiksi C# tai Visual Basic. Koodin avulla voidaan toteuttaa asioita, jotka olisivat ohjelman muilla sisäisillä toiminnoilla hankalia tai mahdottomia. (Learning Technologies 2018c; UiPath s.a.b.)

Ohjelmistorobotit työskentelevät joko työasemalla tai virtuaalityöpöytäympäristössä (Staria 2020). Virtuaalitietokoneita kutsutaan ohjelmistorobotiikan termein resursseiksi. Robotit käynnistyvät niille luodun aikataulun mukaisesti, aikataulun määräämällä resurssilla.

(DMS-Solutions s.a.) Robotille voidaan antaa aloitusparametreja joita robotti hyödyntää ajonsa aikana (Wells 30.5.2019).

2.1 RPA-ohjelmistot

Ohjelmistorobotiikan saralle on viime vuosina tullut useita eri ohjelmistoja. Gartnerin keräämien tietojen mukaan vuonna 2018 neljä markkinaosuudeltaan suurinta RPA-ohjelmistoa olivat UiPath 13,6 % markkinaosuudella, Automation Anywhere 12,8 % markkinaosuudella, Blue Prism 8,4 % markkinaosuudella ja NICE 7,3 % markkinaosuudella. Loput ohjelmistot omaavat kukin alle viiden prosentin markkinaosuuden. Tässä työssä keskitymme yritys X:n käytössä olevaan Blue Prism -ohjelmistoon. (Gartner 2019.)

2.2 Center of Excellence

Ohjelmistorobotiikan osaamiskeskus (Robotics Center of Excellence, CoE) on taho, joka määrittää robotiikkatekemisen perusteet, ohjeistukset ja standardit organisaation sisällä. CoE:n vastuihin kuuluu ohjeistuksien ja sääntöjen luomisen ohella myös robotiikkainfrastruktuurin ylläpito ja kehittäminen, sekä uusien automatisointimahdollisuuksien tunnistaminen. CoE on usein myös vastuussa uusien työntekijöiden perehdyttämisestä robotiikan pariin. (RPA-prosessinomistaja 27.4.2020.)

CoE määrittää organisaation RPA-toimintojen käyttöönoton, hallinnan ja skaalautumisen, ja se luetaankin tärkeäksi osaksi onnistunutta RPA:n toteuttamista (Stoddard 30.10.2019). CoE voi olla ulkoistettu esimerkiksi RPA-ohjelmiston tarjoajalle, tai RPA:ta hyödyntävä yritys voi muodostaa myös oman CoE:n (UiPath s.a.a; Rebeiro s.a.).

2.3 RPA:n hallintamallit

RPA:lle on olemassa kolme erilaista hallintamallia, joista kahdessa CoE on keskeisessä osassa. Hallintamalli kuvaa sitä, miten RPA on organisaatiossa toteutettu. Hallintamallien nimistä löytyy eri näkemyksiä, mutta tyypillisesti hallintamallit muodostuvat alla olevista kolmesta mallista.

Keskitetyssä hallintamallissa on keskitetty tiimi, joka tekee ohjelmistorobotit kaikkiin yrityksen osastoihin. Keskitetyssä mallissa resurssiriski minimoidaan, kun osaaminen RPA-tiimissä leviää paremmin kuin muissa hallintamalleissa keskitetyn rakenteen ansiosta. Keskitetty hallintamalli mahdollistaa parhaan keskittymisen yhteen RPA:n suurimpaan tavoitteeseen, eli kulujen karsimiseen. Keskitetyssä mallissa myös tiettyjä taitoja pystytään paremmin hyödyntämään suuremman osaajapohjan ansiosta. (King 19.2.2017.)

Keskitetty malli ei kuitenkaan ole kovin ketterä. Keskitetty malli myös heikentää yhteydenpitoa liiketoimintaan suuren rakenteensa vuoksi. RPA-tiimeillä on useiden liiketoiminnan prosessien kanssa yhteistyötä, eivätkä näin ollen pysty keskittymään yhteen tiettyyn liiketoiminnan alueeseen samalla tasolla kuin hajautetussa mallissa. (King 19.2.2017.)

Hajautetussa mallissa, joka kulkee myös nimellä federoitu hallintamalli, jokainen liiketoiminnan osasto kehittää omat ohjelmistorobottinsa. Hajautetussa mallissa robotiikkatiimit ovat läheisesti liitoksissa liiketoimintaan ja ovat kykeneviä nopeisiin toimenpiteisiin. Hajautetussa mallissa ei ole CoE:tä, joka olisi millään tavalla ohjaamassa tai rajoittamassa robotiikkatekemistä. Tämä on kuitenkin myös huono asia, sillä taitojen vaihtelevuus yrityksen sisällä kasvaa ja tekeminen on erilaista joka tiimissä. Tämä lisää robotiikan kuluja ja alentaa kontrollointimahdollisuuksia organisaation sisällä. Malli on myös altis resurssiriskille, sillä pienissä tiimeissä ei välttämättä ole resursseja toteuttaa laajempia kokonaisuuksia. Myös uusien teknologioiden implementointi on haasteellisempaa pienessä tiimissä. (King 19.2.2017.)

Hybridimalli on välimalli keskitetystä ja hajautetusta mallista. Hybridimallissa on CoE, joka määrittää standardit, rakenteen, käytetyn RPA-ohjelmiston ja robotiikan hallinnan. Hybridimallissa CoE:n tarkoitus on toimia paikallisten tiimien tukena päivittäisessä tekemisessä. (King 19.2.2017.)

Hybridimalli on ketterämpi keskitettyyn malliin verrattuna, sekä se tasoittaa muutosten sietokykyä hajautetun ja keskitetyn mallin välillä. Hybridimalli mahdollistaa myös olemassa olevan tiedon hyödyntämisen yrityksen sisällä, mitä hajautettu malli ei mahdollista. Hybridimallissa kehittämisen nopeus ei kuitenkaan ole korkeimpana prioriteettina, mikä voi hidastaa robottien valmistumista. Tyypillisesti myös ensimmäisen tason tekninen tuki on hybridimallissa sama kuin liiketoiminnalla. Hybridimallissa paikalliset tiimit ovat vastuussa robottien versiohallinnasta. (King 19.2.2017.)

2.4 RPA-ydintiimin roolit

Robottien rakentamisen ja ylläpidon ydintiimi muodostuu kolmesta roolista. **RPA Developer**, jäljempänä kehittäjä, osallistuu robotin suunnitteluun, kehitykseen, implementointiin ja dokumentointiin. Kehittäjä siis suunnittelee robotin teknisen osuuden, sekä rakentaa ja testaa robotin hänelle annettujen ohjeiden mukaisesti. Kehittäjä myös korjaa robotin, mikäli siinä esiintyy jotain ongelmia (Gates 26.2.2020).

Process designer, joka tunnetaan myös nimellä RPA Business Analyst, jäljempänä analyytikko, tutkii liiketoiminnan puolella olevia prosesseja ja niiden automatisointimahdollisuuksia hyödyntäen RPA:ta. Analyytikko tekee tarvittavan pohjatyön alkuarviointia ja kehitystä varten. Pohjatyöhön kuuluu muun muassa mahdollisten projektin esteiden ja nopeuttavien tekijöiden tunnistaminen sekä prosessin alustava dokumentointi. (Roboyo 2019.)

Production Manager, joka kulkee myös nimellä RPA Controller, jäljempänä kontrolleri, on vastuussa robotin tuomisesta tuotantoon, sekä vastuussa robotista tuotantoon siirtymisen jälkeen. Kontrolleri varmistaa, että robotit pyörivät aikataulussa, sekä tutkii mahdolliset vikailmoitukset ja robottien virhetilanteet. Kontrollerin vastuisiin kuuluu myös ongelmien raportointi kehittäjille. Kontrollerit analysoivat robottien tilastoja ja ajolokeja, sekä tekevät analyytikolle pyyntöjä mahdollisista parannuksista robottien toimintaan. (Labbe 28.1.2017.)

3 Blue Prism

Blue Prism on pörssilistatun Blue Prism Groupin markkinoille tuoma RPA-ohjelmisto. Gartnerin (Gartner 2019) vuonna 2019 keräämien tietojen mukaan Blue Prism on maailman kolmanneksi suosituin RPA-ohjelmisto. Tässä työssä tarkasteltava finanssikonserni käyttää robotiikassa Blue Prism -ohjelmistoa. Tässä osiossa katsomme syvällisemmin Blue Prismin ominaisuuksia ja toimintoja.

Robotin kehitystyö Blue Prismissä tapahtuu kahdessa kehitystilassa, Process studiossa ja Object studiossa. Molemmissa on samoja toimintoja, mutta ne eroavat silti olennaisesti toisistaan. Object studiossa mallinnetaan automatisoitava ohjelma ja tehdään tämän pohjalta toiminnot, joita myöhemmin hyödynnetään Process studiossa. Blue Prism toimii vuokaavioperiaatteella. Ohjelmassa on selkeyttämisen vuoksi mahdollista pilkkoa prosessi ja objekti usealle eri välilehdelle.

3.1 Blue Prismin kehitystilat ja tiloissa hyödynnettävät vaiheet

Object Studiossa mallinnetaan automatisoitava ohjelma elementtiensä osalta. Näiden elementtien pohjalta luodaan erilaisia toimintoja, eli actioneita, jotka ovat esimerkiksi robotin käyttämän liiketoimintaohjelmiston näppäinten klikkauksia, kenttään kirjoittamista tai tiedon lukemista. Object studiossa ohjelmien mallintamiseen hyödynnetään ominaisuutta nimeltä Application Modeller. (RPA-kehittäjä 27.4.2020.)

Application Modeller on ominaisuus Object Studion sisällä, jolla mallinnetaan automatisoitavan ohjelman elementtejä. Elementit ovat automatisoitavan ohjelman osia, kuten kenttiä, tekstejä tai taulukoita. Mallinnus tapahtuu hyvin yksinkertaisesti haluttua elementtiä klikkaamalla. Ohjelma osaa lähes aina automaattisesti poimia mallinnettavan elementin tiedot, eli attribuutit. Jos automaattinen tietojen poiminta ei jostain syystä onnistu, on Application Modellerilla mahdollista myös hakea mallinnettavan ohjelman koko elementtipuu Application Navigator toiminnon avulla. (RPA-kehittäjä 27.4.2020.)

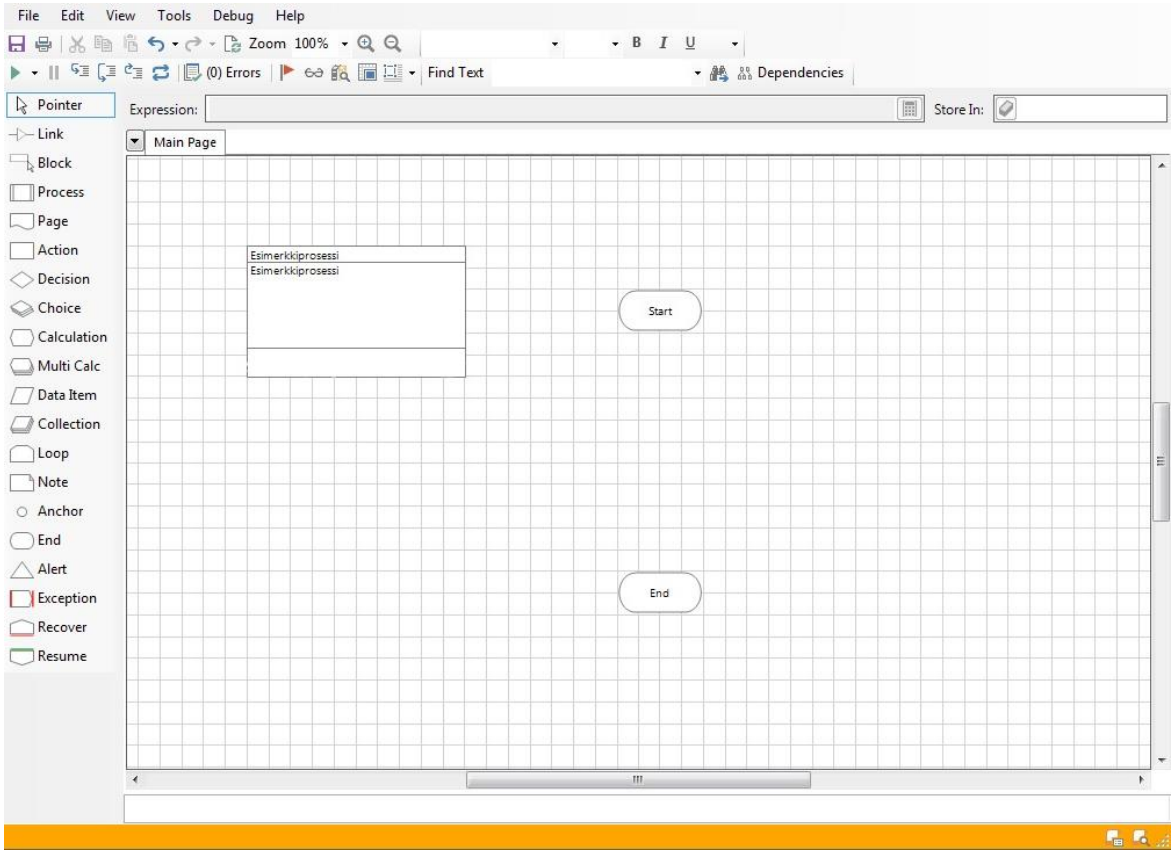
Application Modellerissä on myös tuki automatisointiin kuvien perusteella. Tätä automatisointitapaa kutsutaan Surface Automationiksi. (Tutorialspoint s.a.c.) Emme mene tähän aiheeseen sen syvällisemmin, muuten kuin kertomalla tämän tuen olemassaolosta.

Attribuutit ovat mallinnetun elementin ominaisuuksia, joiden perusteella elementti on löydettävissä. Attribuutteja voivat olla esimerkiksi mallinnettavan ohjelman sisällä oleva ele-

mentin järjestysluku, arvo tai suora polku elementtiin. Attribuutteja voi laittaa päälle ja ottaa pois päältä, tai rajata niitä tietylle välille, esimerkiksi niin, että numeraalisen attribuutin arvon pitää olla kolmen ja seitsemän välillä. Attribuuteista on mahdollista tehdä myös dynaamisia, jolloin yhtä Application Modellerin elementtiä on mahdollista hyödyntää löytämään useita mallinnettavan ohjelman samankaltaisia elementtejä, eikä tällöin ole tarvetta mallintaa jokaista elementtiä erikseen. Mallinnetut elementit näkyvät Application Modelerin reunassa olevassa Application Explorer puunäkymässä, josta näkyy elementtien järjestys mallinnettavan ohjelman sisällä. (RPA-kehittäjä 27.4.2020.)

Object Studiossa luodut actionit ovat kaikkien Process Studiossa luotujen prosessien käytettävissä. Tämä objektien jakaminen useiden prosessien kesken pienentää vaadittavien muutosten määrää, mikäli automatisoitavissa ohjelmissa tapahtuu jotain muutoksia. Muutokset voidaan tehdä objektiin, ja ne päivittyvät automaattisesti objektia käyttäviin prosesseihin. Blue Prismissä on myös ohjelman mukana tulevia oletusobjekteja, joilla voi automatisoida esimerkiksi Excelin tai muiden Office-ohjelmien käyttöä. (RPA-kehittäjä 27.4.2020.)

Process Studiossa robotti rakennetaan Object Studiossa luotuja toimintoja, eli actioneita, hyödyntämällä. Process Studiossa luotua robottia kutsutaan Blue Prismin termein prosessiksi. Prosesseja on tarkoitus ajaa Control Roomista, siitä lisää seuraavassa osiossa. Prosesseja pystyy myös kutsumaan toisista prosesseista, näitä kutsuttavia prosesseja kutsutaan aliprosesseiksi (Vivek 8.1.2019). Aliprosesseja käytetään tilanteissa, joissa jokin monen prosessin käytössä oleva samankaltainen toiminto halutaan helpommaksi ylläpitää. Aliprosesseihin pätee sama sääntö kuin objekteihin: jos aliprosessiin tekee muutoksia, ne päivittyvät kaikkiin aliprosessia käyttäviin prosesseihin. (RPA-prosessinomistaja 27.4.2020.)



Kuva 1. Kuvakaappaus Blue Prismin Process Studion käyttöliittymästä

Yritys X:n RPA-kehittäjän (RPA-kehittäjä 27.4.2020) mukaan **vaiheet**, joita on molemmissa studioissa käytettävissä laaja valikoima, ovat yksinkertaisuudessaan palasia ohjelmassa näkyvästä kaaviosta. Seuraavana lista käytetyimmistä ohjelman työkalupalkissa olevista vaiheista ja työkaluista:

- Link - työkalu, jonka avulla eri vaiheet liitetään toisiinsa
- Block - ruudukossa näkyvä alue, jonka alueelta poikkeukset otetaan kiinni
- Read - (Käytössä vain Object Studiossa) käytetään tiedon lukemiseen automatisoitavasta ohjelmasta
- Write - (Käytössä vain Object Studiossa) käytetään tiedon kirjoittamiseen automatisoitavaan ohjelmaan
- Navigate - (Käytössä vain Object Studiossa) käytetään automatisoitavassa ohjelmassa navigoimiseen
- Wait - (Käytössä vain Object Studiossa) käytetään odottamiseen. Tällä vaiheella on mahdollista odottaa automatisoitavan ohjelman tiettyä elementtiä tai odottaa tietty aika
- Code - käytetään koodin kirjoittamiseen
- Process - kutsutaan aliprosesseja
- Page - viitataan prosessissa tai objektissa oleviin välilehtiin
- Action - kutsutaan objektissa olevia toimintoja
- Decision - tehdään päätöksiä siitä, onko jokin tosi tai epätosi
- Choice - voidaan tehdä useita päätöksiä yhdenaikaisesti
- Calculation - käytetään laskutoimituksiin, tekstin muokkaamiseen ja tiedon tallentamiseen
- Multi Calculation - mahdollistaa useiden laskutoimitusten tekemisen
- Loop - toistorakenne, jolla käydään läpi taulukkotietoa
- Note - käytetään kommenttien lisäämiseen

- Anchor - simppeli ankkuripiste ruudussa näkyvälle viivalle. Käytetään selkeyttämistarkoituksessa
- End - kyseisen sivun lopetus. Lopetuksia voi olla yhdellä sivulla useita
- Start – kyseisen sivun aloitus. Aloituksia ei voi olla kuin yksi per sivu
- Exception - poikkeus, jonka tietoihin voidaan määritellä mikä poikkeus on kyseessä, tai hyödyntää jo olemassa olevaa poikkeustilaa
- Alert - (Käytössä vain Process Studiossa) tarkoituksena on ilmoittaa kontrollerin ruudulle prosessin tapahtumista
- Recover - otetaan kyseisen recoveryn alueella olevat poikkeukset kiinni
- Resume - jatkaa prosessin toimintaa virheenhallinnan jälkeen

Jokainen Blue Prismin vaihe on mahdollista saada näkymään ajolokille robotin ollessa käynnissä. Lokitus asetetaan päälle tai pois päältä kehitysvaiheessa, jokaisen vaiheen omista asetuksista. Blue Prismin (Blue Prism 2018, 5) oman ohjeistuksen mukaan mahdollisia asetuksia vaiheiden lokitukselle ovat:

- Enabled, eli vaihe näkyy ajolokilla.
- Disabled, eli vaihe ei näy ajolokilla.
- Errors Only, eli vain virheet näkyvät ajolokilla.

Objektin toimintoa kutsuessa on mahdollista laittaa toiminnon parametrilokitus pois päältä. Tällöin toiminnolle toimitettavat parametrit eivät näy ajolokilla. (Blue Prism 2018, 5.)

3.2 Control Room toiminnallisuus ja jonot Blue Prismissä

Control Room on Blue Prismin robottien ajamiseen, seuraamiseen ja aikatauluttamiseen käytettävä ominaisuus. Control Roomista käynnistetään robotti halutulla resurssilla haluttuun aikaan, kontrollerin asettamilla aloitusparametreilla (Wells 30.5.2019). Aikatauluun on mahdollista asettaa ajettavaksi useita prosesseja peräjälkeen (Learning Technologies 2018d). Tämä on kätevää esimerkiksi silloin, kun robotti muodostuu useammasta prosessista. Control Roomista näkee myös reaaliaikaisen tilanteen ajossa olevista prosesseista, resurssien tilasta ja jonojen tilanteesta. Tuotantoympäristössä lähtökohtaisesti vain kontrollerilla on valtuudet hallinnoida robotteja Control Roomin kautta. (Wells 30.5.2019.)

Jono on paikka, jossa kaikki robottien käsiteltävänä olevat tapaukset, eli itemit, ovat. Jono on nähtävissä Control Roomissa. Item on käytännössä rivi tietoa taulukosta. Kuvitellaan taulukko tietoa Excelissä, item on tällöin yksi rivi tästä taulukosta. Jokaisella itemillä on Item Key, eli ikään kuin itemin nimi, jonka perusteella eri itemit ovat erotettavissa toisistaan (Busy Ping 2017g). Jonosta näkyy myös itemin käsittelyn tila, joka voi olla jokin seuraavista: Pending, Locked, Completed tai Exception. Nämä ovat samassa järjestyksessä suomeksi: odottaa, lukittu, valmis tai poikkeus. Odottaa-tilaiset itemit odottavat käsittelyä, lukittuna olevat ovat käsiteltävänä, valmis-tilaisten käsittely on valmis ja poikkeus-tilaiset ovat käsittelyn jälkeen poikkeuksia joko liiketoiminnan määrittelyn perusteella tai järjestelmävirheen/prosessin sisäisen ongelman vuoksi. (Busy Ping 2017f.) Itemillä on kuitenkin

myös toinen tila. Tämä tila on tekstiä, jonka prosessi voi määrittää halutuksi (Busy Ping 2017g).

Jokaiselle jonossa olevalle itemille on mahdollista asettaa tageja eli merkkejä tai merkkijonoja, joiden perusteella kontrolleri pystyy erottamaan mitä tyyppiä kyseinen item on (Busy Ping 2017c; Ying 2018, luku Tagging the item). Myös prosessi pystyy hyödyntämään tagia toiminnassaan (RPA-kehittäjä 27.4.2020). Esimerkiksi Y-prosessi ottaa käsiteltäväkseen tietyllä tagilla olevat itemit tiettyyn kellonaikaan, ja jollain toisella tagilla olevat itemit toiseen kellonaikaan. Itemiä pystyy myös viivästämään prosessin toimesta. Item voidaan viivästyttää käsiteltäväksi vaikkapa viikon päästä (Learning Technologies 2018b). Jonossa olevalle itemille on myös mahdollista asettaa tietty prioriteetti. Tämä prioriteetti määrittää, minkä itemin prosessi ottaa seuraavana käsiteltäväksi. (Learning Technologies 2018e; Busy Ping 2017d.)

3.3 Muuttujat ja parametrit Blue Prismissä

Muuttuja on tietovarasto. Muuttujasta on mahdollista hakea tietoa ja sinne on mahdollista kirjoittaa tietoa. (Helsingin Yliopisto 2019.) Blue Prismissä muuttujaa kutsutaan Data Itemiksi. Muuttujan tietotyyppi voi olla Busy Pingin (Busy Ping 2017e) ja Vivek Goelin (VIVEK GOEL 2017) videoiden perusteella jokin seuraavista:

- Date, eli päivämäärä
- DateTime, eli päivämäärä ja aika
- Flag, eli tosi/epätosi arvo
- Number, eli numeraalinen arvo
- Password, eli salasana
- Text, eli tekstiä
- Time, eli aika
- TimeSpan, eli aikaväli
- Picture, eli kuva
- Binary, eli binääriluku

Blue Prismissä on tuki myös taulukkotiedolle. Tätä tietotyyppiä kutsutaan ohjelmassa Collectioniksi. Collectionissa jokaisella sen sarakkeella on jokin ylempänä mainituista Data Itemin tietotyypeistä. (Busy Ping 2017b.)

Busy Pingin (Busy Ping 2017a) videosta selviää, että muuttuja näkyy lähtökohtaisesti vain vaiheille, jotka ovat samalla sivulla muuttujan kanssa. Muuttuja on mahdollista asettaa myös näkymään kaikille prosessin/objektin välilehdille. Muuttujan näkyvyyden pystyy asettamaan myös siten, että kontrolleri pystyy muokkaamaan muuttujan arvoa Control Roomista robotin ollessa käynnissä (Learning Technologies 2018a; Busy Ping 2017a). On olemassa myös ympäristömuuttujia, joiden arvo on näkyvissä kaikille samassa tietokannassa

oleville prosesseille ja objekteille. Näitä muuttujia hallinnoidaan Blue Prismin System Managerissa, ei Process- tai Object Studiossa (Learning Technologies 2018c; Busy Ping 2017a).

Parametrit ovat muuttujia, eli tietoa, kuten tekstiä, numeraalisia arvoja tai tietotaulukoita. Monelle Blue Prismin käytössä olevalle toiminnolle voi antaa parametreja. Parametreja voi antaa esimerkiksi prosessin käynnistysvaiheessa, jolloin niitä kutsutaan Startup Parametreiksi. Parametreja voi hyödyntää myös prosessissa ja objektissa. (RPA-kehittäjä 27.4.2020.) Parametreilla pystyy esimerkiksi toimittamaan tietoa objektien toiminnoille, aliprosesseille tai muille välilehdille (Pro RPA 2018; RPA-kehittäjä 27.4.2020).

3.4 Blue Prismin sisäänrakennettu laadun seuranta

Blue Prismiin on sisäänrakennettu toimintoja tukemaan robottien laadun ja virheiden seurantaa. Ohjelma pystyy kertomaan yleisimmät virheet. Tällaisia virheitä ovat esimerkiksi jos jokin vaihe on unohtunut yhdistää kaaviossa eteenpäin, tai jollekin arvolle on määritelty tallennettava muuttuja, jota ei ole olemassa. Ohjelma antaa myös neuvoja, kuten antaa ilmi, jos jotain välilehteä ei käytetä ollenkaan. Tämän laadunseurannan pystyy määrittämään ohjelman asetuksista. Halutessaan ohjelman voi asettaa antamaan virheilmoituksen vaikkapa jonkin vaiheen kuvauksen puuttuessa. Tämä toiminnallisuus on rajattu siten, että perustason käyttäjät, kuten kehittäjät tai kontrollerit eivät pääse muokkaamaan sitä, minkä perusteella virheitä tai neuvoja tulee. (RPA-prosessinomistaja 27.4.2020.)

3.5 Poikkeukset Blue Prismissä

Blue Prismissä olevista ajonaikaisista virheistä on tyypillisesti käytössä kolmea eri tyyppiä (Tutorialspoint s.a.b). System Exception on automatisoitavassa ohjelmassa esiin tuleva virhe. Tämä virhe tulee tilanteessa, jossa ohjelma ei toimi odotetulla tavalla, esimerkiksi ohjelma jumittaa latausruutuun tai antaa virheilmoituksen. Internal Exception on Blue Prismin sisäinen virheilmoitus, jonka ohjelma luo automaattisesti. Tämä virhe tulee tilanteissa, joissa ohjelma ei esimerkiksi löydä jotain elementtiä tai ohjelma yrittää tehdä jotain mahdotonta, kuten jakaa sanaa luvulla. Business Exception on virheilmoitus, jota hyödynnetään tilanteissa, joissa liiketoiminta on määrittänyt prosessin logiikassa asian poikkeukseksi. Jos tapauksen ehto ei täyty, siitä tehdään Business Exception. (RPA-prosessinomistaja 27.4.2020.)

4 Teknisen laadun muodostuminen ohjelmistokehityksessä

”Ohjelmiston laatu on abstrakti käsite. Sen olemassaoloa voi olla vaikea määritellä, mutta sen puuttuminen on helposti havaittavissa” (XBOSoft s.a). Ohjelmistokehityksessä ohjelmiston laatu mittaa sitä, miten hyvin ohjelmisto on suunniteltu ja miten hyvin ohjelmisto vastaa kyseistä suunnittelua. (XBOSoft s.a.) ISO/IEC 25010 -ohjelmistolaatustandardin (ISO/IEC 25010 kohta 3.1) mukaan ohjelmiston laatu tarkoittaa sitä, missä määrin ohjelmisto täyttää sille sidosryhmien määrittämät vaatimukset ja tarjoaa siten arvoa.

4.1 Koodin laatu ja sen tekijät

Tekninen laatu ohjelmistokehityksessä muodostuu koodin laadusta. On lähtökohtaisesti vaikeaa sanoa mikä on huonoa koodia, ja onkin paljon helpompi sanoa, mikä on hyvää koodia (Helmkamp 2016). Richard Bellairs ohjelmistokehitystyökaluihin erikoistuneelta Perforcelta (Bellairs 11.7.2019) kuvailee hyvää koodia seuraavasti:

- Tekee mitä pitää
- Toimii johdonmukaisesti
- On helposti ymmärrettävää
- On hyvin dokumentoitua
- On mahdollista testata

Bryan Helmkamp Code Climatelta (Helmkamp 2016) taas kuvailee hyvää koodia seuraavalla tavalla:

- Yksinkertaista ja helppolukuista
- Hyvin testattua
- Vapaa virheistä
- Selkeää
- Uudelleenrakennettua
- Dokumentoitua
- Laajennettavissa olevaa
- Nopeaa

Koodin laatu on tärkeää, koska liiketoiminnat ja niiden prosessit perustuvat laajasti ohjelmistoihin. Ohjelmistojen laatu on tärkeää liiketoiminnan luotettavuuden ja kehittämisen kannalta. Codegrip sivustolla (Codegrip s.a.a) laadun kerrotaan tekevän koodista parempaa, sekä helpottavan luettavuutta ja muokkausta. Laadukas koodi tekee ohjelmasta kestävän ja helpottaa koodin siirrettävyyttä. Laadukas koodi myös vähentää teknistä velkaa. Koodin laatu on tärkeää senkin takia, että se parantaa itse ohjelmiston laatua. Laatu vaikuttaa myös koodin turvallisuuteen ja luotettavuuteen. Korkealaatuinen koodi on erityisen tärkeää tiimeille, jotka kehittävät turvallisuuskriittisiä sovelluksia. (Codegrip s.a.a.) Laadukas koodi helpottaa ohjelmiston jatkokehittämistä ja virheiden selvittämistä (Codegrip s.a.b). Tämän luvun alla esitellään ohjelmiston laadun mittarit, joiden jäsentämiseen on

hyödynnetty ISO/IEC 25010-standardin (ISO/IEC 25010 kohta 3.3) jäsentelyä. Monet mittareista ovat läheisesti riippuvaisia toisistaan tai liitonnaisia toisiinsa.

Luettavuus koodissa tulisi olla mahdollisimman helppoa. Koodin luettavuus helpottuu, kun koodi sisennetään oikein ja käytetään kommentteja. Itsestäänselvien asioiden kommentointi ei kuitenkaan ole suositeltavaa. (Sadangi 11.10.2017.)

Kommentit voivat selittää koodin tarkoitusta ja logiikkaa. Kommentit auttavat kehittäjää ymmärtämään esimerkiksi liiketoiminnan tarpeita, erikoistapauksia, jäljellä olevaa kehittämistä ja väliaikaisia ratkaisuja. Koodin ylläpidettävyys ja ongelmien ratkominen helpottuvat järkevän kommentoinnin avulla, sekä olemassaolevan koodin ymmärtämiseen menevä aika pienenee. Kehittäjä voi helposti eksyä koodiin, eikä ymmärrä sen tarkoitusta kommenttien puuttuessa. (Jackson s.a.)

Myös koodin dokumentointi on tärkeää luettavuuden kannalta. Koodin luettavuutta parantaa myös se, että nimeämistyylit koodissa ovat johdonmukaisia. Koodin luettavuus taas kärsii, mikäli koodissa on paljon toistoa. (Sadangi 11.10.2017.) Toistolla tarkoitetaan sitä, että kirjoitetaan jokin asia koodissa useaan kertaan, vaikka se olisi mahdollista toteuttaa toistorakenteella.

Koodin luettavuutta pystyy myös parantamaan tekemällä koodia kapeammaksi, sillä ihmisen silmät ovat tottuneet lukemaan tekstiä, joka on marginaalien sisällä. Samaa käyttötarkoitusta varten olevat väliaikaiset muuttujat olisi koodin luettavuuden kannalta hyvä nimetä johdonmukaisella tavalla. (Sadangi 11.10.2017.) Muiden palaute koodista on kuitenkin todellinen mittari koodin luettavuudelle. Koodikatselmointi luetaan yhdeksi parhaimmista keinoista saada palautetta koodin luettavuudesta. (Gergely 27.12.2019.)

Ylläpidettävyys kertoo yksinkertaisuudessaan sen, miten helppo koodia on ylläpitää. Koodin elinkaaren aikana siihen usein joudutaan tekemään muutoksia ja korjauksia. Ylläpidettävyyteen vaikuttaa useita tekijöitä, kuten koodin laajuus, johdonmukaisuus, rakenne ja kompleksisuus. Koodin ymmärrettävyys ja testattavuus ovat ylläpidettävän koodin aikaansaamiseen liittyviä tekijöitä. (Bellairs 11.7.2019.)

Ylläpidettävyyden mittaamiseen ei ole vain yhtä mittaria. Tapoihin parantaa koodin ylläpidettävyyttä kuuluvat esimerkiksi koodin tarkistusohjelmat, ja erityyppisten kompleksisuuden mittareiden hyödyntäminen kehityksessä. Ylläpidettävän koodin kehityksessä sekä ihmisen tekemät että automaattiset tarkistukset ovat tärkeitä. (Bellairs 11.7.2019.)

Modulaarisuus tarkoittaa koodin erottelua itsenäisiksi moduuleiksi, joilla on selkeät toiminnot. Nämä eri moduulit voivat olla vuorovaikutuksessa keskenään, mutta ne eivät tiedä toistensa sisäisiä funktioita tai muuttujia. Modulaarisuutta suositellaan, koska se pienentää koodin osien riippuvuutta toisistaan. Koodiin voi tehdä helpommin muutoksia ilman, että muutos vaikuttaa muualle koodiin. Ihannetilanteessa moduuli näkee toisesta moduulista vain mitä sinne pystyy antamaan syötteenä ja sen, mikä on moduulin ulostulo. (Takata 2013.)

Testattavuus mittaa sitä, miten hyvin ohjelma tukee testejä. Koodin manuaalinen tarkistus ei todennäköisesti löydä kaikkia virheitä, ja tästä syystä koodin tarkistamista ei suositella tehtäväksi pelkästään manuaalisesti. Testattavuus on riippuvainen siitä, miten hyvin automaattista testausta koodissa tuetaan. (Bellairs 11.7.2019.) Testattavuuden mittaustapoihin kuuluu muun muassa se, miten monta testitapausta tarvitaan virheiden löytämiseen järjestelmästä. Koodin koolla ja kompleksisuudella voi olla vaikutusta testattavuuteen, joten ottamalla esimerkiksi syklomaattinen kompleksisuus huomioon koodausvaiheessa, voidaan koodin testattavuutta parantaa. (Bellairs 11.7.2019.)

Syklomaattinen kompleksisuus on yksi koodin laadun mittareista. Syklomaattinen kompleksisuus kertoo koodissa olevien päätösten lukumäärän. Sitä kompleksisempaa koodi on, mitä suurempi luku on kyseessä. Syklomaattista kompleksisuutta voidaan käyttää kahdella tapaa: rajoittamaan koodin kompleksisuutta tai määrittämään vaadittavien testitapausten määrää. (Britton 22.10.2016.)

Siirrettävyys kertoo alustariippumattomuudesta, ja mittaa miten hyvin ohjelma toimii erilaisissa ympäristöissä ja alustoissa. Mitään mittaustapaa siirrettävyydelle ei ole, mutta siirrettävyyden parantamiseen on joitakin keinoja. Koodia kannattaa esimerkiksi testata useammilla alustoilla jo kehitysvaiheessa, ja on suositeltavaa asettaa ohjelmointiympäristön syntaksin tarkistus korkeimmalle asetukselle. Myös kahden ohjelmointiympäristön käyttöä suositellaan. Koodausstandardien noudattaminen kuuluu myös siirrettävyyttä parantaviin keinoihin. (Bellairs 11.7.2019.)

Luotettavuus on toimintojen suorittamista johdonmukaisesti ilman häiriöitä, kun ohjelma on käynnissä (Jayasinghe 16.7.2018). Riskien mahdollisuus, sekä ohjelman vakaus odottamattomissa tilanteissa ovat luotettavuuden mittareita. Datan korruptoituminen sekä virheet, jotka johtavat käyttökatkoksiin, ovat syitä joiden takia luotettavuutta mitataan ja monitoroidaan. (QISQ s.a.)

Ohjelmassa olevien virheiden määrä ja ohjelman saatavuus vaikuttavat siihen, miten todennäköisesti ohjelma toimii virheettä tietyn vaiheen yli. Tätä todennäköisyyttä käytetään ohjelman luotettavuuden mittaukseen. Staattisen analysoinnin työkaluilla on mahdollista mitata koodissa olevien virheiden määrää. Ohjelman saatavuutta taas pystyy mittaamaan MTBF-mittaustavalla, joka tulee englanninkielisestä ”mean time between failures” -käsitteestä. Tämä tarkoittaa yksinkertaisuudessaan sitä, miten paljon keskimäärin kuluu aikaa ajonaikaisten virheiden välillä. Luotettava koodi muodostuu mahdollisimman pienestä määrästä virheitä. (Bellairs 11.7.2019.)

”**Uudelleenkäytettävyys** tarkoittaa olemassa olevan koodin hyödyntämistä uuteen toimintoon tai ohjelmaan” (Bellairs 7.7.2017). Koodin on oltava laadukasta, eli tässä tapauksessa turvallista ja luotettavaa, jotta sitä voidaan käyttää uudelleen. Sellaisen koodin tekeminen, joka täyttäisi nämä vaatimukset, on haastavaa. Uudelleenkäytettäviä osien ja niiden rakenteiden tekeminen systemaattisesti on vielä haastavampaa. (Bellairs 7.7.2017.)

Olemassaolevan koodin uusiokäyttö on mahdollista, kun koodi on siirrettävissä eri laitteistolle, helposti jatkettavissa, eikä koodissa ole virheitä tai ongelmia, jotka voisivat vaikuttaa uuteen ohjelmaan. Todellisuudessa koodin uusiokäyttö ei ole kovin yleistä sen haastavuuden vuoksi. Kehittäjät päätyvätkin useimmiten tekemään ohjelmiston kokonaan alusta, tai pystyvät korkeintaan hyödyntämään olemassaolevasta koodista pienen osan uuteen ohjelmaan. (Bellairs 7.7.2017.)

Turvallisuus kertoo siitä, missä määrin järjestelmä suojaa tietoja siten, että henkilöillä, muilla tuotteilla, tai järjestelmillä, on niiden käyttöoikeustasoille sopiva tietojen saatavuusaste. Turvallisuuden määritelmiin kuuluu myös siirrossa oleva data. (ISO/IEC 25010 kohta 4.2.6.) Turvallisuus kertoo myös siitä, miten hyvin ohjelmisto on suojattu mahdollisilta hakkereilta. Turvallisuuden puute voi johtaa pahimmillaan pääsyn rajoituksiin, salaisen tiedon joutumisen vääriin käsiin, palvelun kaatumiseen, järjestelmien vaurioihin, jotka koskettavat lukemattomia ihmisiä, tai jopa kuolemaan. Koodin turvallisuutta voi parantaa käyttämällä turvallisia koodausstandardeja. (Bellairs 15.7.2019.) Koodin turvallisuuden parantamiseen on olemassa myös erityyppisiä koodin analysointityökaluja.

Tehokkuus kertoo siitä, miten hyvin koodi on optimoitu. Ohjelman optimoinnin tarkoituksia on useita. Tavoitteena voi olla esimerkiksi, että ohjelman koko pienenee, se vie vähemmän muistia, toimii nopeammin, tai välittää vähemmän parametreja. Optimoinnin perustana on, että optimoidulla ohjelmalla on tuotettava samat tulokset kuin sen optimoimattomalla versiolla. Kuitenkin, mikäli ohjelman käyttäytymisen muutosten todennäköiset seu-

raukset ovat pienempiä kuin optimoinnista saatu hyöty, tämä vaatimus voidaan jättää huomiotta. Optimointia voi tehdä optimointiohjelmilla tai ohjelmoijan toimesta. (PVS-Studio s.a.a.)

4.2 Keinoja parantaa koodin laatua

Koodin laadun analysointiin ja parantamiseen on useita keinoja, joista suosituimpia ovat koodin dynaaminen ja staattinen analysointi. Tässä luvussa kerrotaan tarkemmin näistä kahdesta analysointitavasta, sekä niiden vahvuuksista ja heikkouksista.

Staattinen analyysi on koodin analysointia, joka suoritetaan ilman ohjelman ajamista (Bellairs 2020). Staattisessa analyysissä tarkistetaan ohjelma mahdollisten ohjelmointivirheiden tai haavoittuvuuksien varalta. Staattisessa analyysissä tarkistetaan myös, että koodi on tehty useiden 4.1. luvun alla esiteltyjen laatumittareiden mukaisesti. Staattista analyysiä voidaan tehdä analysaattoreiden avulla, joiden vahvuuksiin kuuluu useiden koodausongelmien, kuten muistivuotojen ja tyhjien osoittimien löytäminen. Staattinen analyysaattori ei kuitenkaan löydä kaikkia virheitä, kuten esimerkiksi täyttääkö ohjelmisto sille asetetut vaatimukset, tai toimiiko jokin ohjelman funktio halutulla tavalla. (Bellairs 2018.)

Koodikatselmoinnit luokitellaan myös staattiseksi analyysiksi, sillä koodia ei lähtökohtaisesti ajeta koodikatselmoinnin aikana. Koodikatselmoinneissa ryhmä kehittäjiä tarkastelee koodin ulkoasua, mahdollisia virheitä tai haavoittuvuuksia ja sitä, tekeekö koodi sen, mitä sen kuuluukin (Codacy 14.3.2016). Katselmoinnissa tarkastetaan organisaation ohjelmointi- ja kehitysohjeiden noudattamista. Koodikatselmoinnissa tarkistellaan myös sitä, ovatko uudet automatisoidut testit riittäviä uuteen koodiin. Koodikatselmoinnit ovat myös hyviä paikkoja tiedonjakamiseen, sekä uusien kehittäjien koodiin perehdyttämiseen ja mentorointiin. Koodikatselmoinnit helpottavat myös aika- ja työmääräarvioiden tekemistä, sekä mahdollistavat useiden kehittäjien ymmärryksen koodista. Tämä taas vähentää riippuvuutta tietyistä kehittäjistä ja mahdollisesti helpottaa kehittäjien siirtymistä uusien projektien pariin. (Radigan s.a.)

Dynaaminen analyysi noudattaa päinvastaista lähestymistapaa kuin staattinen analyysi ja suoritetaankin ohjelmaa ajettaessa. Monissa kehitysympäristöissä on sisäänrakennettu tuki dynaamiselle analyysille. Dynaaminen analyysi suoritetaan toimittamalla analysoitavalle ohjelmalle tai ohjelman osalle testattava syöte, ja tämän vuoksi dynaamisen analysoinnin tehokkuus onkin pitkälti kiinni testattavan datan määrästä. Dynaaminen analysointi kertoo meille, miten paljon ohjelma vaatii resursseja, syklomaattisen kompleksisuuden, sekä joitain ohjelmointivirheitä ja haavoittuvuuksia. Dynaaminen analysointi on tärkeää,

mikäli ohjelman luotettavuus, responsiivisuus ja resurssivaatimukset koetaan tärkeäksi. (PVS-Studio s.a.b.)

Dynaamisen analyysin vahvuuksiin kuuluu toimivuus ilman lähdekoodia, eli testata voi siis myös ohjelman yksittäisiä osia. Toinen vahvuus on se, että useimmissa tapauksissa virheiden läpipääsy on mahdotonta, sillä analysointi ei perustu siihen, voiko jokin virhe tapahtua, vaan siihen, tapahtuuko se. Dynaamisella analyysillä on myös huonot puolensa. Se löytää virheet vain polulta, jota käytettävä testidata kulkee, eikä se pysty myöskään kertomaan tekeekö koodi sen, mitä sen pitäisi tehdä. Dynaamiseen analyysiin kuuluva ohjelmien testaaminen vaatii myös paljon resursseja, eikä testata voi kuin yhden polun kerrallaan. Loppujen lopuksi parasta on analysoida koodi sekä dynaamisella, että staattisella analyysillä. (PVS-Studio s.a.b.)

Helmkampin (Helmkamp 2016) mukaan yksi keinoista parantaa koodin laatua on tehdä pieniä muutoksia jatkuvasti ja välttää suurien muutosten tekemistä. Pienissä muutoksissa mahdollisten virheiden huomaaminen on helpompaa. Mitä suurempi muutos ja mitä suurempi muutetun koodin määrä, sitä vaikeampaa koodista on huomata virheitä. Tämä sääntö pätee eritoten teknisen velan pienentämisessä, jossa vanhaa koodia rakennetaan uudelleen nykyisten standardien mukaiseksi. (Helmkamp 2016.)

Selkeät vaatimukset ovat myös tärkeä osa laadukkaan ohjelmiston aikaansaamista. Projektit, joissa on selkeät dokumentit ja määrittelyt, joiden mukaan kehitystä tehdään, saavat paljon todennäköisemmin aikaan laadukasta ohjelmistoa kuin projektit, joiden dokumentit ja määrittelyt ovat laadultaan heikompia ja epäselviä. (Sealights s.a.)

4.3 Tekninen velka

Tekninen velka tarkoittaa koodiin sen kehityksen elinkaaren myötä syntynyttä kehitys-, korjaus- ja uusimistarvetta. Kehityksen aikana saatetaan käytettävissä olevan rajallisen tiedon tai esimerkiksi kehityksen aikataulupaineiden takia joutua tekemään ratkaisuja, jotka eivät ole tehokkaita pitkällä tähtäimellä. Voi myös olla, että tiettynä ajankohtana ja tiettyyn tarpeeseen tehty ratkaisu on silloin ollut hyvä, mutta vaatimukset, tekniset mahdollisuudet ja tapa jäsentää hyvää koodia muuttuvat. (Tietohallintojohtaja 7.5.2020.)

Teknistä velkaa koskevassa tutkimuksessaan Johannes Holvitie ja muut tutkimuksen tekijät (Holvitie ym. 2017) kuvaavat teknistä velkaa seuraavasti: ”Tekninen velka kuvaa seurausta niistä ohjelmistokehitystoimenpiteistä, kun tarkoituksella tai epähuomiossa priori-

soidaan asiakkaan arvoa ja/tai projektin rajoituksia, kuten toimitusaikatauluja, teknisen implementoinnin ja tarkan suunnittelun sijaan". Yrityksen ohjelmistokehityksen näkökulmasta teknisen velan jatkuva ymmärtäminen ja vähentäminen on tärkeää, koska ohjelmistot ovat tyypillisesti käytössä pitkään.

4.4 Ketterä kehitys

Scrum Masterina työskentelevän Joonas Kosken (Koski s.a.) mukaan ketterän ohjelmistokehityksen menetelmiä hyödynnetään IT-projektien läpiviemiseksi, ja kirjoittaa ketterän kehittämisen olevan vaihtoehto perinteiselle vesiputousmallille. IT-projektin tarpeiden määrittely tarkentuu ketterässä kehityksessä sprinteissä. Myös ketterän kehityksen ja vesiputousmallin yhdistelmä on nykypäivänä yleinen. (Koski s.a.)

Ketterä kehitys pohjautuu 17 merkittävän ketterän ohjelmistokehityksen puolestapuhujan (Beckm ym. 2001a) yhteiseen manifestiin. Manifesti on seuraavanlainen:

> "Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:
> **Yksilöitä ja kanssakäymistä** enemmän kuin menetelmiä ja työkaluja
> **Toimivaa ohjelmistoa** enemmän kuin kattavaa dokumentaatiota
> **Asiakasyhteistyötä** enemmän kuin sopimusneuvotteluja
> **Vastaamista muutokseen** enemmän kuin pitäytymistä suunnitelmassa
> Jälkimmäisilläkin asioilla on arvoa, mutta
> arvostamme ensiksi mainittuja enemmän."

Koski (Koski s.a.) on suomentanut manifestiin pohjautuvat ketterän kehityksen 12 periaatetta (Beckm ym. 2001b) seuraavasti:

- Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
- Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.
- Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.
- Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä koko projektin ajan.
- Rakennamme projektit motivoituneiden ihmisten ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat, ja luotamme siihen, että he saavat työn tehtyä.
- Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.
- Toimiva ohjelmisto on edistymisen ensisijainen mittari.
- Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.
- Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
- Yksinkertaisuus – tekemättä jätettävän työn maksimointi – on oleellista.
- Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.
- Tiimi tarkastelee säännöllisesti omaa tehokkuuttaan ja mukauttaa toimintaansa sen mukaisesti

5 Yritys X:n konsernin ohjeistusten arviointi yritys X:n näkökulmasta

Tässä luvussa esitellään yritys X:n konsernin RPA-ohjeistuksia ja arvioidaan niiden soveltuvuutta yritys X:lle. Konsernin ohjeistuksia arvioidaan yritys X:n RPA-tiimin työntekijöille suorittujen haastattelujen, sekä ohjelmistokehityksen teknisen laadun teorian avulla.

5.1 Yritys X:n konsernin robotiikan ohjeistukset ja vaatimukset

Tässä osiossa käymme läpi konsernin ohjeistuksia, ohjeistuksista kerrotaan keskeiset asiat. Ohjeistuksen yksityiskohtia ei käydä läpi materiaalin luottamuksellisuuden vuoksi. Aineisto on kerätty yritys X:n sisäisessä verkossa olevista yritys X:n konsernin CoE:n ohjeistuksista. Ohjeistuksista löytyy ylätason kuvaus robotiikkakehityksestä, sekä runsaasti osioittain jaoteltua yksityiskohtaista ohjeistusta. Ohjeistuksesta on pyritty poimimaan tämän työn kannalta oleellinen materiaali ohjelmistorobotiikan teknistä kehitystä silmällä pitäen.

Ensin käymme läpi RPA-projektin vaiheet. Vaiheiden esittelemiseen hyödynnämme yritys X:n konsernin RPA-operointimallia (liite 2). Operointimallista selviää, että RPA-projektilla on viisi vaihetta. Vaiheet ovat Assess, Prepare, Configure, Stabilize ja Operations. Vaiheiden aloitukseen ja niistä eteenpäin siirtymiseen liittyy päätöspisteet ja niiden päätöskriteerit.

Assess-vaiheessa RPA-idea vastaanotetaan liiketoiminnalta. Analyytikko selvittää, onko idea toteuttamiskelpoinen ja priorisoi tapauksen kehitystilanteen mukaisesti. Analyytikko selvittää myös, onko liiketoiminnalta saatavilla riittäviä tukiresursseja projektin ajaksi. Assess-vaiheen lopuksi tehdään päätös (D0) RPA-projektin aloittamisesta.

Prepare-vaiheessa analyytikko luo liiketoiminnan prosessikuvauksen avulla vaadittavat dokumentit, joihin kuuluvat vaihe-vaiheelta kuvaus, robotin riskianalyysi ja henkilötietojen käsittelyyn liittyvä arviointi. Analyytikko luo myös liiketoiminnan puolelta tulevan aihealueen asiantuntijan kanssa mahdolliset testiskenaariot. Analyytikko tilaa tässä vaiheessa myös tarvittavat järjestelmävaltuudet kehittäjille. Kontrollerin tehtäviin prepare-vaiheessa kuuluu lisenssien ja resurssien riittävyyden tarkistaminen. Prepare-vaiheen lopuksi tehdään päätös (D1) robotin konfiguroinnin aloittamisesta.

Configure-vaiheessa tapahtuu robotin konfigurointi. Konfigurointi-vaihe aloitetaan kick-off tapaamisella, jonka jälkeen kehittäjä/kehittäjät suunnittelevat, miten robotti aiotaan rakentaa. Tässä vaiheessa aletaan kirjoittaa robotille käyttöohjetta. Seuraavaksi pidetään ensimmäinen Design Board -tapaaminen, jonka aikana suunnitelma esitellään Design Board

-ryhmälle, joka kertoo oman näkemyksensä asiaan ja antaa mahdollisia vinkkejä kehityksen toteuttamiseen. Design Board on konsernin CoE:n laadunvarmistustaho, joka seuraa, että annettuja ohjeistuksia noudatetaan ja että kehitys on laadukasta. Tapaamisia Design Boardin kanssa on tarkoitus pitää säännöllisesti koko kehityksen ajan. Kehitys tehdään testiympäristössä siinä määrin, kuin ohjelmistojen testiversioita on saatavilla. Kehittämisen jälkeen robotti testataan testiympäristössä testidokumentaatiossa määritellyillä testattavilla tapauksilla. Mikäli testi onnistuu, seuraavaksi järjestetään tapaaminen Design Boardin kanssa. Tapaamisen tarkoituksena on saada hyväksyntä robotin testaamiselle tuotantoympäristön datalla, sekä saada lupa tuotantoympäristöön viemiselle. Tapaamisessa itsessään ei kuitenkaan tätä hyväksyntää anneta, vaan robotti lähetetään tapaamisen jälkeen Design Boardille syvällistä tarkastelua varten. Jokaisen robotin osan, eli prosessien ja objektien, on oltava kaikilta osiltaan ohjeistusten mukaisia. Design Board dokumentoi mahdolliset virheet, ja robotti korjataan näiden virheiden osalta. Näiden korjausten jälkeen testit suoritetaan uudelleen, ja robotti lähetetään jälleen Design Boardin arvioitavaksi. Kun robotti on lopulta saanut hyväksynnän, ajetaan robotti tuotantodataa sisältävässä ympäristössä aidoilla tapauksilla. Mikäli tämäkin ajo menee onnistuneesti, antaa liiketoiminta robotille hyväksynnän oikenlaisesta toiminnasta. Jotta robotti viimein saadaan tuotantoympäristöön, robotti menee muutostikettiprosessin läpi, jonka avulla varmistetaan, että kaikki vaadittava dokumentaatio on olemassa ja hyväksyttynä vaadittavien henkilöiden osalta. Tämä prosessi on viimeinen portti (G1) tuotantoympäristöön.

Stabilize-vaiheessa robotin vastuu siirtyy kehittäjältä kontrollerille. Robotti ajetaan kontrolloidusti tuotantoympäristössä ja varmistetaan, että se toimii odotetulla tavalla. Jos robotti toimii odotetulla tavalla, liiketoiminta antaa hyväksynnän robotin toiminnalle, ja tehdään päätös (D2) robotin Operations-vaiheeseen siirtymisestä. Jos robotti ei toimi odotetulla tavalla, siihen tehdään tarvittavat korjaukset ja suoritetaan uudelleen tässä vaiheessa esitellyt toimenpiteet.

Operations-vaihe alkaa sillä, että robotti saavuttaa täyden käyttöasteensa ja alkaa tuottamaan hyötyjä. Tässä vaiheessa tarkastellaan vaatiiko robotti huoltoa tai muutoksia. Huollot ja muutokset menevät normaalin priorisointiprosessin mukaisesti, sekä vaativat aina omat muutostikettinsä. Mikäli muutoksessa tehdään muutoksia liiketoiminnalliseen logiikkaan, tai muutetaan prosessia laajasti teknisessä mielessä, se vaatii kaikkien esiteltyjen vaiheiden läpikäynnin. Tämän muutoksen ajasta pelkästään muutostiketti vie kaikkinensa noin kolme päivää. Mikäli taasen muutos tai korjaus on pieni, voidaan tehdä pienempi tiketti, jonka saa vietyä alusta loppuun alle päivässä, eikä vaadi edellämainittujen vaiheiden läpikäyntiä. Operations-vaiheeseen kuuluu myös päätös (D3) siitä, pitäisikö robotti poistaa käytöstä.

Tässä työssä keskitymme RPA-projektin Configure-vaiheeseen, sen koskettaessa nimenomaan ohjelmistorobotin teknistä kehitystä. Seuraavana käymme läpi konsernin teknisen kehityksen ohjeistusta.

Nimeämistyylit konsernin ohjeistuksessa

Nimeämistyyleissä otetaan kantaa erityyppisten Data Itemien ja Collectioneiden nimeämisiin. Objekteissa kantaa otetaan itse objektien, toimintojen, parametrien sekä elementtien nimeämiseen. Elementtien nimeämisessä pyydetään ottamaan huomioon mahdolliset dynaamiset elementit ja mallinnustapa. Prosessien nimeämisissä otetaan kantaa itse prosessien, aliprosessien, välilehtien ja jonojen nimeämiseen. Kantaa otetaan myös erityyppisten poikkeuksien nimeämiseen.

Objektit konsernin ohjeistuksessa

Objektit suositellaan pilkkomaan mahdollisimman pieniksi palasiksi. Ohjeissa määritellään, mitä objektin toiminnon sisällä saa, tai ei saa olla. Toiminto ei ohjeiden mukaan lähtökohtaisesti saisi tehdä kuin yhden asian. Kantaa otetaan myös siihen, miten objekti toimittaa tai sille toimitetaan parametreja. Ohjeistuksissa parametrien ja toimintojen kuvaukset ovat pakollisia. Ohjeistuksissa otetaan kantaa Data Itemien oletusarvoihin, sekä muihin kuin oletusobjekteihin viittaamiseen objektin sisältä. Toiminnoista kantaa otetaan toiminnon aloitukseen, lopetukseen ja elementtien odottamiseen. Odotusvaiheissa otetaan kantaa niiden toimintaan eri mallinnustavoilla mallinnettujen elementtien kanssa. Objekteissa otetaan kantaa myös Data Itemien näkyvyyteen, sekä annetaan erityisohjeita tiettyjen navigointitoimintojen käyttöön. Kantaa otetaan myös kuvan perusteella toimivaan Sufrace Automationiin ja sen toimintatapoihin.

Ohjeistuksissa neuvotaan ottamaan toimintojen tekemisessä uudelleenkäytettävyys huomioon. Ohjeissa kerrotaan monipuolisesti uudelleenkäytettävyydestä, sekä toimintatavoista, joilla uudelleenkäytettävyyttä voidaan parantaa. Objektiin liittyvien ohjeistuksien lopuksi otetaan kantaa elementtien attribuuttien arvoihin.

Prosessit konsernin ohjeistuksessa

Prosessien rakentamiseen neuvotaan käytettäväksi valmista mallia, jota muokataan tilanteen mukaisesti. Mallissa on valmiina logiikka itemien poimimiseksi jonosta ja niiden merkkaamiseksi valmiiksi tai poikkeukseksi. Jononlatauksen ja käsittelevän osuuden erotteluun on otettu kantaa. Prosessia suositellaan pilkottavaksi useille välilehdille. Ohjeissa otetaan kantaa eri poikkeustilanteiden kanssa toimimiseen prosessissa. Ohjeissa kerrotaan myös, miten prosessin tulisi toimia kaatumistilanteessa.

Jonot konsernin ohjeistuksessa

Jonojen ohjeistuksissa otetaan kantaa jonojen salaukseen, sekä itemin jonossa näkyvän Item Keyn arvoon. Ohjeistuksissa neuvotaan huomioimaan prosessissa tapahtuvat asiat, jotka voivat vaikuttaa jonoon päätyvään dataan. Ohjeissa mainitaan myös mahdollisuus tagien käyttämiseen.

Lokitukset konsernin ohjeistuksessa

Lokitusohjeistusten tarkoitus on ottaa huomioon asiakassalaisuus lokituksessa. Ohjeistuksissa otetaan kantaa actionin, decisionin, choicen, sivun alotuksen ja lopetuksen, calculationin, exceptionin, recoveryn ja resumen lokituksiin. Ohjeissa neuvotaan, miten lokitukset saa säädettyä kohdilleen. Ohjeissa kerrotaan myös, mitkä ovat minimilokitusasetukset kullekin vaiheelle virheiden selvityksen kannalta. Myöskin toimintojen parametrilokitukseen otetaan kantaa. Viimeisenä lokitusohjeistuksessa neuvotaan, miten toimitaan mahdollisen asiakasdatan kanssa Application Modellerin attribuuteissa. Lokituksien hallintaan on luotu myös työkalu, joka tekee edellämainitut lokitusmuutokset automaattisesti.

Tyylit ja asettelut konsernin ohjeistuksessa

Tyylien ja asettelujen ohjeistuksissa otetaan kantaa objektien ja prosessien eri komponenttien kuvauksiin. Ohjeissa otetaan kantaa myös Blue Prismin oman laadunseurannan antamiin virheisiin. Mainintoja on myös prosessiflown siisteydestä ja ulkonäöstä. Ohjeistuksissa neuvotaan myös selittämään sellaisten odotusvaiheiden tarkoitusta, jotka eivät odota mitään elementtiä. Ympäristömuuttujien käytöstä on myös kannanotto.

Alan parhaat käytänteet konsernin ohjeistuksessa

Ohjeistuksissa kerrotaan myös niin sanottuja yleisiä parhaita käytänteitä robotin kehitykseen. Näissä ohjeissa otetaan kantaa Data Itemien ryhmittelyyn, sekä niiden näkyvyyksiin. Hyvissä käytänteissä on vielä erikseen mainintaa eri näkyvyyksillä olevien Data Itemien nimeämisestä sekä Data Itemien ja Collectioneiden studiossa näkyvistä ko'oista. Ohjeistuksissa on myös yleisiä ohjeita, joiden kerrotaan olevan vain suosituksia. Suositukset koskevat oletusarvoja, tekstidataa ja Data Itemin salasana-tietotyyppiä. Ohjeistuksissa otetaan kantaa myös oletusobjekteina toimivien objektien tuotantoympäristöön vientiin ja niiden rajoituksiin.

Parhaissa käytänteissä otetaan kantaa myös tilanteisiin, joissa elementtiä odottava odotusvaihe ei löydä odotettavaa elementtiä. Kannanotto löytyy myös uudelleenyrittämislogiikkaan, ohjelmien sammuttamiseen, sekä olemassaolevan poikkeustiedon eteenpäin toi-

mittamiseen. Ohjeissa neuvotaan myös kuvaamaan tehdyt muutokset tallennuksen yhteydessä muutosten seuraamista helpottamaan. Konsernin ohjeistuksissa on myös määrittely sille, miten hyödyt raportoidaan RPA-prosesseissa.

Robottien testaus konsernin ohjeistuksessa

Testausohjeistuksissa kerrotaan, miten testaus pitää tehdä ja mitä siinä pitää ottaa huomioon. Ohjeissa neuvotaan erityyppisten testitapausten huomioimista testausvaiheessa. Ohjeissa annetaan käsitystä siitä, miten kauan testeihin yleensä menee aikaa. Testausohjeissa neuvotaan, miten voidaan testata robotin toimivuutta useammalla tietokoneella samanaikaisesti.

Vertaisarviointi

Vertaisarviointi vaaditaan kaikissa muutoksissa. Kuittaus vaaditaan vertaisarvioinnin tekijältä pienen muutoksen tuotantoon viennissä. Pienissä muutoksissa vertaisarvioinnin tekee kollega, ja suuremmissa muutoksissa vertausarviointi tapahtuu Design Board -ryhmässä. Vertaisarvioinnissa pyydetään kiinnittämään huomiota uudelleenkäytettävyyteen, tyyliin, parhaiden käytäntöjen noudattamiseen, virheellisen, päällekkäisen tai käyttämättömän tiedon hävittämiseen, teknisen velan vähentämiseen, sekä sen varmistamiseen, että tehdyt muutokset ovat tehty tilatulla tavalla.

Muutosten vienti tuotantoympäristöön konsernin ohjeistuksessa

Jotta robotti voidaan viedä tuotantoympäristöön, kaikki edellä mainittu, lukuun ottamatta suosittelevia ohjeita, on oltava kunnossa. Design Board -tarkistus vaaditaan kaikille suuremmille muutoksille, kuten on mainittu Operations-vaiheen selostuksessa. Kehitystyön selkeyttämiseksi ohjeistuksissa on saatavilla lista, jossa luetellaan kaikki kohdat, jotka pitää olla kunnossa Design Board -hyväksynnän saamiseksi.

Vaatimuksiin kuuluu myös kaikenkokoisissa muutoksissa muutoksen koon mukaisen tarkistuslistan täyttäminen. Pienien muutosten tarkistuslistassa vaaditaan selostus tehdyistä muutoksista, kuittaus vertaisarvioinnin suorittamisesta, kuittaus muun kuin muutoksen tekijän lokitustarkistuksesta, sekä kehittäjän oma kuittaus siitä, että muutos on tehty ohjeistusten mukaisesti. Suuremmissa muutoksissa tarkistuslistassa vaaditaan vielä edellämainittujen lisäksi liiketoiminnan kuittaukset robotin toimivuudesta.

5.2 RPA yritys X:ssä

Tässä osiossa selvitetään haastattelujen avulla millaista RPA on yritys X:ssä, millaisia kokemuksia RPA:n kanssa työskentelevillä työntekijöillä on konsernin RPA-ohjeistuksista,

sekä selvitetään myös mihin suuntaan RPA:ta halutaan yrityksessä viedä. Näiden vastausten pohjalta, tietoperustassa esiteltyjä ohjelmistorobotiikan laadun mittareita hyödyntäen tapahtuu konsernin ohjeiden arviointi yritys X:n näkökannalta.

5.2.1 Tutkimuksen toteutus ja tutkimusmenetelmät

Tutkimus tehtiin laadullisena tutkimuksena. Laadulliseen tutkimukseen päädyin aiheen vähäisen olemassaolevan materiaalin vuoksi. Määrällinen tutkimustapa ei ollut mahdollinen, sillä ohjelmistorobotiikan teknisestä kehityksestä henkivakuutusyhtiössä ei ollut saatavilla minkäänlaista tilastodataa. Myös tutkimuskysymykset muodostuivat laadulliseen tutkimukseen soveltuviksi. Laadullinen tutkimus sopii tällaiseen tilanteeseen, jossa halutaan korostaa aineiston keruuta todellisessa ympäristössä, tässä tapauksessa yritys X:n sisällä.

Koska analysointivaiheessa hyödynnetään sekä tietoperustaa että aineistoa, analysointi on teoriaohjaavaa sisällönanalyysiä. Aineiston ja teorian suhde on siis abduktiivinen, eli työn lopputulos muodostetaan sekä aineiston että tietoperustan avulla.

5.2.2 Aineiston hankintamenetelmä

Aineiston hankintamenetelmäksi valikoitui haastattelu aluksi suunnittelemani havainnoinnin sijaan. Kyseessä on puolistrukturoitu haastattelu, jossa oli kuusi kysymystä kaikille haastateltaville, sekä kaksi lisäkysymystä tietohallintojohtajalle. Päädyin valitsemaan haastattelutavaksi sähköpostihaastattelun vallitsevan koronatilanteen vuoksi. Sähköpostihaastattelua puolsi myös tiukka aikataulu, joka olisi ollut entistä tiukempi ilman aineiston helppoa koontia sähköposteista.

Haastateltavat valittiin heidän kokemuksensa ja työtehtävien RPA-liitännäisyyden perusteella. Jokainen haastateltava työskentelee RPA:n parissa yritys X:ssä, jokainen eri tehtävässä ja näin ollen omaavat paljon tietoa, näkemyksiä ja kokemusta RPA:sta yritys X:ssä. Itseni lisäksi yrityksessä RPA:n parissa ei työskentele kokopäiväisesti muita kuin haastatteluun valitut henkilöt, eli saturaatiota ei olisi ollut mahdollista saada tämän enempää.

Haastateltavana olivat seuraavat henkilöt:

- RPA-kehittäjä, yritys X
- RPA-kontrolleri/RPA-analyytikko, yritys X
- RPA-prosessinomistaja, yritys X
- Tietohallintojohtaja, yritys X

Haastattelukysymykset muodostin sen perusteella, että konsernin ohjeistuksia voitaisiin arvioida yritys X:n kannalta haastattelujen vastauksien pohjalta. Kysymykset hakevat kuvaa siitä, millaista robotiikkatekeminen on yritys X:ssä, sekä mitä hyviä tai huonoja puolia konsernin ohjeistuksessa on havaittu yritys X:n kannalta. Hain vastausta myös siihen, miten yritys X:n ohjelmistokehitys toteutetaan tällä hetkellä. Vastaus tähän kysymykseen helpottaisi yhteneväisyyksien hakemista ohjelmistokehityksen ja robotiikkatekemisen välillä.

Haastattelukysymykset olivat seuraavanlaiset:

1. Miten kuvailisit robotiikkatekemistä yritys X:ssä?
2. Mikä on yleinen kuvasi konsernin ohjeistuksista, niiden selkeydestä ja hyödyllisyydestä?
3. Mitkä asiat konsernin ohjeistuksessa nopeuttavat/hidastavat asioita?
4. Mitä konsernin ohjeistuksen osuuksia on helppo seurata/toteuttaa? Entä vaikea seurata/toteuttaa?
5. Onko yritys X:n robotin kehityksessä ja tuotantoon viennissä osuuksia, joihin konsernin ohjeistuksesta ei ole apua?
6. Onko yritys X:llä omia ohjeita/tehtäviä/vaiheita konsernin ohjeistuksen lisäksi tai niitä täydentämässä?

Tietohallintojohtajalle oli kaksi lisäkysymystä:

7. Miten yritys X toteuttaa tällä hetkellä ohjelmistokehitystä?
8. Mikä on yritys X:n näkemys robotiikan hyödyntämisestä?

5.2.3 Haastattelun tulokset

Haastattelujen vastauksista (liite 1) pyrin löytämään toistuvia asioita, haastateltavien korostamia asioita, sekä ristiriitoja tai erimielisyyksiä.

Haastatteluista nousee esiin seuraavat neljä asiaa:

1. Robotiikkatekeminen yritys X:ssä on ketterää, nopeaa ja läheistä liiketoiminnan kanssa.
2. Yleisesti, konsernin ohjeet ovat hyviä ja kattavia, mutta vaadittavat tarkistukset ja hyväksynnät koetaan raskaiksi ketterää toimintatapaa hyödyntävässä yritys X:ssä.
3. Konsernin ohjeessa olevat teknisen kehityksen ohjeistukset koetaan hyödyllisiksi muotoseikkoja lukuun ottamatta.
4. Yritys X:llä on omia käytänteitä joko konsernin ohjeistuksen lisäksi, tai niitä korvaamassa.

Vastauksissa nostetaan paljon esille tarkastus- ja hyväksymisprosesseihin menevää aikaa, mittakaavaeroja ja ohjeistuksien löytämisen vaikeutta. Minkäänlaisia ristiriitoja tai erimielisyyksiä haastatteluissa ei ilmennyt.

Tietohallintojohtajalta saaduista vastauksista selvisi, että yritys hyödyntää myös ohjelmistokehityksessään ketterän kehityksen periaatteita. Vastauksista selvisi myös, että RPA on

yritys X:n näkökulmasta ketterä transitioteknologia, jolla tarjotaan nopeaa apua liiketoimintaprosessien manuaalityötä vaativiin osiin. Myös perusjärjestelmien kehitystä ja toiminnallisuutta tuetaan ohjelmistorobottien avulla.

5.3 Konsernin ohjeistusten tarkastelu teoriaohjaavan sisällönanalyysin avulla

Tässä osiossa arvioimme konsernin ohjeistuksia yritys X:n näkökannalta, hyödyntäen tietoperustassa esitettyjä asioita. Analyysi on teoriaohjaavaa sisällönanalyysiä. Teoriaohjaavassa sisällönanalyysissä tutkijan ajattelua ohjaavat sekä teoria että aineisto (Leinonen 12.12.2018). Muita vaihtoehtoja analyysille olisivat olleet aineistolähtöinen tai teorialähtöinen sisällönanalyysi. Kumpikaan edellä mainituista ei kuitenkaan sovi tähän työhön yhtä hyvin kuin teoriaohjaava sisällönanalyysi, sillä työssä hyödynnetään sekä tietoperustaa että aineistoa analyysin tekemiseen. Myös lähtötilanne, eli konsernin ohjeistukset, ovat olennainen osa työtä. Analyysi tehdään haastatteluista saatujen tietojen, sekä tietoperustassa esiteltyjen asioiden avulla. Analyysiä aletaan tekemään tutkimalla toteutuvatko ohjelmiston laadun teoriassa esitellyt laadun tekijät nykyisessä ohjeistuksessa. Teemat ovat jaettu tietoperustassa esiteltyjen laadun tekijöiden mukaisesti. Tämän jälkeen arvioidaan, mitkä ohjeistuksista ovat hyödyllisiä yritys X:n kannalta ja mitkä eivät ole. Tähän arviointiin hyödynnetään haastattelujen tuloksia.

Ohjeistusten arviointi ohjelmiston laadun mittareiden avulla

Aloitetaan analysointi **luettavuudesta**. Konsernin ohjeistuksissa nimeämiskäytänteet on kerrottu selkeästi, ne ottavat erilaiset tilanteet huomioon ja ovat pääosin johdonmukaisia. Epäjohdonmukaista on se, että Data Itemien nimet pyydetään kirjoittamaan pienellä, vaikka Blue Prism -työkalu luo ne oletusarvollisesti alkamaan suurilla alkukirjaimilla.

Blue Prismissä kommentteja pystyy tallentamaan studioon Note-työkalun avulla, mutta kommenttien käytöstä ei konsernin ohjeistuksessa puhuta lainkaan. Sen sijaan ohjeistuksissa vaaditaan asettamaan kuvaus jokaiselle parametrille, sivulle ja toiminnolle. Kuvauksen lisääminen on lähtökohtaisesti hyvä tavoite, mutta kuten tietoperustassa on kerrottu, itsestäänselvien asioiden kommentointi ei ole suositeltavaa. Mikäli parametrin nimi tai toiminnon nimi on itseään kuvaava, voidaan kyseenalaistaa tällaisen ohjeistuksen hyödyllisyyttä. Asetteluun ja ulkonäköön liittyvät ohjeistukset ovat samoilla linjoilla tietoperustassa esiteltyjen asioiden kanssa. Ohjeistuksissa pyydetään myös välttämään turhaa toistoa ja hyödyntämään toistorakenteita, mikä on myös samoilla linjoilla tietoperustan kanssa.

Ylläpidettävyydestä ohjeistuksissa mainitaan samoja asioita, kuin tietoperustassa esitellyssä teoriassa. Turhaa kompleksisuutta ja laajuutta pyydetään välttämään. Ohjeissa

opastetaan myös Blue Prismin sisäänrakennetun laadunseurannan hyödyntämistä, sekä vaaditaan tuotantoympäristöön menevän robotin tarkistuksen myös toiselta kehittäjältä.

Modulaarisuus otetaan huomioon konsernin ohjeistuksessa useassa kohdassa. Linja on kokonaisuudessaan täysin yhtenevä tietoperustassa esitellyn kanssa, eikä puutteita löydy.

Testattavuus ei ole täysin verrattavissa RPA:n ja ohjelmistokehityksen välillä, sillä RPA:ssa testejä ei suoriteta testausohjelmilla, vaan testaus on täysin riippuvainen testitapauksista ja niiden monipuolisuudesta. Testitapaukset taas riippuvat siitä, millaisia testitapauksia liiketoiminta testaajille toimittaa. Testattavuus ei ole siis täysin soveltuva mittari analysoitavaksi RPA:n ja ohjelmistokehityksen välille.

Siirrettävyys on myös mittari, joka ei ole soveltuva RPA:han. Robotit pyörivät vain sillä alustalla, mille ne ovat tehty, toisin kuin ohjelmistot, joissa voi olla tuki eri alustoihin.

Luotettavuus otetaan konsernin ohjeistuksessa hyvin huomioon. Robottien tulee täyttää niille asetetut vaatimukset, ja robotille tulevan datan tarkistus otetaan ohjeistuksissa huomioon. Tuotantoympäristössä ei ohjeita noudattamalla ole Blue Prismin omalla laadunseurannalla kiinni jääviä kehitysvirheitä.

Uudelleenkäytettävyydestä mainitaan konsernin ohjeistuksessa useassa eri kohdassa. Uudelleenkäytettävyyden hyödyntäminen on olennainen osa ohjeistuksia, ja ohjeistukset ovat järkeviä tietoperustaan peilaten.

Turvallisuus otetaan konsernin ohjeistuksessa huomioon. Asiakassalaisuudesta pidetään kiinni, eivätkä ulkopuoliset henkilöt pääse dataan käsiksi. Valtuudet on rajattu tiukasti. Asiakassalaisuuden ylläpitämiseksi on kehitetty avustavia työkaluja. Ohjeistus on linjassa tietoperustassa esitellyn kanssa. Ohjeista löytyi kuitenkin yksi ylimääräiseksi luettava vaihe. Automaattisella lokitustyökalulla robotin lokitukset tulevat aina säännösten mukaisiksi, mutta silti vaaditaan toiselta kehittäjältä varmistus sille, että lokitukset ovat säännösten mukaiset. Lokitustyökalu luo joka tapauksessa robotin kuvaukseen merkinnän siitä, että se on ajettu lokitustyökalun läpi, mikä tekee ylimääräisen varmistuksen hyödystä kyseenalaisen.

Tehokkuudesta ei konsernin ohjeistuksissa ole erikseen mainintaa. Ohjeita tarkastellessa kuitenkin selviää, että tehokkuuteen pyrkiminen on huomaamattomana osana monia ohjeistuksia. Tehokkuuteen pyrkiminen on linjassa tietoperustan kanssa.

Laadun parantaminen, tekninen velka sekä ketteryys

Konsernin ohjeistuksessa laadun parantaminen on oleellisena osana. Robotin tuotantoympäristöön viennissä on useita tarkistuksia, sekä Design Boardin tuki katselmointeineen kehitysvaiheessa kehittäjillä että muiden kehittäjien vertaistuki.

Teknisen velan pienentäminen tapahtuu kuitenkin kyseenalaisella tavalla. Mikäli robottiin tehdään pienikin muutos, koko robotti pitää olla kaikilta osiltaan uusien standardien mukainen. Tämä ei ole tietoperustassa esitellyn mukaista. Tietoperustassa opimme, että pienemmissä muutoksissa mahdollisten virheiden huomaaminen on helpompaa.

Pienien muutosten kannalta robotiikkaa pystyy tekemään ketterästi, muutoksen tuotantoympäristöön viennin ajan ollessa vain alle yksi päivä. Suurempia muutoksia vietäessä aika on kaikkine tarkastuksineen kuitenkin jopa kolme päivää, mikä kuulostaa pitkältä ketterää toimintaa ajatellen.

Ohjeistusten arviointi yritys X:n kannalta

Haastattelujen perusteella konsernin ohjeiden teknisen kehityksen ohjeistukset ovat asianmukaiset yritys X:n kannalta. Ohjeistuksia tulisi kuitenkin jaotella niin, että etsittävä tieto olisi helpommin löydettävissä. Ohjeiden ajan tasalla pitämistä tulisi parantaa.

Haastatteluissa eniten esiin nousseet tarkistukset ja hyväksynnät ovat raskaita ketterää toimintatapaa hyödyntävälle yritys X:lle. Ohjeistuksia olisi hyvä arvioida uudelleen esimerkiksi Design Boardin osalta. Yritys X:n robotiikkatekemisen ketterän luonteen vuoksi olisi järkevää miettiä, voisiko yritys X:llä olla oma Design Board, joka pohjautuisi mahdollisesti yrityksen omiin ohjeistuksiin ja jonka avulla toimintaa saisi tehokkaammaksi ja ketterämmäksi. Näin myös robotiikkatekemisen vastuuta saataisiin siirrettyä yritys X:lle itselleen, mikä on ollut yrityksellä toiveena. Tällä hetkellä Design Boardin ei tarvitse kantaa vastuuta tarkistuksistaan. Oman Design Boardin myötä yrityksellä olisi kuitenkin iso vastuu teknisestä laadusta ja ohjeistusten mukaisuudesta. Tällainen ketterä toiminta yrityksen sisällä toisi myös toivottua yhteneväisyyttä yrityksen ketterän ohjelmistokehityksen kanssa.

5.4 Yhteenveto

Yritys X:lle soveltuva malli olisi hyvin samankaltainen konsernin ohjeistuksien kanssa, konsernin mallin jo laajalti hyödyntäessä ohjelmistokehityksen laadun tekijöitä. Design Board -logiikka muuttuisi siten, että yrityksellä olisi oma Design Board, joka olisi vastuussa teknisestä laadusta ja ohjeistusten mukaisuudesta. Muutoksen tuotantoympäristöön vientiin tulisi muutos, joka poistaisi tarpeen toisen kehittäjän lokitustarkistukselle. Yritys X:n

mallissa vaatimus siitä, että koko robotti pitäisi muokata uusien ohjeistusten mukaiseksi pienenkin muutoksen takia, poistuisi. Tilalle tulisi vaatimus siitä, että se muokattaisiin uusien ohjeistusten mukaiseksi vain osioiltaan, joihin on tehty muutoksia. Myös teknisen kehityksen ohjeistukset muuttuisivat seuraavilta osin:

- Data Itemien nimeämiskäytäntö muuttuisi Blue Prismin oletusta vastaavaksi.
- Ohjeistuksiin tulisi suositus lisätä kommentteja, eli noteja kehitystyön helpottamiseksi.
- Turhista parametrien ja toimintojen kuvauksista luovuttaisiin.

6 Pohdinta

Tutkimuksen pääkysymyksenä oli saada selville, miten robotiikkakehittämisen tekninen laatu ja laatuvaatimukset tulisi kuvata henkivakuutusyhtiön näkökulmasta. Pääkysymykseen on saatu vastaus. Myös kaikkiin alakysymyksiin on saatu vastaus. Opinnäytetyön alakysymykset olivat:

1. Mitä kehityksessä tulee ottaa huomioon kehittämisen tehokkuuden näkökulmasta?
2. Mitä tulee ottaa huomioon robottien käytön elinkaaren aikaisten virheselvitysten näkökulmasta?
3. Mitä tulee ottaa huomioon robottien jatkokehityksen näkökulmasta?

Ensimmäiseen kysymykseen vastaus muodostui ketterän kehityksen periaatteista, joita yritys X:ssä jo harjoitetaan. Toiseen kysymykseen vastaus muodostui lokituksesta, jonka avulla virheiden selvitys tapahtuu. Kolmannen kysymyksen vastaus muodostui uudelleenkäytettävyydestä ja modulaarisuudesta.

6.1 Johtopäätökset sekä tutkimuksen kehittämis- ja jatkotutkimusehdotukset

Suositukseni on, että yritys X ottaisi käyttöönsä oman ohjeistuksen/mallin tässä työssä esiteltyjen ehdotuksien pohjalta. Ehdotukset ottavat huomioon ohjelmistokehityksen laadun mittarit, sekä yritys X:n omat vaatimukset. Muutokset ovat kooltaan ja vaikutuksiltaan sen luokkaisia, että ehdotusten käyttöönotto on perusteltavaa. Vaikutukset yritys X:n toimintaan olisi selkeästi nähtävissä muutosten ja korjausten nopeamman tuotantoympäristöön viennin myötä. Myös kehityksen laatu paranisi teknisen kehityksen ohjeistusten muutosten avulla.

Jatkotutkimusta ajatellen operointimallin muistakin vaiheista olisi hyvä tehdä tutkimusta, sillä tässä työssä on keskitytty vain Configure-vaiheeseen. Tässä työssä saatuja tuloksia pystyy hyödyntämään yritys X:n kaltaisessa ohjelmistorobotiikkaa ja ketterää toimintatapaa hyödyntävässä yrityksessä, jonka strategia on samankaltainen yritys X:n kanssa.

6.2 Tutkimuksen luotettavuus sekä opinnäytetyön ja oman oppimisen arviointi

Tutkimuksessa käytetyissä lähteissä jouduin vallitsevan koronavirustilanteen vuoksi tyytymään suurelta osin internet-lähteisiin. Lähteissä olen kiinnittänyt huomiota siihen, onko lähteenä oleva henkilö oikeasti olemassa, millainen kokemus hänellä on, sekä vältellyt yritysten markkinointimateriaaliksi luokiteltavia lähteitä. Blue Prism -luvussa hyödynsin lähteenä kollegoiden henkilökohtaisia tiedonantoja, sillä heillä on omakohtaista kokemusta kyseisestä ohjelmistosta. Työskentelen myös itse Blue Prismin kanssa päivittäin, jonka

vuoksi tietojen todenperäisyys oli helppoa todentaa. Lähteinä toimivat YouTube-videot ovat helppoja kenen tahansa todentaa oikeaksi, sillä videot ovat nauhoitteita ruudulla näkyvästä ohjelmasta. Videoista on siis helppoa nähdä ja kuulla tekstissä todetut asiat. Ohjelmiston laadun teoriassa olen hyödyntänyt alan sivustoja, ISO-standardia, sekä alan ammattilaisia. Lukijan on myös hyvä ottaa huomioon, että työskentelen yritys X:ssä ja työn tulokset voivat vaikuttaa omaan työhöni.

Olen oppinut työn tekemisen aikana tekemiseni aikatauluttamista ja mukautumista erilaisiin tilanteisiin. Aineiston hankintatapa muuttui työn edetessä havainnoinnista haastatteluun. Olen myös oppinut työn tekemisen myötä kirjoittamaan selkeämpää tekstiä. Tämän lisäksi olen oppinut paljon ohjelmistokehityksen laadusta ja sen teorioista.

Mielestäni tietoperusta tukee riittävästi empiiristä osuutta, enkä muuttaisi tietoperustaa millään tavalla näin jälkeenpäin mietittynä. Mikäli aikataulu tämän työn tekemiselle ei olisi ollut niin tiukka, olisin tehnyt empiirisen osuuden haastattelut ryhmähaastatteluperiaatteella. Haastattelu olisi ollut teemahaastattelu. Uskon, että olisin voinut saada tämänkaltaisella haastattelutavalla vielä syvällisempää ja tarkempaa tietoa robotiikasta yritys X:ssä.

Haluan lopuksi kiittää kollegoitani tuesta opinnäytetyön tekemisen aikana. Haluan myös erityisesti kiittää esimiestäni tuesta ja neuvoista opinnäytetyön tekemisen aikana.

Lähteet

Azets. 2020. Ohjelmistorobotiikka. Luettavissa: https://www.azets.fi/ohjelmistopalvelut/ohjelmistorobotiikka/. Luettu: 10.4.2020.

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Martin, R., Mellor, S., Thomas, D., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Schwaber, K., Sutherland, J. 2001a. Ketterän ohjelmistokehityksen julistus. agilemanifesto. Luettavissa: https://agilemanifesto.org/iso/fi/manifesto.html. Luettu: 11.5.2020.

Beckm, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Martin, R., Mellor, S., Thomas, D., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Schwaber, K., Sutherland, J. 2001b. 12 Principles Behind the Agile Manifesto. Agile Alliance. Luettavissa: https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/. Luettu: 11.5.2020.

Bellairs, R. 10.2.2020. What Is Static Analysis? And What Is Static Code Analysis? Perforce. Luettavissa: https://www.perforce.com/blog/sca/what-static-analysis. Luettu: 14.5.2020.

Bellairs, R. 11.7.2019. What Is Code Quality? And How to Improve it. Perforce. Luettavissa: https://www.perforce.com/blog/sca/what-code-quality-and-how-improve-it. Luettu: 2.5.2020.

Bellairs, R. 15.7.2019. What Is Secure Coding? Perforce. Luettavissa: https://www.perforce.com/blog/sca/what-secure-coding. Luettu: 15.5.2020.

Bellairs, R. 28.11.2018. How Static Code Analysis Works. Perforce. Luettavissa: https://www.perforce.com/blog/qac/how-static-code-analysis-works. Luettu: 14.5.2020.

Bellairs, R. 7.7.2017. The Challenge of Code Reuse (And How to Reuse Code Effectively). Perforce. Luettavissa: https://www.perforce.com/blog/qac/challenge-code-reuse-and-how-reuse-code-effectively. Luettu: 6.5.2020.

Blue Prism 2018. Intranet. Logging Best Practises Guide. PDF. Luettu 28.4.2020.

Britton, J. 22.10.2016. What Is Cyclomatic Complexity? Perforce. Luettavissa: https://www.perforce.com/blog/qac/what-cyclomatic-complexity. Luettu: 3.5.2020.

Busy Ping 2017a. Blue Prism Video Tutorial | 036 | Environment and session variables. Video. Katsottavissa: https://youtu.be/wVpJclKSLeI. Katsottu: 24.4.2020.

Busy Ping 2017b. Blue Prism Video Tutorial | 014 | Collection and Loop Stages. Video. Katsottavissa: https://youtu.be/lxZLdPOOnZ4. Katsottu: 24.4.2020.

Busy Ping 2017c. Blue Prism Video Tutorial | 039 | Work Queues - Tag and Tag filter. Video. Katsottavissa: https://youtu.be/hytYaVDvZ44. Katsottu: 24.4.2020.

Busy Ping 2017d. Blue Prism Video Tutorial | 040 | Work Queues – Priority. Video. Katsottavissa: https://youtu.be/J3BauvskEUg. Katsottu 24.4.2020.

Busy Ping 2017e. Blue Prism Video Tutorial | 011 | Calc, Data item and multi calc stages. Video. Katsottavissa: https://youtu.be/690vk1P3m8g. Katsottu 25.4.2020.

Busy Ping 2017f. Blue Prism Video Tutorial | 029 | Work Queues - Part 1. Video. Katsottavissa: https://youtu.be/2Q-B00TqrRI. Katsottu 26.4.2020.

Busy Ping 2017g. Blue Prism Video Tutorial | 030 | Work Queues - Part 2. Video. Katsottavissa: https://youtu.be/MUjbuZIRXrQ. Katsottu 26.4.2020.

Codacy 14.3.2016. Code Review vs. Testing. Codacy. Luettavissa: https://blog.codacy.com/code-review-vs-testing/. Luettu: 15.5.2020.

Codegrip s.a.a. What is code quality, how to measure and improve it? Luettavissa: https://www.codegrip.tech/productivity/what-is-code-quality-how-to-measure-and-improve-it/. Luettu: 2.5.2020.

Codegrip s.a.b. Why is code quality important? Luettavissa: https://www.codegrip.tech/productivity/why-is-code-quality-important/. Luettu: 15.5.2020.

Digital Workforce 2020. OHJELMISTOROBOTIIKKA (RPA). Luettavissa: https://digitalworkforce.com/fi/digityontekija/rpa-ohjelmistorobotiikka. Luettu 23.4.2020.

DMS-Solutions s.a. Luettavissa: https://dms-solutions.co/blog/scheduling-uipath-robot-the-robot-trigger-tool/. Luettu: 7.4.2020.

Gartner 2019. Luettavissa: https://www.gartner.com/en/newsroom/press-releases/2019-06-24-gartner-says-worldwide-robotic-process-automation-sof. Luettu: 31.3.2020.

Gates, L. 26.2.2020. What an RPA Developer Does. Automation Anywhere. Luettavissa: https://www.automationanywhere.com/au/company/blog/company-culture/what-an-rpa-developer-does. Luettu: 18.5.2020.

Gergely, O. 27.12.2019. Readable Code. The Pragmatic Engineer. Luettavissa: https://blog.pragmaticengineer.com/readable-code/. Luettu: 5.5.2020.

Grand View Research 2020. Luettavissa: https://www.grandviewresearch.com/press-release/global-robotic-process-automation-rpa-market. Luettu: 31.3.2020.

Helmkamp, B. 2016. Confreaks. GORUCO 2016 - Keynote: Code Quality Lessons Learned Bryan Helmkamp. Video. Katsottavissa: https://youtu.be/vcH0RBe4Eew. Katsottu: 2.5.2020.

Helsingin Yliopisto 2019. Muuttujat ja ohjelmien kielellistäminen. Luettavissa: https://ohjelmointi-19.mooc.fi/osa-1/3-muuttujat. Luettu: 24.4.2020.

Holvitie, J., Licorish, S., Spínola, R., Hyrynsalmi, S., MacDonell, S., Mendes, T., Buchan, J., Leppänen, V. 2017. Technical debt and agile software development practices and processes: An industry practitioner survey. PDF. Luettavissa: https://reader.elsevier.com/reader/sd/pii/S0950584917305098?token=5BEA55B26CA37D648342190309B40B427E40483BBA9069225D77837B36FAD210405D29C15E1D808E7113670A41D65C88. Luettu: 12.5.2020.

ISO/IEC 25010:2011. International Organization for Standardization & International Electrotechnical Commission. Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – System and software quality models.
Jackson, M. s.a. Writing readable source code. Sustainability Institute. Luettavissa: https://software.ac.uk/resources/guides/writing-readable-source-code. Luettu: 15.5.2020.

Jayasinghe, S. 16.7.2018. Medium. The Importance of Code Quality. Luettavissa: https://medium.com/emblatech/the-importance-of-code-quality-ac7afa598c0d. Luettu: 2.5.2020.

King, R. 19.2.2017. Developing an RPA Centre of Excellence: Delivering the promise of Robotic Automation. Luettavissa: https://www.linkedin.com/pulse/developing-rpa-centre-excellence-delivering-promise-robotic-rob-king/. Luettu: 16.4.2020

Koski, J. s.a. Ketterät menetelmät, agile, LEAN ja scrum. iteWiki. Luettavissa: https://www.itewiki.fi/opas/ketterat-menetelmat-agile-lean-ja-scrum/. Luettu: 11.5.2020.

Labbe, J. 28.1.2017. RPA Projects - Roles and Responsibilities. Luettavissa: https://www.linkedin.com/pulse/rpa-projects-roles-responsibilities-joe-labbe/. Luettu: 19.4.2020.

Learning Technologies 2018a. (BluePrism) Part 30 : Working with Session variables in Control Room. Video. Katsottavissa: https://youtu.be/95U1zNOA1Xg. Katsottu 24.4.2020.

Learning Technologies 2018b. (BluePrism)Part 33 : WORK QUEUES || Defer Items in Queue Management. Video. Katsottavissa: https://youtu.be/GUQiFopMepM. Katsottu: 22.4.2020.

Learning Technologies 2018c. (Blue Prism) Part 23 : Working With CODE Stage in Object Studio. Video. Katsottavissa: https://youtu.be/n3tKmNUKYD0. Katsottu: 24.4.2020.

Learning Technologies 2018c. (BluePrism)Part 34 : Working with Environment Variables. Video. Katsottavissa: https://youtu.be/OMIcUh5TiJ4. Katsottu: 24.4.2020.

Learning Technologies 2018d. (BluePrism)Part 38 : Working Session with SCHED-ULES,TASKS AND TIMETABLE under Control Room(Detailed). Video. Katsottavissa: https://youtu.be/ns9VhzUyzfw. Katsottu 25.4.2020.

Learning Technologies 2018e. (BluePrism) Part 35 : WORK QUEUES || Priority in Queue Management (Detailed). Video. Katsottavissa: https://youtu.be/LH7ucEHhz0E. Katsottu 24.4.2020.

Leinonen, R. 12.12.2018. Sisällönanalyysi. Spoken. Luettavissa: https://spoken.fi/sisal-lonanalyysi/. Luettu: 19.5.2020.

Pro RPA 2018. 08 Input Parameters. Video. Katsottavissa: https://youtu.be/s6ClBa5sOCw. Katsottu: 26.4.2020.

ProductPlan 2020. Technical Dept. Luettavissa: https://www.productplan.com/glossary/technical-debt/. Luettu: 19.4.2020.

PVS-Studio s.a.a. Code Optimization. Luettavissa: https://www.viva64.com/en/t/0084/. Luettu: 9.5.2020.

PVS-Studio s.a.b. Dynamic code analysis. Luettavissa: https://www.viva64.com/en/t/0070/. Luettu 14.5.2020.

QISQ s.a. CODING RULES FOR RELIABLE SOFTWARE. Luettavissa: https://www.it-cisq.org/standards/code-quality-standards/reliability/index.htm. Luettu: 4.5.2020.

Radigan, D. s.a. Why code reviews matter (and actually save time). Luettavissa: https://www.atlassian.com/agile/software-development/code-reviews. Luettu: 15.5.2020.

Rebeiro, S. s.a. A Robotics Center of Excellence – Key Considerations. WNS Insights – Blogs. Luettavissa: https://www.wns.com/insights/blogs/blogdetail/531/a-robotics-centre-of-excellence-key-considerations. Luettu: 12.4.2020.

Roboyo 2019. Luettavissa: https://www.roboyo.de/en/blog/the-role-of-the-rpa-business-analyst-the-hidden-champion-of-the-rpa-world/. Luettu: 19.4.2020.

RPA-kehittäjä. Yritys X. 27.4.2020. Haastattelu. Helsinki.

RPA-prosessinomistaja. Yritys X. 27.4.2020. Haastattelu. Helsinki.

Sadangi, M. 11.10.2017. 10 Tips for Improving the Readability of Your Code. DZone. Luettavissa: https://dzone.com/articles/10-tips-how-to-improve-the-readability-of-your-sof. Luettu: 26.4.2020.

Sealights s.a. Code Quality Metrics: Is Your Code Any Good? Luettavissa: https://www.sealights.io/code-quality/code-quality-metrics-is-your-code-any-good/. Luettu: 15.5.2020.

Staria 2020. Mitä on ohjelmistorobotiikka? Luettavissa: https://staria.com/fi/blogi/mita-ohjelmistorobotiikka/. Luettu: 25.4.2020.

Stoddard, C. 30.10.2019. The Enterprisers Project 2019. Adobe CIO: How we scaled RPA with a Center of Excellence. Luettavissa: https://enterprisersproject.com/article/2019/10/rpa-robotic-process-automation-how-build-center-excellence. Luettu: 12.4.2020.

Takata, M. 2013. 2. Maintainability depends on modularity: Stop using namespaces! Luettavissa: http://singlepageappbook.com/maintainability1.html. Luettu: 15.5.2020.

Tietohallintojohtaja. Yritys X. 7.5.2020. Haastattelu. Helsinki.

Tutorialspoint s.a.a. Blue Prism - Introduction to RPA Luettavissa: https://www.tutorialspoint.com/blue_prism/blue_prism_quick_guide.htm. Luettu 7.4.2020.

Tutorialspoint s.a.b. Blue Prism - Exceptions Handling. Luettavissa: https://www.tutorialspoint.com/blue_prism/blue_prism_exceptions_handling.htm. Luettu: 28.4.2020.

Tutorialspoint s.a.c. Email and Surface Automation. Luettavissa: https://www.tutorialspoint.com/blue_prism/blue_prism_email_surface_automation.htm Luettu: 28.4.2020.

UiPath s.a.a. Build your Center of Excellence. Luettavissa: https://www.uipath.com/rpa/center-of-excellence. Luettu: 12.4.2020.

UiPath s.a.b. Invoke Code. Luettavissa: https://docs.uipath.com/activities/docs/invoke-code. Luettu: 24.4.2020.

UiPath s.a.c. Desktop Automation Software – RPA. Luettavissa: https://www.uipath.com/solutions/technology/desktop-automation. Luettu: 25.4.2020.

Wells, C. 30.5.2019. Senior Product Consultant. Control Room: everything you need to know (and more). Blue Prism. Webinaariesitys. Katsottavissa: https://www.blueprism.com/resources/webinars/control-room-everything-you-need-to-know-and-more/. Katsottu: 19.4.2020

VIVEK GOEL 2017. RPA-BP-03: Blue Prism Process Studio. Video. Katsottavissa: https://youtu.be/A6MGFYa9lZ0. Katsottu 27.4.2020.

Vivek, G. 8.1.2019. Dynamic Process Execution. RPATools.com. Luettavissa: https://rpa-tools.com/2019/01/dynamic-process-execution-blue-prism/. Luettu: 28.4.2020.

XBOSoft s.a. Definition of Software Quality – What is Software Quality? Luettavissa: https://xbosoft.com/definition-software-quality/. Luettu: 2.5.2020.

Ying, L. 2018. Robotic Process Automation with Blue Prism Quick Start Guide. Packt Publishing. Birmingham. Luettavissa: https://learning.oreilly.com/library/view/robotic-process-automation/9781789610444/f1d0dda9-50d8-444b-9f22-f09bd1cfbdfe.xhtml. Luettu: 7.5.2020.

Liitteet

Liite 1. Haastattelujen vastaukset

Kysymys 1 - Miten kuvailisit robotiikkatekemistä yritys X:ssä?

- Robotiikka on yrityksen kannalta keskeinen osaamisalue. Robotiikkaa käytetään liiketoimintaprosessien tukena automatisoimalla prosesseja ja niiden osia/työvaiheita. Isossa kuvassa yrityksen tavoitteena on jatkuvasti kehittää vakuutusjärjestelmää niin, että asiakastoimeksiantoihin liittyvä ihmisen tekemä työ vähenee. Tällöin on tärkeätä pystyä ketterästi sovittamaan robotteja tähän kehitykseen. Robotiikkaosaaminen on tärkeätä olla itsellä, koska prosessien tarpeet elävät jatkuvasti ja suuri hyöty saadaan nopeasti tehtävistä ja sovitettavista roboteista.
- Strategian mukaisesti tekemisen on tarkoitus olla ketterää ja nopeaa tuottaen laadukkaita ratkaisuja. Toiminta ja tekeminen on hyvin läheisesti sidoksissa liiketoiminnan eri osa-alueisiin, mikä on olennaisen tärkeää ketterän ja nopean palvelun tuottamiseksi. Melko voimakkaasti säännellyllä toimialalla pyrkimykset on pystytty täyttämään verrattain hyvin.
- Robotiikkatekeminen on meillä ketterää ja joustavaa. Työskentelemme läheisessä yhteistyössä liiketoiminnan kanssa ja toimimme järjestelmän kehityksen tukena. Prosesseihin tulee usein muutoksia, joihin pystytään vastaamaan nopeasti.
- Ketterää, tukee järjestelmäkehitystä ja asiakaspalvelua. Läheinen yhteistyö ”tilaajan” kanssa. Prosessit muuttuvat jatkuvasti ja ulkopuoliset tekijät vaikuttavat toimialaan ja niihin pystytään vastaamaan nopealla aikataululla.

Kysymys 2 - Mikä on yleinen kuvasi konsernin ohjeistuksista, niiden selkeydestä ja hyödyllisyydestä?

- Perustavoite on hyvä, ja konsernin yhteisen robotiikan tuotantoalustan hyödyntäminen on helpompaa, kun perusmallit ovat yhteiset. Ohjeistuksen kehittämiseen on panostettu (Sharepoint-sivustot, mallipohjat).
- Haaste yrityksen kannalta on siinä, että konsernin isojen robotiikkahankkeiden tarpeisiin kehitetty malli näyttäytyy raskaana ja liian monia ulkopuolisia tarkistuksia sisältävänä yrityksen tapaan hyödyntää robotiikkaa.
- Konsernin ohjeet ovat kattavat ja kehittyneet maturiteetin kasvaessa. Ohjeistukset on kerätty yhteen paikkaan, mistä ne on kaikki löydettävissä, joskin kasvava määrä ohjeita aiheuttaa sen, että toisinaan etsiminen on hieman työlästä. Pääosin ohjeet ovat selkeitä ja hyviä, mutta mittakaava- ja näkemyserot konsernin ja tytäryhtiön välillä aiheuttavat ristiriitoja strategian mukaisen toimintatavan toteuttamiseen. Ohjeistuksesta löytyy myös hyödyllisiä ns. alan parhaita käytäntöjä tukevia ohjeistuksia sekä vaatimuksia.
- Monet konsernin laatimat ohjeet ovat selkeitä ja hyödyllisiä ja ne on tallennettu yhteen paikkaan. Välillä niitä on kuitenkin vaikea löytää, koska ohjeistuksia on niin paljon. Joidenkin ohjeiden päivityksessä on ollut viiveitä, joka luonnollisesti vaikuttaa niiden hyödyllisyyteen negatiivisesti, eikä pysty luottamaan onko niissä oleva tieto ajan tasalla. Usein on vaikea soveltaa ohjeistuksia pienen tiimin tekemiseen, koska ne on laadittu isomman yksikön näkökulmasta ja toisaalta sisältävät myös paljon yksityiskohtaista tietoa.
- Ohjeet ovat selkeästi laadittu ja ovat yhdessä ja samassa paikassa saatavilla. Ohjeiden päivityksessä ja ajan tasalla pitämisessä on aukkoja.

Kysymys 3 - Mitkä asiat konsernin ohjeistuksessa nopeuttavat/hidastavat asioita?

- Nopeuttavat: vaihejako, pääosa mallipohjista, robottien tuotantoon viemisen tekniset ratkaisut.

- Vaikeuttavat: paikallisiinkin yksityiskohtiin menevä tekninen tarkastus/hyväksymisprosessi.
- Edellä mainitut vaatimukset parhaista käytänteistä edesauttavat nopeutta ja laadullisia seikkoja. Lisäksi standardisoidut kaaviot helpottavat uusien työntekijöiden perehdyttämistä sekä vanhojen tekijöiden perehdyttämistä uusiin robotisoituihin prosesseihin ja niiden teknisiin kaavioihin.
- Hidastavia asioita ovat mittakaava- ja näkökulmaeroon liittyviä. Monet ohjeistukset eivät sellaisenaan toimi pienelle tiimillemme, jonka tavoitteena on ketterä toiminta ja robotiikkaprosessien käsittelemät määrät eivät ole isoja.
- Ohjeet ovat yleispätevät ja laadittu silmälläpitäen konsernin robotiikkatekemistä. Pankin puolella tekeminen on suurempaa ja busineksen tilaaja toimii täysin eri organisaation osassa kuin robotiikan tekijät. Moni prosessi on liian raskas meille tytäryhtiössä, jossa toimitaan hyvin tiiviisti liiketoiminnan kanssa yhteistyössä ja vastuu säilyy tuotantoon oton jälkeenkin meillä tekijöillä.

Kysymys 4 - Mitä konsernin ohjeistuksen osuuksia on helppo seurata/toteuttaa? Entä vaikea seurata/toteuttaa?

- Helppoa: vaihejako, terminologia, robottien tuotantoon viennin tekniikka.
- Vaikeaa: tekniseen tarkastukseen/hyväksymiseen kuluva aika ja siitä tehtäväksi tulevat ”muotoseikkojen” muutokset.
- Tytäryhtiön toimintamallia tukevia ohjeistuksia on helpointa toteuttaa. Sen sijaan suuresta mittakaavasta aiheutuvia varmistuksia ja tarkistuksia on nopeaan ja ketterään toimintaan pyrkiessä haasteellista toteuttaa. Lisäksi joissain kohdissa ohjeistuksen vaatimat ulkopuolisen päätöksentekijän päätökset, mistä itse päätöksentekijän ei tarvitse kantaa vastuuta, ovat haasteellisia toteuttaa.
- Konsernin tasolta meille asetettujen menetelmien vaatimusten toteuttaminen on usein haastavaa, ja se vaikeuttaa pyrkimystämme ketterään toimintatapaan. Menetelmien muutosten seuraaminen ja toteuttaminen on haastavaa, koska tietoa tulee eri kanavista melko nopeassa tahdissa ja ne vaativat usein soveltamista omaan toimintaan sopivaksi edellä mainittujen mittakaava- ja näkökulmaerojen vuoksi.
- Helppoa on esimerkiksi tietynlaisten parhaiden käytäntöjen ja design-asioiden toteuttaminen. Tämä on myös ihan hyödyllistä robotiikkaprosessien helppolukuisuuden edistämiseksi. Vaikeaa on liian monimutkaiset dokumentointiasiat, joita tekiessä tuntuu, että tehdään voin tekemisen ilosta (ei mitään käytännön hyötyä).

Kysymys 5 - Onko yritys X:n robotin kehityksessä ja tuotantoon viennissä osuuksia, joihin konsernin ohjeistuksesta ei ole apua?

- Ketterän kehitysmallin tuki, esim. kehityksen ja virhekorjausten läpimenoaikojen seuranta.
- Konsernin ohjeistus luo toiminnalle raamit, mutta juurikin edellisissä kohdissa mainitut näkökulmaeroista ja vastuun jakautumisesta koituvat seikat eivät edesauta tuotantoon viemistä.
- Controllerin näkökulmasta tuotantoon viennin ohjeistukset ovat riittävän selkeät.
- Kts edellinen vastaus. Esimerkkinä Design Board ja jotkut pakolliset dokumentit.

Kysymys 6 - Onko yritys X:llä omia ohjeita/tehtäviä/vaiheita konsernin ohjeistuksen lisäksi tai niitä täydentämässä?

- Ketterän kehityksen läpimenoajan tarkastelu.
- Kuten edellä mainittu, konsernin ohjeistus luo toiminnalle raamit, mutta niitä pyritään aktiivisesti sovittamaan paremmin tytäryhtiölle sopiviksi.

- Olemme pyrkineet löytämään oman tavan tehdä ohjelmistorobotiikkaa yhtiössämme. Esimerkiksi uusien roboteilla automatisoitavien kohteiden soveltuvuuden arviointiin käytämme konsernin työkalun sijasta keskustelufoorumia.

Tietohallintojohtajalle esitetyt lisäkysymykset:

Kysymys 7 - Miten yritys X toteuttaa tällä hetkellä ohjelmistokehitystä?

- Yritys vastaa itse keskeisen vakuutusjärjestelmäalustan kehityksestä ja ylläpidosta. Yrityksellä on omat tekniset ydintiimit ohjelmistokehityksen ja tuotannon tukeen, ja iso osa tekemiskapasiteetista on ulkoistettu intialaiselle yhteistyökumppanille pohjautuen konsernin puitesopimusmalliin. Ohjelmistokehitys toteutetaan ketterän kehityksen malleilla käyttäen SAFe (Scaled Agile Framework) viitekehystä. ”Kehitysjunassa” (Agile Release Train) on tällä hetkellä seitsemän kehitystiimiä ja systeemitiimi. Yrityksen ohjelmistokehitysprosessi pohjautuu ketterän kehityksen scrum/agile-malliin ja SAFe-viitekehykseen ja yhden vakuutusjärjestelmäalustan sovellusarkkitehtuuriin. Vakuutusjärjestelmä on Java-pohjainen palvelupohjainen järjestelmä. Ohjelmistokehityksen laadun kehittäminen perustuu scrum-mallin tehtävien aloitus- ja hyväksymiskriteerien määrittelyn (Definition of Ready, Definition of Done), kehityksen aikaiseen koodin tarkastukseen (code review) ja teknisen analyysin työkalujen käyttöön. Tuotantoon menneiden virheiden määrää seurataan ja tavoite on pienentää määrää jatkuvasti. Tavoitteena on myös pienentää koodin teknistä velkaa (teknisen analyysin työkaluilla laskettuna) jatkuvasti.

Kysymys 8 - Mikä on yritys X:n näkemys robotiikan hyödyntämisestä?

- Robotiikka on yritykselle strateginen osaamisalue ja yksi tapa toteuttaa yrityksen strategista tavoitetta olla ”modernein ja ketterin suomalainen henkivakuutusyhtiö”. Robotiikka nähdään ketteränä transitioteknologiana, mikä tarkoittaa sitä, että ohjelmistoroboteilla haetaan nopeaa apua liiketoimintaprosessien ihmistyötä vaativiin vaiheisiin myös yksittäisissä prosesseissa ja työvaiheissa. Tavoitteena on, että sellaisten työvaiheiden automatisointi toteutetaan liiketoimintaprosessin vakiintuessa perusjärjestelmään, mutta robotti pystyy tuottamaan liiketoimintahyötyä siihen saakka. Roboteilla voidaan myös tukea perusjärjestelmien kehityksen vaiheistusta niin, että robotit tukevat ja täydentävät perusjärjestelmien toiminnallisuutta. Robotteja käytetään myös lyhytjaksoisemmissa tehtävissä kuten eri tilanteisiin räätälöidyissä asiakkaiden kontaktoinnin tuessa.

Liite 2. Yritys X:n konsernin RPA-operointimalli